Technische Universität München
Fakultät für Maschinenwesen
Lehrstuhl für Leichtbau

Efficient Procedures for Structural Optimization with Integer and Mixed-Integer Design Variables

Yang Zhang

Vollständiger Abdruck der von der Fakultät für Maschinenwesen der Technische Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

Genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr.-Ing. Oskar J. Haidn

Prüfer der Dissertation:

1. Univ.-Prof. Dr.-Ing. Horst Baier
2. Univ.-Prof. Dr.-Ing. Kai-Uwe Bletzinger

Die Dissertation wurde am 08.01.2015 bei der Technische Universität München eingereicht und durch die Fakultät für Maschinenwesen am 06.05.2015 angenommen.

Abstract

Structural optimization with pure integer and mixed-integer design variables are important topics in structural design. General gradient-based mathematical optimization algorithms fail to solve these problems because they usually do not treat integer variables. Evolutionary methods or genetic algorithms handling such problems may be computationally prohibitively expensive due to the requirement of large number of time-consuming functions evaluations, which are often carried out by finite element analysis with a large number of degrees of freedom. To more efficiently solve such problems, several procedures are developed and improved in this dissertation. A procedure to generate a high quality starting point for the nonlinear assignment problem is developed. In addition, traditional neighbor search algorithms then are investigated and improved. A constraints-directed algorithm is developed for mixed-integer nonlinear optimization. An enforced neighbor search strategy is also presented to improve the quality of the solution. These procedures aim at obtaining a high quality solution with significantly reduced number of function evaluations than algorithms at hand. The efficiencies of the developed methods are demonstrated through three practical engineering problems. These are optimal parts allocation problem, the stacking sequence optimization in laminate design and the simultaneous material selection and size optimization problem.

Kurzfassung

Strukturoptimierungen mit reinen ganzzahligen und gemischt-ganzzahligen Designvariablen sind wichtige Themen im Strukturentwurf. Mit allgemeinen gradientenbasierten mathematischen Optimierungsalgorithmen, es gelingt nicht, diese Probleme zu lösen, weil sie in der Regel nur kontinuerlige Variable behandeln. Evolutionäre Methoden oder genetische Algorithmen sind in der Handhabung solcher Probleme rechnerisch aufwendig infolge der erforderlichen großen Anzahl von zeitraubenden Funktionsauswertungen, die häufig von Finite-Elemente-Analyse mit einer großen Anzahl von Freiheitsgraden durchgeführt werden müssen. Um derartige Probleme effizienter zu lösen, werden verschiedene Verfahren untersuchte und entwickelt. Es wird ein Verfahren entwickelt, um einen hochwertigen Startpunkt für die nichtlinearen Zuordnungsprobleme zu erzeugen. Zusätzlich werden Suchalgorithmen für dessen diskrete Nachbarschaft untersucht und verbessert. Für das gemischt-ganzzahlige nichtlineare Optimierung Problem, es wird ein Nebenbedingungen¬-gerichteter Algorithmus entwickelt. Eine erzwungene Nachbar Suchstrategie wird auch dargestellt, um die Qualität der Lösung zu verbessern. Diese Verfahren zielen auf den Erhalt eine hochwertige Lösung mit deutlich reduzierten Anzahl von Funktionsauswertungen als Algorithmen zur Hand. Die Effizienz der entwickelten Methoden werden über drei praktische technische Probleme gezeigt. Dies sind die optimale Allokation von Komponenten, die Reihenfolgeoptimierung von Laminatschichte der faserverstärkte Kunststoffe sowie die gleichzeitige Materialauswahl und Geometrieoptimierung von Fachwerken und versteiften Platten.

Acknowledgements

I would like to express my gratitude to my supervisor Prof. Dr.-Ing. Horst Baier for his support regarding my study and his consideration and help with my living in Germany. I would like to acknowledge German Academic Exchange Service (DAAD) for awarding the DAAD Scholarship to support me to carry out study at Institute of Lightweight Structures, Technische Universität München. My thanks go to all colleagues at the Institute not only for their help but also for their friendship. Especially I would like to thank Mr. Erich Wehrle for academic discussions and Mr. Tanut Ungwattanapanit for his close collaboration in the research.

München, 01.01.2015

Yang Zhang

Contents

Abstract i

Kurzfassung ii

Acknowledgements iii

Contents iv

List of Figures ix

List of Tables ix

Nomenclature xv

1. Introduction 1

1.1 Motivation 1

1.2 Scope and objective of thesis 1

1.3 Outline of thesis 2

2. State of the art 4

2.1 Structural optimization containing integer and mixed-integer variables 4

2.2 Pure Integer assignment problem 5

2.2.1 Mathematical formulation 5

2.2.2 Linear assignment problem (LAP) 7

2.2.3 Nonlinear assignment problem (NAP) 7

2.2.4 State-of-the-art algorithms for NAP 8

2.2.4.1 Local search or neighbor search based algorithms 8

2.2.4.2 Genetic algorithms (GA) 9

2.2.4.3 Permutation discrete particle swarm optimization algorithm (PDPSO) 12

2.2.4.4 Other heuristic algorithms 15
2.2.4.5 Discussion on good starting points 15
2.2.5 Applications 16
2.3 Mixed-integer nonlinear programming (MINLP) 16
2.3.1 Mathematical formulation 16
2.3.2 State-of-the-art algorithms for MINLP 17
2.3.2.1 Branch and bound algorithm 18
2.3.2.2 Outer-approximation algorithm (OA) 20
2.3.2.3 OA based branch and cut algorithm 22
2.3.2.4 Trust region sequential quadratic programming algorithm (MISQP) 24
2.3.3 Applications 25
2.4 Criteria for evaluation and comparison of algorithms 26
3. An efficient procedure for solving nonlinear assignment problems 27
3.1 Linear assignment problem based starting point generation 27
3.2 Accelerated neighbor search algorithm 28
3.3 Computational complexity analysis 29
3.4 Test results on Quadratic Assignment Problem Library (QAPLIB) 29
3.4.1 Test suites and numerical results 30
3.4.2 Computational cost analysis 31
3.4.3 Effects of the developed acceleration technique through comparison with a traditional neighbor-search algorithm 32
4. An efficient algorithm for MINLP with binary assignment 34
4.1 MINLP with binary assignment 34
4.2 Constraints-directed binary assignment algorithm 34

4.3 Computational complexity analysis 37

4.4 Benchmark mathematical examples and comparison of several different algorithms 37

5. Optimal parts allocation for structural systems 40

5.1 Introduction to optimal parts allocation problem 40

5.2 Mathematical modeling into an assignment problem 41

5.3 Examples and numerical results of parts allocation w.r.t. coefficient of thermal expansions 42

5.4 Examples of parts allocation w.r.t. CTE, E and area of cross-section and comparison of different algorithms 46

5.4.1 A 10-bar truss structure (a small scale single group example) 46

5.4.2 A 25-bar truss structure with stress constraints (a medium scale multiple groups example) 48

5.4.3 A 72-bar truss structure (a large scale multiple groups example) 51

6. Stacking sequence optimization (SSO) in fibre-reinforced laminate structure design 54

6.1 Introduction to stacking sequence optimization 54

6.2 Theory of fibre-reinforced laminate analysis 55

6.2.1 Material properties matrices (A B D matrix) 55

6.2.2 Strain failure load factor 58

6.2.3 Tsai-Wu failure criterion load factor 58

6.2.4 Critical buckling load analysis 59

6.3 Mathematical formulation of stacking sequence optimization into a nonlinear assignment problem 61

6.4 Improvements on the optimization procedure 63

6.4.1 Linear assignment based starting point generation 63

6.4.1.1 Method with interpolation of material stiffness matrices 63

6.4.1.2 Method with quasi-homogenization concept..... 64

6.4.2 Repair operators for contiguity constraints..... 66

6.4.2.1 Introduction to Baldwinian repair operator 66

6.4.2.2 Maximum improvement repair operator 66

6.4.3 Optimization procedure..... 69

6.5 Demonstration through test problems and comparison of different algorithms..... 71

6.5.1 A thin panel design examples 71

6.5.1.1 Optimization without repair operators on the stacking sequence..... 72

6.5.1.2 Optimization with repair operators on the stacking sequence 74

6.5.2 Problems with large number of layers: thick composite panels design 75

6.6 Application in engineering problems 77

6.6.1 Composite panel with stiffeners..... 77

6.6.2 Hybrid-stiffness laminated fuselage panels with window cutouts 83

6.6.2.1 Introduction to hybrid-stiffness laminated fuselage panels with window cutouts..... 83

6.6.2.2 Stacking sequence optimization with respect to compression buckling 87

6.6.2.3 Stacking sequence optimization with respect to shear buckling 89

7. Simultaneous material selection and size optimization in structural design 91

7.1 Mathematical formulation into MINLP 91

7.2 Illustration with a single beam optimization problem 92

7.3 Demonstration with bar truss structures optimization through comparison of algorithms at hand..... 94

7.3.1 A small scale 3-bar truss example 94
7.3.2 A medium scale 25-bar truss examples 96
7.3.3 A large scale 72-bar truss example 99
7.4 Enforced neighbor-search strategy 102
7.4.1 Procedure with enforced neighbor-search strategy 102
7.4.2 Demonstration with the 72-bar truss example 103
7.5 Application in the design of a panel with stiffeners 104
7.5.1 Model description 104
7.5.2 Optimization problem and its solution 107
8. Conclusion and outlook 109
Bibliography 111
Appendix A. Full test results of QAPLIB problems 117
Appendix B. Optimal stacking sequence 125
B.1. Thin composite panel examples 125
B.2. Very thick composite panel examples 125
B.3. Composite panel with stiffeners 127

List of Figures

Figure 2-1: Flowchart of traditional 2-exchange neighbour search algorithm.....9

Figure 2-2: Flowchart of GA 10

Figure 2-3: An example to illustrate the work flow of partially mapped crossover 12

Figure 2-4: Flowchart of Particle swarm optimization algorithm.....13

Figure 2-5: An example to illustrate the definition of addition between position and velocity.....15

Figure 2-6: An example of branch in the Branch and Bound algorithm.....18

Figure 2-7: Flowchart of Branch and bound algorithm 20

Figure 2-8: Flowchart of Outer-approximation algorithm.....22

Figure 2-9: OA based Branch and Cut algorithm 23

Figure 2-10: Flowchart of MISQP algorithm 24

Figure 3-1: Procedure of initial solution generation 27

Figure 3-2: Accelerated neighbour search algorithm 28

Figure 3-3: Number of iteration V.S. size of problem.....31

Figure 3-4: Linear regression of number of iteration with size of problem 33

Figure 4-1: Procedure to generate initial solution for the problem.....36

Figure 5-1: A 10-bar truss structure.....40

Figure 5-2: 10-bar truss under thermal load (L=1000mm, E=68.95Gpa, A=1000mm^2) 44

Figure 5-3: 10-bar truss under both mechanical forces and a uniform thermal load.46

Figure 5-4: 25-bar truss structure.....49

Figure 5-5: 72-bar truss structure.....52

Figure 6-1: An individual fibre-reinforced ply (left) and a 8-layer fibre-reinforced laminate stacking with plies in four different orientations (right).....54

Figure 6-2: Distance of the layer to the midplane.....56

Figure 6-3: Composite laminated plate under in-plane loads.....60

Figure 6-4: Quasi-homogenization of a composite laminate material.....65

Figure 6-5: Modified quasi-homogenized material to evaluate c_{ij}.....65

Figure 6-6: Flowchart of maximum improvement repair operator.....67

Figure 6-7: Initial generation procedure with repair operator embedded.....69

Figure 6-8: Deterministic algorithm with repair operator embedded.....70

Figure 6-9: Layout of stacking sequence.....71

Figure 6-10: Composite panel with stiffeners.....78

Figure 6-11: Geometry of the cross-section.....78

Figure 6-12: Boundary conditions and normal loads.....79

Figure 6-13: First-order buckling mode of the structure with optimized stacking sequence.....79

Figure 6-14: Optimization history.....80

Figure 6-15: Distribution of critical load factors for buckling case.....81

Figure 6-16: Optimization history.....82

Figure 6-17: A fuselage panel with window cutouts to be reinforced.....84

Figure 6-18: Composition of layers for outer flange.....85

Figure 6-19: Linearly varied fiber layer.....85

Figure 6-20: Composition of plate layers.....86

Figure 6-21: Compressive load and buckling mode.....87

Figure 6-22: Distribution of load factor.....88

Figure 6-23: Histogram of load factors when the stacking sequence of only one component changes 89

Figure 6-24: Shear load and buckling mode 90

Figure 7-1: Cantilever beam under axial force 93

Figure 7-2: Three bar truss structure and loads 94

Figure 7-3: 72-bar truss structure 100

Figure 7-4: Single layer of the 72-bar structure with group illustration 100

Figure 7-5: 72-bar truss structure and loads 101

Figure 7-6: Optimization procedure with enforced neighbour-search strategy 102

Figure 7-7: Optimization structure 104

Figure 7-8: Panel structure with stiffeners 105

Figure 7-9: Cross-section of the stiffeners 105

Figure 7-10: Load case 1: buckling under compression 106

Figure 7-11: Load case 2: buckling under compression and shear forces 106

Figure 7-12: Load case 3: static analysis under extension and shear forces 107

List of Tables

Table 2-1: An example to illustrate the GR 11

Table 3-1: Results of the largest 11 problems in QAPLIB 30

Table 3-2: Problems that are solved with different error levels 30

Table 3-3: Results of the largest 11 problems in QAPLIB 32

Table 4-1: Results of the demonstration example at step 2 38

Table 4-2: Results of step 2 in new iteration 38

Table 4-3: Results of different algorithms 39

Table 5-1: CTEs of ten bars available 45

Table 5-2: Optimal allocation of bars 45

Table 5-3: Result with 5% properties deviation of 10-bar truss example 47

Table 5-4: Result with 10% properties deviation of 10-bar truss example 47

Table 5-5: Result with 50% properties deviation of 10-bar truss example 47

Table 5-6: Influence of deviations on the 10-bar truss structure 48

Table 5-7: Composition of groups 49

Table 5-8: Load cases for 25-bar structure 50

Table 5-9: Average results of 25-bar truss structure 51

Table 5-10: Load cases for 72-bar structure 52

Table 5-11: Average results of 72-bar truss structure without stress constraints 52

Table 5-12: Average results of 72-bar truss structure with stress constraints 53

Table 6-1: Empirical value of coefficient β regarding shear buckling load factor 60

Table 6-2: Example of evaluation of degree of improvement ($\mathrm{vio}_{\mathrm{original}} = 2$, N=7) 68

Table 6-3: Material properties 71

Table 6-4: Test cases 71

Table 6-5: Test results of different deterministic procedure 72

Table 6-6: Comparison of optimum and number of function evaluations with heuristic algorithms 73

Table 6-7: Results with Baldwinian repair operator 74

Table 6-8: Results with repair considering maximum degree of improvement 74

Table 6-9: Test cases of a very thick composite panel 75

Table 6-10: Test cases with different repair operators 76

Table 6-11: Material properties 77

Table 6-12: Composition of layers 78

Table 6-13: Critical load factors results 80

Table 6-14: Relevant strain and stress limits 81

Table 6-15: Critical load factors results 82

Table 6-16: Stacking sequence optimization result with compression buckling 88

Table 6-17: Optimum stacking sequence 88

Table 6-18: Stacking sequence optimization result with shear buckling 90

Table 6-19: Optimum stacking sequence 90

Table 7-1: Available material and related properties for the beam example 93

Table 7-2: Load cases of three bar truss example 94

Table 7-3: Available material and related properties for three-bar truss example 95

Table 7-4: Results of different algorithms for three bar truss problem 95

Table 7-5: Loading conditions for 25-bar truss example 96

Table 7-6: Global optimal of test 1 97

Table 7-7: Optimal found by developed algorithm of test 1 97

Table 7-8: Results of different algorithms for test 1 of 25-bar truss problem 97

Table 7-9: Global optimum of test 2 98

Table 7-10: Optimum found by developed algorithm of test 2 98

Table 7-11: Results of different algorithms for test 2 of 25-bar truss problem 98

Table 7-12: Available material and related properties for 72-bar truss 100

Table 7-13: Loading conditions for 72-bar truss example 101

Table 7-14: Optimal found by developed algorithm 103

Table 7-15: Material properties of four kinds of aluminum alloys 107

Table 7-16: Optimal objective value (Cost in Euro) of different material combinations 108

Table A-1: Test results of developed procedure on QAPLIB 117

Table A-2: Test results of procedure without acceleration technique on QAPLIB 120

Table B-1: Optimal stacking sequence for the thin composite panel 125

Table B-2: Optimal stacking sequence for the very thick composite panel 125

Table B-3: Optimal stacking sequence considering only buckling 127

Table B-4: Optimal stacking sequence considering buckling, strain and stress 127

Nomenclature

E	Young's modulus
A	Area of cross section
AP	Assignment problem
MINLP	Mixed integer nonlinear programming
MILP	Mixed integer linear programming
LAP	Linear assignment problem
NAP	Nonlinear assignment problem
QAP	Quadratic assignment problem
3-AP	3-dimensional assignment problem
GA	Genetic algorithm
GR	Gene-Rank Crossover
PMX	Partially mapped crossover
PSO	Particle swarm optimization algorithm
PDPSO	Permutation discrete particle swarm optimization algorithm
OA	Outer-approximation algorithm
B&B	Branch and bound algorithm
B&C	Branch and cut algorithm
MISQP	Mixed integer sequential quadratic programming algorithm
QAPLIB	A library of the quadratic assignment problem
NS	Traditional neighbor search algorithm
ANS	Accelerated neighbor search algorithm
K	Stiffness matrix in finite element analysis
U	Displacement vector in finite element analysis

F	Force vector in finite element analysis
α or CTE	Coefficient of thermal expansion
ΔT	Change of temperature
$\mathbf{F}_M$	Mechanical force load
$\mathbf{F}_T$	Thermal load
σ_{max}	Maximum stress in a structure
$\sigma_{allowable}$	Allowable stress
E_1	In-plane Young's modulus in longitudinal directions
E_2	In-plane Young's modulus in transverse directions
E_3	Young's modulus in vertical directions
ν_1	Poisson's ratio between longitudinal and vertical direction
ν_2	Poisson's ratio between transverse and vertical direction
Q, U, V	Material invariants in laminate analysis
θ	Layer fibre direction
A, **B**, **D**	Material stiffness matrices in laminate analysis
N	Uniformly distributed load
λ_{nc}	Critical buckling load factor under only normal loads
λ_{sc}	Critical buckling load factor under only shear loads
λ_n	Normal critical buckling load factor under both normal and shear loads
λ_c	Critical buckling load factor
ε_L	Strains in the longitudinal direction
ε_T	Strains in the transverse direction
ε_{LT}	Shear strains between longitudinal and transverse direction
ε_{LU}	Allowable value of longitudinal strain
ε_{TU}	Allowable value of transverse strain

ε_{LTU}	Allowable shear strain
σ_{Lt}	Longitudinal tensile strength
σ_{Lc}	Longitudinal compressive strength
σ_{Tt}	Transverse tensile strength
σ_{Tc}	Transverse compressive strength
τ_s	Shear strength
λ_σ	Maximum load factor regarding Tsai-Wu criterion
SSO	Stacking sequence optimization
G_{12}	In-plane shear modulus
v_{12}	In-plane Poisson's ratio
λ_ε	Maximum load factor regarding strain limits
ρ	Density
σ_A	Maximum allowable stress
σ_{cr}	Critical buckling stress

1. Introduction

1.1 Motivation

The employment of integer variables provides a more straightforward way of describing many structural optimization problems in mechanical engineering world. It simplifies the efforts on both mathematical modelling of practical problems and explanation of optimization results. These problems can be formulated into the categories of either pure integer or mixed-integer optimizations in mathematics, which are widely studied theoretically in recent decades.

Comparing to traditional optimization problems with continuous design variables, the algorithms for integer and mixed-integer optimization usually require much more computational efforts for a same problem size, especially in terms of number of function calls. This deficiency is magnified significantly and sometimes even hinders the application in structural optimization, due to the fact that structural performance evaluations are to be carried out by time-consuming finite element analysis. Therefore, efficient algorithms and procedures designed for pure integer and mixed-integer problems that often occur in the structural optimization are highly desired.

1.2 Scope and objective of thesis

The objective of this thesis is the development of procedures to efficiently solve integer and mixed-integer type optimization problems and the investigation of their applications in structural optimization. Specifically, solution methods for pure integer nonlinear assignment problem and mixed-integer nonlinear programming problems are of the interest of this thesis. With the developed tools, the solutions of three optimization problems in mechanical engineering world, namely optimal parts allocation problem, stacking sequence optimization problem in laminate design and simultaneous material selection and size optimization problem, are discussed.

The importance and the generation of a high quality staring point as the trigger of an optimization procedure are studied. Optimization algorithms to solve optimization problems are also investigated and improved. Their efficiencies, in terms of number of function evaluations, are demonstrated through comparisons with other algorithms with both mathematical and engineering problems. The presented procedures are also applied on large-scale problems in practice.

1.3 Outline of thesis

In Chapter 2, varieties of optimization problems with integer type design variables in engineering field are introduced. Literature surveys for the assignment problem (AP), especially the nonlinear assignment problem (NAP), mixed integer nonlinear programming problem (MINLP) are presented. Both the mathematical formulation for each problem and state-of-the-art deterministic and heuristic algorithms are summarised.

To solve nonlinear assignment problems efficiently, accelerated neighbour search algorithm and quasi-homogenous linear assignment procedure for generation of a high quality initial solution are developed in Chapter 3. The effectiveness of the algorithms and the whole procedure are demonstrated by benchmark test suites.

In Chapter 4, a constraints-directed binary assignment algorithm is developed for one type of MINLP problem. The performance of the algorithm is compared with several algorithms with small to medium scale problems.

The developed algorithms are applied to three structural optimization problems in mechanical engineering. The optimal parts allocation problem for structure systems and its formulation into a NAP are presented in Chapter 5. The optimization procedure developed in Chapter 3 is applied to solve this problem. Comparison of different solution strategies is also discussed.

Chapter 6 focuses on stacking sequence optimization in fibre-reinforced laminate structure design. The basic theories of composite laminate analysis are introduced at first. Then the formulation of the problem into a constrained nonlinear assignment problem is introduced. The optimization method developed in Chapter 3 is applied to this problem, and a new repair operator is integrated to accelerate the procedure. The reduction of computational cost is

presented by comparison with several heuristic algorithms on small to medium-scale problems. Finally, the procedure is applied on a large scale problem and a new concept hybrid-stiffness laminate design problem.

Chapter 7 concentrates on the simultaneous material selection and size optimization problem in structural lightweight design. The problem is formulated into a MINLP with binary assignment. The algorithm developed in Chapter 4 is applied directly. The performance is compared with several algorithms with small to large scale problems. An application in panel structures design is also presented.

The thesis is finally summarized in Chapter 8 together with an outlook on possible further research work.

2. State of the art

In this chapter, structural optimization problems that contain integer type design variables are introduced. General mathematical formulations of the nonlinear assignment problem and mixed-integer nonlinear optimization problem that are mainly investigated in this thesis are introduced. Reviews of state-of-the-art algorithms are presented.

2.1 Structural optimization containing integer and mixed-integer variables

In a structural optimization formulation, design variables denote parameters that describe the design of a structure. The optimization problem is defined to find the optimal values of design variables so that the represented structure achieves the best performance i.e. the maximal or the minimal objective value under certain constraints.

The constraints of an optimization problem are composed of two forms. One common form is the constraint functions, which are limitations that are applied on the system responses. The other form is constraints on the design variables. These constraints refer to not only ranges of design variables, but can also refer to the types of them. In a most general optimization formulation, types of design variables can be continuous, integer or mixed continuous and integer. The integer variables can be subdivided to pure integers, which take values of normal integers, binary integers, which take values of only zero or one, and quasi-integers, which take discrete sets of decimal values.

The integer type variables provide convenient and straightforward descriptions of practical problems, and therefore are at least equally important as the continuous ones if not more. Discrete structural optimization is frequently the case in practical engineering designs, where the structure parameters are limited to choose from sets of several specific values [Gutkowski 1994]. Typical examples are optimization thicknesses of material sheets [Li 2000] and optimization the discrete plies angles of laminate composites [Gürdal 1999]. Binary encoding is widely adopted in topology optimization [Bendsoe 2003], where element

densities are either zero or one. The binary description is also employed to represent the selections of materials in the discrete material optimization [Stegmann 2005]. Normal integers are utilized to describe the quantities of physical parameters, such as the number of layers in laminate composite [Apalak 2011] or number of reinforcing elements profile of metal matrix composites [Huber 2010]. In addition, the ordered integers provide natural representations of sequences and therefore are also utilized in the stacking sequence optimization of laminate composites [Riche 1993] and in the assembling procedure optimization [Graves 1983].

Although the advantages of employing integer variables are significant, the computational difficulties increase greatly after the integer variables are brought in. Therefore, many branches of mathematical optimization have emerged focusing on the solutions of specific types of integer optimization problems. Discrete optimization [Syslo 2006], integer programming [Li 2006], combinatorial optimizations [Papadimitriou 1998] and assignment problems [Pardalos 2000] are subsets that mainly investigate pure integer type optimization. Mixed integer nonlinear programming is another interesting topic [Floudas 1995, Lee 2012] investigating on solution algorithms for optimizations with both integer and continuous type design variables.

In this dissertation, the efficient solution procedures for nonlinear assignment problem and the mixed-integer nonlinear programming problem are mainly investigated. These two types of problems have important applications in structural optimization. Three typical examples are optimal parts allocation problem for structural systems, stacking sequence optimization problem and simultaneous material selection and size optimization.

2.2 Pure Integer assignment problem

2.2.1 Mathematical formulation

Assignment problem is a type of the most important problems in the field of combinatorial optimization. The most abstract description of this problem is that: given two sets of the same size, say set **S** and **T** both contain N elements. Find a

bijective mapping g: **S**→**T**, such that the so-called cost function i.e. the objective function f(g(**S**)) reaches its minimum or maximum.

For clarity, a small example is given here. Let set **S**={worker 1, worker 2, worker 3} and **T**={job 1, job 2, job3} both contain three elements. A bijective mapping g from **S** to **T** is a plan how to assign the jobs to workers, and here we assume that each job can be assigned only to one worker and each worker can only take one job. For example, g could be assign job 1 to worker 1, assign job 3 to worker 2 and assign job 2 to worker 3. The objective function f is the time required to finish all the jobs under the assignment plan g. The assignment problem here is to find the best assignment plan g so that f is minimized.

For each such mapping function g, it can be bijective mapped to a N by N binary square assignment matrix $\mathbf{X} = (x_{ij}) \in \{0,1\}^{N\times N}$, where

$$\mathrm{x}_{ij} = \begin{cases} 1 & if\ g(s[i]) = t[j] \\ 0 & otherwise \end{cases} \tag{2.1}$$

Still takes the above assignment example, then the corresponding assignment matrix **X** would be:

$$\mathrm{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

Due to the bijective properties of g, the following two row-wise and column-wise assignment constraints must be satisfied:

$$\sum_{j=1}^{N} x_{ij} = 1\ (i = 1,2,\dots,N) \tag{2.3}$$

$$\sum_{i=1}^{N} x_{ij} = 1\ (j = 1,2,\dots,N) \tag{2.4}$$

Therefore the problem is that, to find a binary assignemnt matrix **X** that satisfied Eq.(2.1), Eq.(2.3) and Eq.(2.4) so that the objective function f(**X**(**S**)) reaches its minimum or maximum.

It's worth pointing out that, in a standard assignment problem formulation, there is no other constraints except the assignment constraints. Additional constraints are normally taken into consideration by the penalty method which adds a penalty term to the objective function if violations of the constraints occur.

2.2.2 Linear assignment problem (LAP)

The problem with a special type of the objective function, i.e. the linear assignment problem (LAP) has been completely solved mathematically. For this type of problem, the objective is expressed as:

$$f = \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} \cdot x_{ij} \tag{2.5}$$

where the cost coefficients $\{c_{ij}\}$ are given and f is a linear function of entries in **X**. The most famous algorithm to solve this problem is the Hungarian algorithm [Kuhn 1955] or called Munkres' assignment algorithm [Munkres, Burkard 2009]. The computational complexity of this algorithm is of a low order polynomial time $O(N^3)$ in the worst case.

2.2.3 Nonlinear assignment problem (NAP)

Generally, the objective function is nonlinear functions of the assignment matrix , which is a straightforward extension of LAP. However, to design an algorithm to solve this type of problems in polynomial time is very difficult.

Actually, it has been proved that the quadratic assignment problem (QAP) where the objective is a quadratic function of entries in **X** as shown in the following:

$$f = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{l=1}^{N} a_{ij} \cdot b_{kl} \cdot x_{ik} \cdot x_{jl} \tag{2.6}$$

and the 3-dimensional assignment problem (3AP) where the objective is a quadratic function of entries of two assignment matrices:

$$f = \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} a_{ijk} \cdot x_{ij} \cdot \tilde{x}_{ik} \tag{2.7}$$

are both NP-hard problems [Sahni 1976, Frieze 1983]. It means that until now no algorithms could solve these problems or even guarantee of finding a good enough solution (with some given factor from the optimal solution) in polynomial time.

2.2.4 State-of-the-art algorithms for NAP

Although there is no proof for other types of nonlinear objective function, it is logic to think that for even more complicated nonlinear objective functions, the problems are also NP-hard problems. Since an algorithm to efficiently find the global optimum are not possible, many deterministic and heuristic algorithms have been presented to find good enough solutions for general NAPs.

2.2.4.1 Local search or neighbor search based algorithms

Local search or called neighbour search based algorithm is widely applied in combinatorial optimization problems [Aarts2003]. The flowchart of the traditional 2-exchange neighbour search algorithm for assignment problems is illustrated in the following figure.

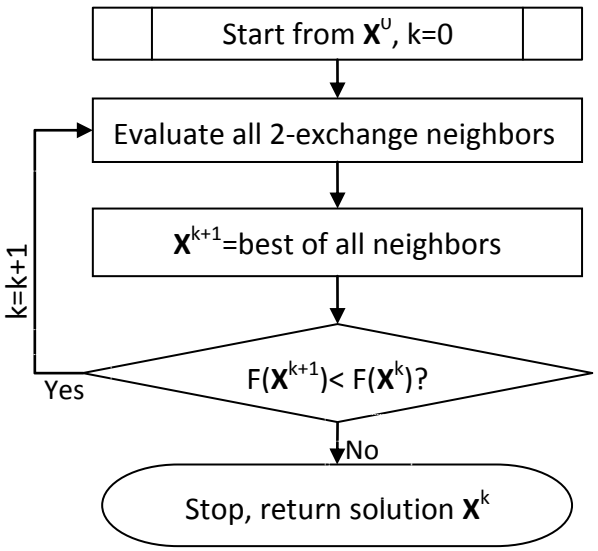


Figure 2-1: Flowchart of traditional 2-exchange neighbour search algorithm

A 2-exchange neighbour of a current solution **X** is constructed through switching exactly two rows or two columns of the design matrix. The advantage of this algorithm is that it ensures the final solution is better than all of its neighbours, which is at least a local optimum.

The main disadvantages of this algorithm lie in two points: firstly, the number of function calls of neighbourhood evaluations is quite large which is approximately N^2; secondly, the improvement after each iteration is limited only to assignments of two elements. On one hand, this can cause that a lot of iterations is required until the stop criteria is met, on the other hand, it means that the algorithm may easily got stuck in a local optimal due to its restricted search area in each iteration.

2.2.4.2 Genetic algorithms (GA)

Genetic algorithms (GA) [Goldberg1989] have also been presented for the solution of assignment problems. The flowchart of a general genetic algorithm is depicted in Figure 2-2. In each iteration, new individuals, the so-called children, are reproduced from the previous population, the so-called parents, through mutations and crossovers on the parents' genes. The fitness evaluation refers to the function evaluations of individuals. The most commonly utilized selection strategies include Tournament Selection, Roulette Wheel Selction and Uniform Selection, with which a new generation of population will be chosen from the individuals.

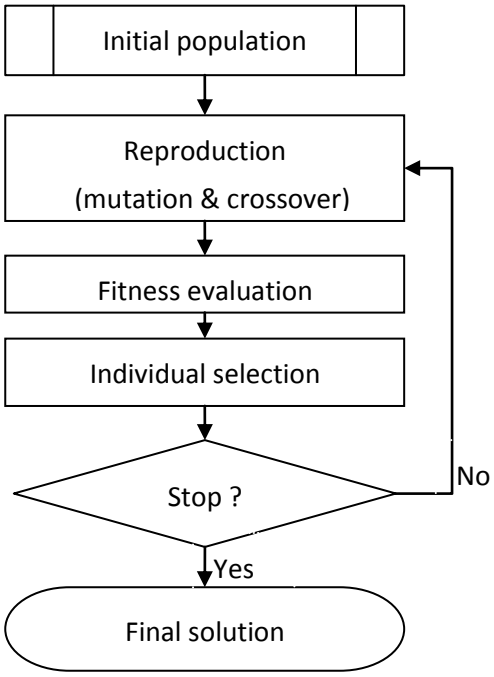


Figure 2-2: Flowchart of GA

To apply the GA to assignment problems, one assignment of size N is usually represented by a vector of integers from 1 to N (called chromosome) following the relation:

$$\mathbf{Y} = \mathbf{X} \cdot [1,2, \dots, \mathrm{N}]^{\mathrm{T}} \tag{2.8}$$

Generally, mutation operators and crossover operators are two ways of generating new children from parents. Mutation operators randomly change several genes of one chromosome, and crossover operators interchange the genes between two or more chromosomes. In assignment problems, both of these two operators are slightly different due to that the generated new chromosomes must still satisfy the assignment constraints. To ensure this limitation is respected, a mutation of the chromosome means swaps randomly selected two genes in one chromosome.

There are two effective crossover operators that have been presented for assignment problems: the Gene-Rank Crossover (GR) and the partially mapped crossover (PMX) [Liu 2000]. The gene-rank crossover firstly calculates the weighted sum of each gene of the chromosomes that are involved in the crossover. Then sort the values in ascending order and assign an integer to each

gene according to their sequences. An example of applying the GRX on two chromosomes of size 5 is presented in the following table.

Table 2-1: An example to illustrate the GR

Gene	Chromosome1	Chromosome2	Weighted sum	Children (=Ranking)
1	5	1	2.6	3
2	1	3	2.2	1
3	4	4	4	5
4	2	5	3.8	4
5	3	2	2.4	2
Weighting factor	0.4	0.6	----	----

The partially mapped crossover defines four steps to create a child from two parents by crossover. Firstly, randomly select two breaking position of both chromosomes of parents. Secondly, the child inherits the middle genes between two breaking points of the second parent. Thirdly, the child inherits the genes before the first breaking point and after the second breaking point from the first parent if no confliction with already inherited genes from the second parent. Lastly, if there are conflictions, defines mapping relations of conflicted genes and fill them in the corresponding position of the child utilizing this mapping.

An example of applying the PMX on two chromosomes of size 5 is here described. The two chromosomes are [1,2,3,4,5] and [5,4,3,2,1]. Firstly, one breaking point is randomly selected between gene 1 and gene 2 and the other is between gene 3 and gene 4. Secondly, the child inherit the middle part of chromosome 2 i.e. [*,4,3,*,*]. Thirdly, since the first gene of chromosome 1 is “1” which has no confliction with “4” and “3” that inherited from chromosome 2, it fills in the first gene of the child, i.e.[1,4,3,*,*]. In the same way, “5” is filled also in the child, i.e. [1,4,3,*,5]. Because “4” is already appears in the child, it has a confliction and therefore can’t be inherited to the child. Lastly, a mapping is defined for this confliction: “4” is at the second gene of the chromosome 2. The second gene of chromosome 1 is “2” which has no confliction of all already fixed genes of the child, therefore it is filled into the child, and the child generated by this crossover operator finally is [1,4,3,2,5]. The following flowchart depicts the entire procedure.

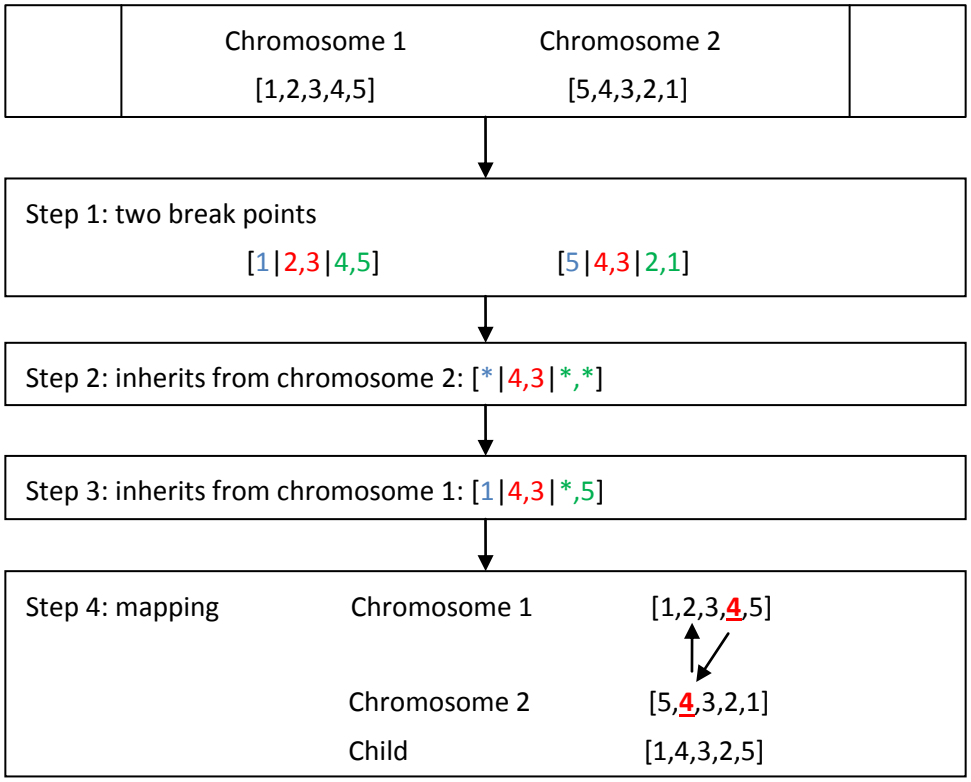


Figure 2-3: An example to illustrate the work flow of partially mapped crossover

2.2.4.3 Permutation discrete particle swarm optimization algorithm (PDPSO)

Particle swarm optimization algorithm imitates the swarm behaviour of birds or fish in searching for food. Permutation discrete particle swarm optimization (PDPSO) [Salman 2002] is a derivative of PSO, which has been modified to suit the nature of assignment problems. And it achieves success in the stacking sequence optimization problems that will be presented later.

The following figure depicts the flow chart of a general PSO algorithm. Firstly, at starting positions of individuals among a swarm, the velocity of each individual (i.e. the direction of flying or swimming in the next step) is calculated. Then the positions of them are updated respectively. Finally, the fitness of each new position (i.e. the distance from food) is evaluated. The iterative procedure stops until one of the stopping criteria is met.

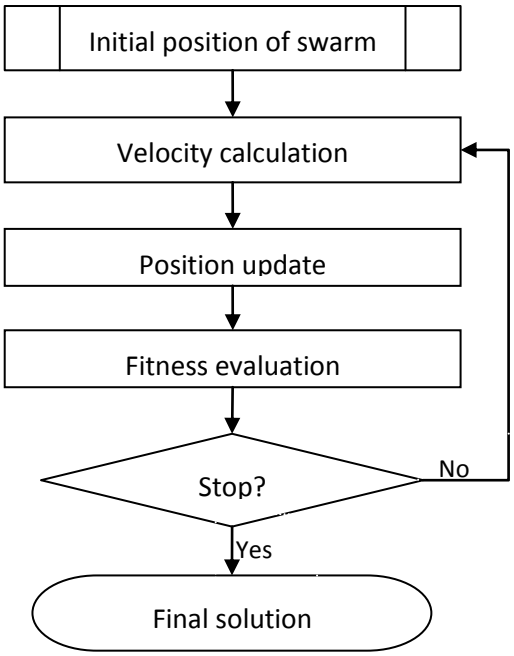


Figure 2-4: Flowchart of Particle swarm optimization algorithm

The velocity vector of individual i in the iteration k can be defined as:

$$\mathbf{V}_{\mathrm{k}}^{\mathrm{i}} = \omega \mathbf{V}_{\mathrm{k-1}}^{\mathrm{i}} + \mathrm{c}_1(\mathbf{X}_{\mathrm{best}}^{\mathrm{i}} - \mathbf{X}_{\mathrm{k-1}}^{\mathrm{i}}) + \mathrm{c}_2(\mathbf{X}_{\mathrm{k-1}}^{\mathrm{best}} - \mathbf{X}_{\mathrm{k-1}}^{\mathrm{i}}) \quad (2.2)$$

Where $\omega, \mathrm{c}_1, \mathrm{c}_2$ are acceleration factors, $\mathbf{V}_{\mathrm{k-1}}^{\mathrm{i}}$ is the velocity of individual i in the k-1 iteration, $\mathbf{X}_{\mathrm{best}}^{\mathrm{i}}$ is the best position vector that individual i has ever reached in the previous iterations, and $\mathbf{X}_{\mathrm{k-1}}^{\mathrm{best}}$ is the best position vector of all the individuals in the last iteration.

The positions update straightforwardly following the relation:

$$\mathbf{X}_{\mathrm{k+1}}^{\mathrm{i}} = \mathbf{V}_{\mathrm{k}}^{\mathrm{i}} + \mathbf{X}_{\mathrm{k+1}}^{\mathrm{i}} \quad (2.3)$$

To apply in the assignment problems, there are several limitations similar as those met in GA. Firstly, the entries of the velocity vector must be all integers. Secondly, the addition operator must be properly defined so that the updated position vectors satisfy the assignment constraints. For this purpose, the subtract

operator, the scalar product operator and the addition operator are redefined in PDPSO [Chang 2010]. The subtract operator is defined as:

$$\mathbf{S} = \mathbf{X}^1 - \mathbf{X}^2$$
$$s_j = \begin{cases} 0 & \text{if } x_j^1 = x_j^2 \\ x_j^1 & \text{otherwise} \end{cases} \tag{2.4}$$

The scalar product operator is defined as:

$$\mathbf{P} = c \cdot \mathbf{V}$$
$$p_j = \begin{cases} 0 & \text{rand1}(j) \le c \\ v_j & \text{otherwise} \end{cases} \tag{2.5}$$

In which rand1(j) is a uniformly distributed random number for element j.

The addition operator for velocity vectors is defined as:

$$\mathbf{V} = \sum_{k=1}^{M} \mathbf{V}^k$$
$$v_j = v_j^m \quad \text{if } \frac{m-1}{M} < rand2(j) \le \frac{m}{M} \tag{2.6}$$

In which rand2(j) is another uniformly distributed random number in the range of (0,1) for element j.

The addition operator for velocity vector and the position vector is defined as:

$$\mathbf{X}^{new} = \mathbf{V} + \mathbf{X}^{old} \tag{2.7}$$

In which, $x_j^{new} = x_j^{old}$ if $v_j = 0$. And otherwise, search k such that $x_k^{old} = v_j$, and swap x_k^{old} and x_j^{old} to get $\mathbf{X}^{new}$. This operator is illustrated with an example in the following.

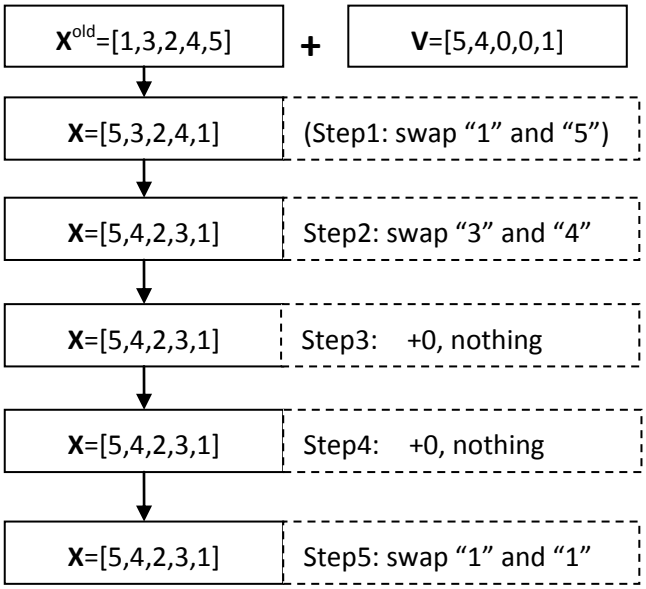


Figure 2-5: An example to illustrate the definition of addition between position and velocity

2.2.4.4 Other heuristic algorithms

Several heuristic algorithms, such as greedy randomized adaptive search (GRASP) [Feo1991], simulated annealing [Dowsland 1993, Kirkpatrick1983], ant systems [Tsutsui 2007] and Tabu search [Glover 1997] have been developed based on the traditional neighbour search strategy to improve the quality of its solution. They have been applied to practical problems such as travelling salesman problems.

One of the most important characteristics of these algorithms is that the price of the improvement of the solution quality always comes along with more number of function calls. As the size of the problem increases, the design space for assignment problems i.e. the total number of combinations increases in a factorial way. Therefore, it becomes much more difficult to generate representative high quality individuals heuristically. Hence they are shown to be limited in solving assignment problems with time-consuming function evaluations like finite element analysis.

2.2.4.5 Discussion on good starting points

The starting points or a starting point has significant influence on the efficiency of above mentioned algorithms. A higher quality trigger will reduce the number of

iterations, thus will significantly reduce overall computational costs. There are several ways to construct a starting point, for instance, by taking the identity permutation, a randomly generated permutation, or a heuristically determined starting point [Pardalos2000]. For the first two methods, they don't include any consideration of a specific problem, so there is no reason to take them as a good starting point. To heuristically determine a high quality starting point, additional information is needed. This information comes from either sophisticated engineering experience, which will not be discussed in this thesis, or delicately designed procedure for direction to a good point. The efficiency of these different strategies in selection of the starting point will be compared through examples in later chapters.

2.2.5 Applications

The assignment problems are most presented in the field of operation research. Travelling salesman, scheduling are typical applications that are widely studied. The application of the assignment problem in engineering is found in turbine fan balancing problems. The optimal parts allocation problem and the stacking sequence optimization of laminate material design which will be discussed in this thesis are several important application examples.

2.3 Mixed-integer nonlinear programming (MINLP)

2.3.1 Mathematical formulation

Different to the problems discussed in section 2.2, where all the design variables are integers, the mixed-integer optimization problems where design variables contain both integer and continuous types are introduced in this section. The most general formulation of this kind of problem is:

$$\begin{aligned} &\min f(\boldsymbol{x}, \boldsymbol{y}) \\ &\text{s.t.: } h_i(\boldsymbol{x}, \boldsymbol{y}) = 0 \quad i = 1,2,\ldots,\text{P} \\ &g_j(\boldsymbol{x}, \boldsymbol{y}) \leq 0 \quad \text{j=1,2,...,Q} \\ &\boldsymbol{x} \in \boldsymbol{X} \subseteq \boldsymbol{\mathcal{Z}}^N \\ &\boldsymbol{y} \in \boldsymbol{Y} \subseteq \boldsymbol{\mathcal{R}}^M \end{aligned} \tag{2.8}$$

The type of integer variables could be categorized as: binary integer, pure integer, pseudo-integer and categorial integer. Binary integers take the value of 0 or 1,

which are often used to represent answers for “yes” or “no” questions, like “Is the component is to be made out of steel?” Pure integers represent the number of physical quantities, like “how many layers does the composite material contain?” Pseudo-integers take the same as pure integers, but the value doesn’t have direct meaning and need to be physically explained according to real problems. For example, a pseudo-integer takes value of 1 to 3 may mean that the thickness of a panel sheet is 1.5mm, 2mm or 2.5 mm. Categorial integers are introduced to represent different situations. For example, the shape of the cross-section area of a tube could be represented by 1-circular, 2-square, 3-triangular. The change of the value leads to a completely different category and thus may lead to significantly different responses.

Integer variables could provide a straightforward way of mathematical modelling of practical optimization problems. However, on the other hand, the existence of integer variables makes the solution procedure much more complicated. Most of the gradient-based algorithms can’t be directly employed unless the gradients with respect to integer variables are properly defined. What’s more, optimality criteria for continuous optimization often fail at the optimum for mixed-integer problems. The lack of optimality criteria makes the design of optimization algorithms more difficult. Actually, it has been proved that the mixed-integer linear programming problem (MILP), where the objective function and the constraints in Eq.(2.14) are all linear functions with respect to both continuous and integer variables, is NP-complete problems [Vavasis 1991], which means that there is no polynomial time algorithms for this type of problem. It is even more complicated when the system responses are nonlinear.

2.3.2 State-of-the-art algorithms for MINLP

The scientific endeavour of developing algorithms for general MINLP problems have been and is being made. An excellent overview of the algorithms has been presented by Floudas [Floudas 1995, Bonami 2008]. In this section, four most important algorithms i.e. Branch and Bound algorithm [Gupta 1985], Outer-Approximation algorithm [Duran 1986], OA-based Branch and Cut algorithm [Quesada 1992] and MISQP algorithm [Schittkowski 2006] will be detailed introduced.

2.3.2.1 Branch and bound algorithm

The idea of branch and bound algorithm is to utilize the sophisticated optimization tools in continuous optimization to find the lower bound of the optimization problem. At the same time, find feasible solutions to reduce the upper bound of the problem. As the algorithm proceeds, the lower bound is increased and the upper bound is decreased. The procedure continues until the lower bound is equal to the upper bound and thus the corresponding feasible solution is the optimum of the problem.

In detail, the branch and bound algorithm first relax the integrality restrictions on integer variables to form a continuous-relaxed nonlinear programming problem (RNLP). The optimal solution to this problem forms a lower bound of the original problem.

And then, a branch regarding one of the integer variables is carried out. It breaks the RNLP further into two RNLPs with an additional constraint on the design domain of the selected integer variable. An example is illustrated in Figure 2-6. There are different strategies to select the branching variables. A basic rule is that the selected variable should take non-integer values in the continuous optimizer.

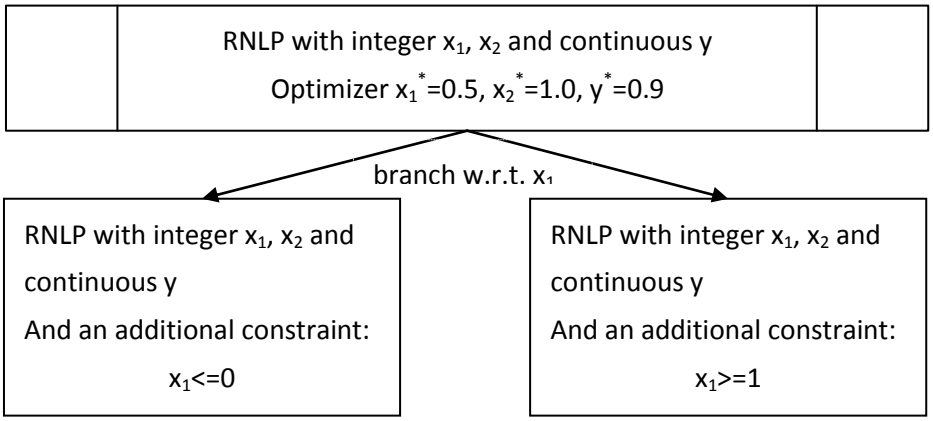


Figure 2-6: An example of branch in the Branch and Bound algorithm

By solving the two new RNLPs, better lower bound may be obtained. The iterative procedure will generate a binary tree structure from the continuous-relaxation of the original problem without any additional constraints. For each node in the tree, it is a RNLP with additional design domain constraints. As the depth of the nodes in the tree gets deeper, more and more additional constraints will be added.

A node will stop branch if any of the following three conditions is met, which is called pruning conditions or fathoming criteria:

(1) Pruning by integrality: the optimal solution of the RNLP is a feasible solution which satisfies all the constraints and integrality limitations on design variables. In this case, the objective value is a upper bound of the original problem.
(2) Pruning by bounds: The objective value of the optimal solution is larger than the upper bound of the original problem.
(3) Pruning by infeasibility: The RNLP is infeasible.

A flowchart of the general branch and bound algorithm is depicted in the following figure.

The computational cost for branch and bound algorithm is large from the two points of view: firstly, for each RNLP, it needs a full solution of the nonlinear programming problem of the same size as the original problem. Secondly, at each branching, two RNLP will be generated with respect to one integer variables. Therefore, the number of nodes, i.e. the number of RNLP to be solved, could grow exponentially as the depth of the branching tree grows. This is a significant disadvantage for a problem with large number of integer design variables.

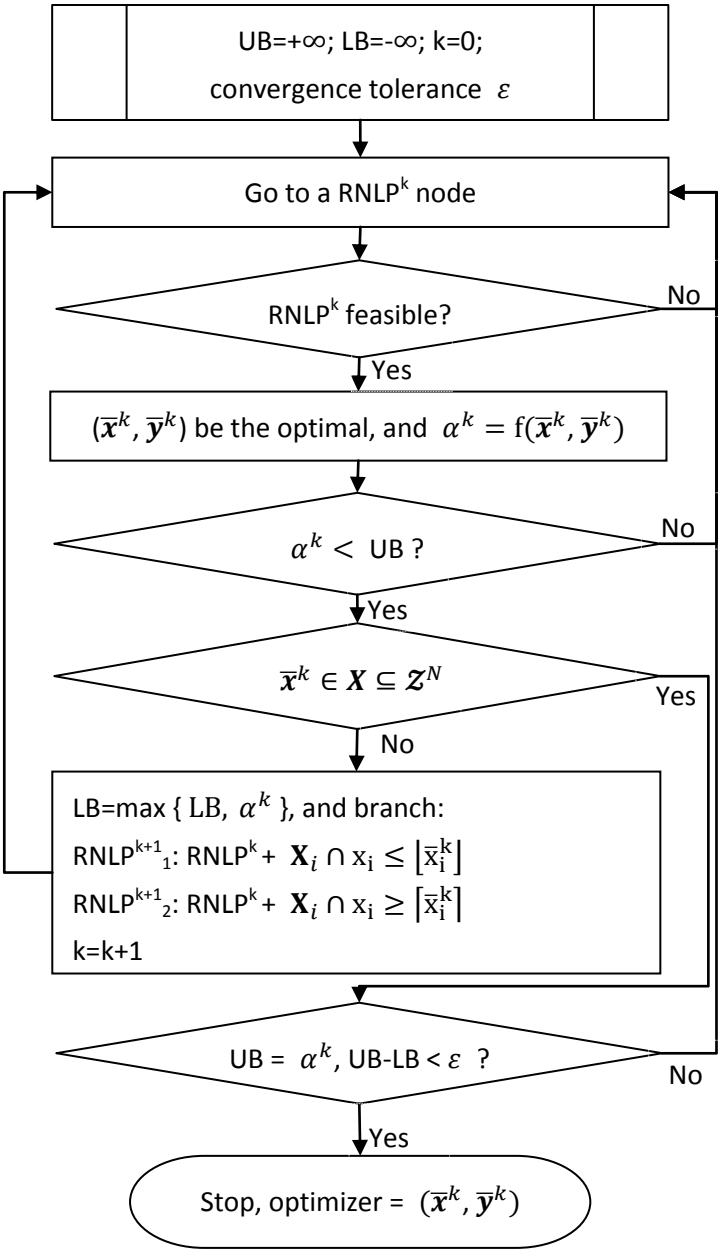


Figure 2-7: Flowchart of Branch and bound algorithm

2.3.2.2 Outer-approximation algorithm (OA)

Outer-approximation algorithm is a method based on linearization of the nonlinear objective and inequality constraints to assist the solution of MINLP problem through a series of mixed-integer linear programming problems [Fletcher 1994, Akrotirianakis 2001].

An outer-approximation of the MINLP problem stated in Eq.(2.16) without equality constraints at point $(\bar{x}, \bar{y})$ is:

$$
\begin{aligned}
&\min\ \alpha \\
&\text{s.t.: } \nabla \mathrm{f}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{x}-\overline{\boldsymbol{x}} \\ \boldsymbol{y}-\overline{\boldsymbol{y}} \end{pmatrix} + f(\overline{\boldsymbol{x}},\overline{\boldsymbol{y}}) \le \alpha \\
&\nabla \mathrm{g_j}^{\mathrm{T}} \begin{pmatrix} \boldsymbol{x}-\overline{\boldsymbol{x}} \\ \boldsymbol{y}-\overline{\boldsymbol{y}} \end{pmatrix} + \mathrm{g_j}(\overline{\boldsymbol{x}},\overline{\boldsymbol{y}}) \le 0,\ \mathrm{j=1,2,\ldots,Q} \\
&\boldsymbol{x} \in \boldsymbol{X} \subseteq \boldsymbol{\mathcal{Z}}^N \\
&\boldsymbol{y} \in \boldsymbol{Y} \subseteq \boldsymbol{\mathcal{R}}^M
\end{aligned}
\tag{2.9}
$$

The left sides of both equations are called linearization of the nonlinear problem. They are obtained through the first order Taylor expansion of multivariable functions, and are linear functions of design variables. Thus the above problem is a MILP problem and could be solved by branch and bound algorithm.

The solution of the outer-approximation problem is quite efficient from the number of system responses evaluations point of view. To establish the OA, gradients of the objective and constraints functions are needed to be evaluated, which requires (1+Q)(N+M) number of function calls when finite differencing method is employed, or 1+Q number of function calls if efficient semi-analytical techniques are employed. Once the OA is generated, the problem is expressed as a pure mathematical problem and there is no need for further system evaluation like finite element analysis. Therefore, this method is more appropriate to apply on structural optimization problems.

A flowchart of the outer-approximation algorithm is depicted in Figure 2-8. The algorithm first attempts to find the integer part from solving the outer-approximation MILP problem, whose solution is a lower bound the original problem. And then, by fixing the integer part, a NLP problem is solved to generate a feasible solution of the original problem as a upper bound. When the lower bound and upper bound come together, an optimum is found.

It should be noted that, the algorithm is named outer-approximation because if the objective and constraints are all convex function, then the feasible region of the linearized problem is an outer hull and thus contains the entire feasible domain of the original problem. Therefore, this algorithm is designed to solve convex optimization problems. For non-convex problems, which is always the case in structural optimization, the performance of the algorithm is not guaranteed.

The computational cost of this algorithm is dominated by the gradient evaluation to generate OA and the solution of NLP problem with only continuous design

variables. As the points set increases, the generated OA is closer to the original problem. And thus the optimal solution of the OA will also be close enough to the solution of the original problem. Therefore, the problem of this algorithm is that it often requires many iterations to establish a large enough point set, on which the OA will be generated.

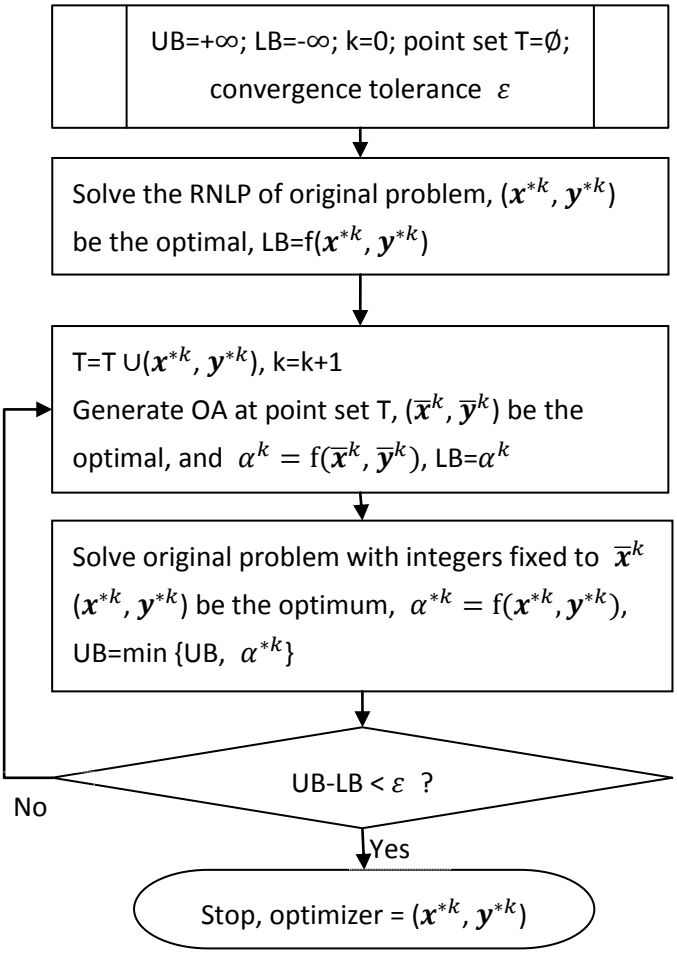


Figure 2-8: Flowchart of Outer-approximation algorithm

2.3.2.3 OA based branch and cut algorithm

As discussed earlier, the computational cost for branch and bound comes from the solution of NLP at a large number of nodes. The NLP problem could be mitigated by outer-approximation method which utilizes the linearization technique to transfer the nonlinear problems into successive linear problems.

If the feasible region of the relaxed problems could further be reduced, then there will be more nodes be pruned due to infeasibility, which means that

number of nodes to be solved will be reduced. Cutting plane method is one of such methods that have been investigated since 1958 [Gomory 1958, Westerlund 1995, Stubbs 1999]. It gradually introduces additional constraints called cuts on to the continuous relaxation problem to help exclude the integer solutions that lead to infeasibility.

Therefore, a combination of the B&B, OA and cutting planes would lead to a better way of solution. The flowchart of the OA based branch and cut algorithm is presented in the following [Quesada 1992].

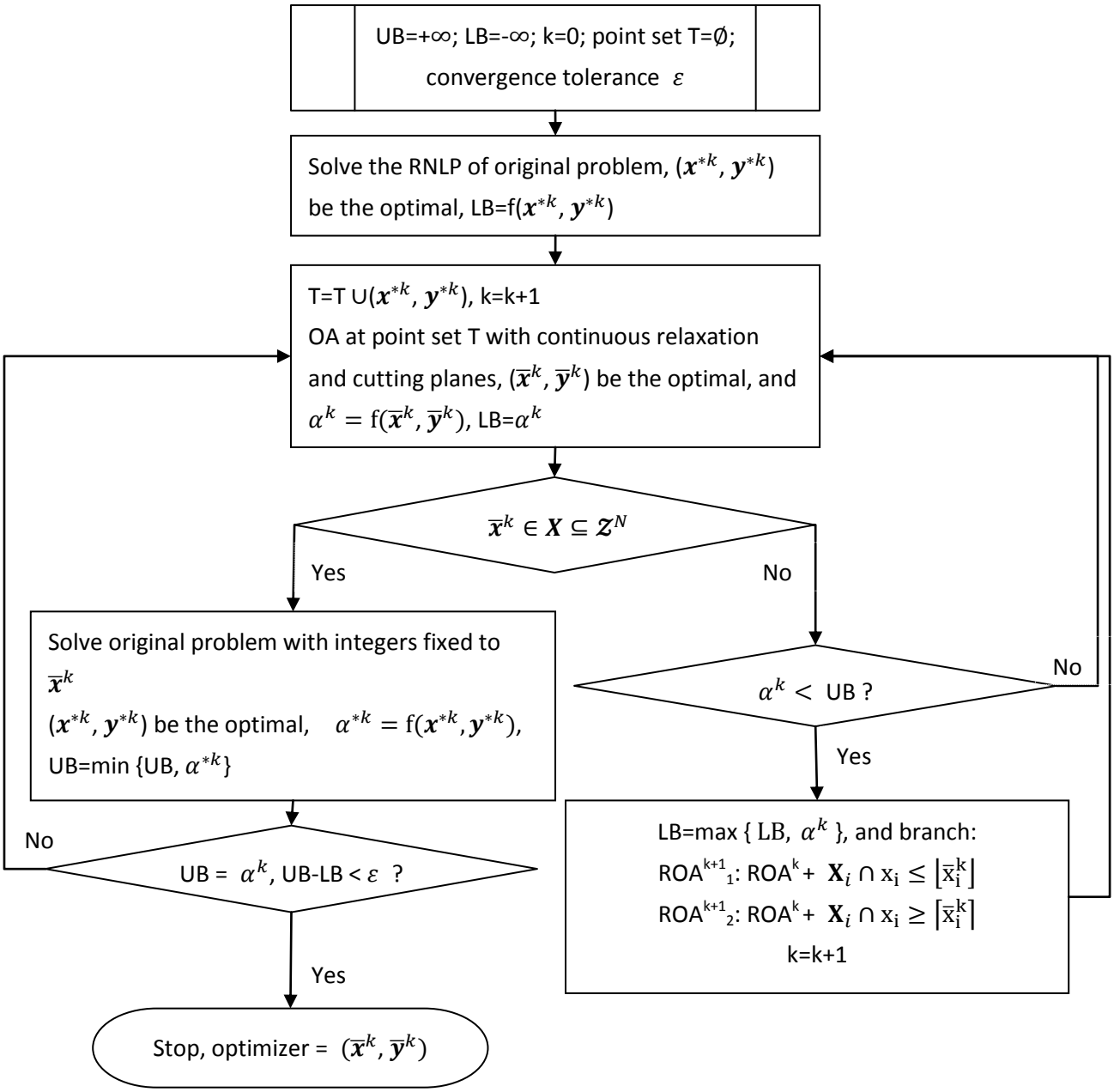


Figure 2-9: OA based Branch and Cut algorithm

It is worth to mention that, the relaxed outer-approximation problem is actually a linear programming problem. Therefore, it is fairly easy to solve.

2.3.2.4 Trust region sequential quadratic programming algorithm (MISQP)

Schittkowski has proposed a trust region sequential quadratic programming algorithm [Schittkowski 2006]. The algorithm iteratively establishes a quadratic optimization problem, to approximate the original general nonlinear problem. The objective function of the quadratic programming is a convex and quadratic function of the design variables, and is built up based on the Hessian matrix and gradient vector at the solution point of the previous iteration. The increment of the solution in each iteration is restricted within a trust region which centered at current solution to improve the stability of the algorithm and to enforce convergence. The Flowchart of the algorithm is depicted in the following.

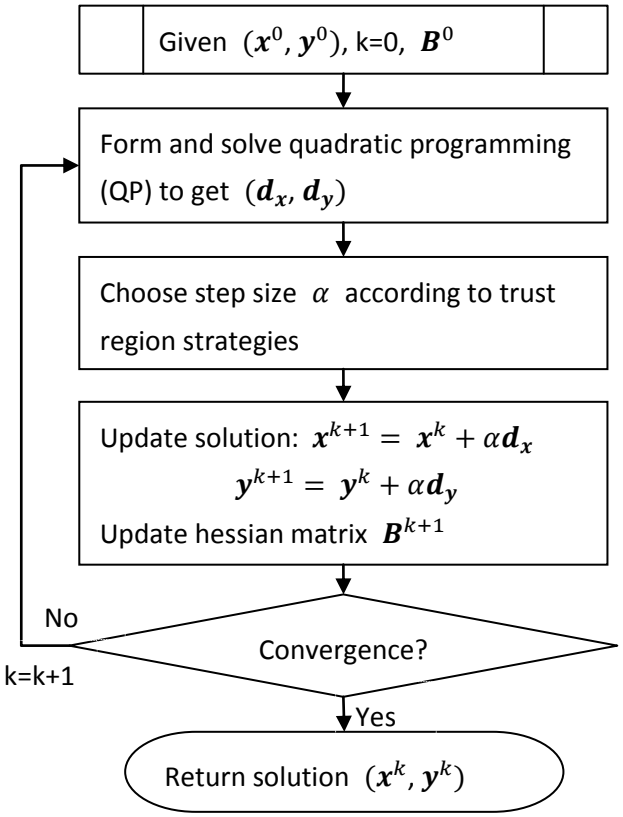


Figure 2-10: Flowchart of MISQP algorithm

The quadratic approximation of the original mixed integer problem is formulated as :

$$\begin{aligned} &\min \frac{1}{2}\begin{pmatrix}\boldsymbol{d_x}\\ \boldsymbol{d_y}\end{pmatrix}^{\mathrm{T}} \boldsymbol{B}^k \begin{pmatrix}\boldsymbol{d_x}\\ \boldsymbol{d_y}\end{pmatrix} + \nabla f(\boldsymbol{x}^k, \boldsymbol{y}^k)^{\mathrm{T}} \begin{pmatrix}\boldsymbol{d_x}\\ \boldsymbol{d_y}\end{pmatrix} + \sigma_{\mathrm{k}}\delta \\ &\text{s.t.:} \left|\nabla \mathrm{h_i}^{\mathrm{T}} \begin{pmatrix}\boldsymbol{d_x}\\ \boldsymbol{d_y}\end{pmatrix} + \mathrm{h_i}(\boldsymbol{x}^k, \boldsymbol{y}^k)\right| \le \delta, \text{ i=1,2,...,P} \\ &\nabla \mathrm{g_j}^{\mathrm{T}} \begin{pmatrix}\boldsymbol{d_x}\\ \boldsymbol{d_y}\end{pmatrix} + \mathrm{g_j}(\boldsymbol{x}^k, \boldsymbol{y}^k) \le \delta, \text{ j=1,2,...,Q} \\ &\max(\boldsymbol{x_l}, -\Delta_k^i) \le \boldsymbol{d_x} \le \min(\boldsymbol{x_u}, \Delta_k^i) \\ &\max(\boldsymbol{y_l}, -\Delta_k^c) \le \boldsymbol{d_y} \le \min(\boldsymbol{y_u}, \Delta_k^c) \\ &0 \le \delta \end{aligned} \tag{2.10}$$

, where Δ_k^i, Δ_k^c and δ are trust region parameters.

The gradients respect to integer variables are approximated with finite differencing at grid point. The inequality constraints with absolute value is derived from the equality constraint of the original problem, and the inequality constraints without absolute value come from original inequality constraints and the region parameters.

The algorithm requires no requirement on the relaxability of integer variables. Since it approximate the gradient with finite differencing, therefore, it is assumed that responses values do not change drastically when an integer value changes. The algorithm has been compared with several other MINLP solvers and shows a lot success in solving general MINLP benchmark test suites [Schittkowski 2013] and a lot reduction on the computational costs. But the applicability of this algorithm into mechanical engineering optimization, where the change of integer value may lead to significant change in system responses, is still need to be investigated.

2.3.3 Applications

Several applications of MINLP programming in structural engineering have been presented, including the discrete material optimization and free material optimization [Stegmann 2005, Lehmann 2013]. A genetic algorithm has been applied on the material selection optimizations with data mining [Huber 2010].

2.4 Criteria for evaluation and comparison of algorithms

In the mathematical optimization world, the focuses of evaluation of algorithms for solving above problems are mostly on the ability to find the global optimum with as few number of function calls as possible.

In the structural engineering world, the system responses are usually evaluated through finite element analysis and thus function evaluations are much more time-consuming. It is reasonable to assume that the function evaluations dominate the computational cost in the whole optimization procedures. Therefore, the number of function evaluations should be practically come first as the most important criterion before the quality of the solution when algorithms are compared. An applicable algorithm should be able to find an improved solution within acceptable function calls.

3. An efficient procedure for solving nonlinear assignment problems

In this section, a procedure for solving nonlinear assignment problems is developed. Firstly, a linear assignment problem based starting point generation procedure is presented. Then an accelerated neighbor search algorithm is defined. The effective of the procedure is demonstrated through the solution of benchmark QAPLIB problems.

3.1 Linear assignment problem based starting point generation

In this section, a linear assignment problem (LAP) based deterministic procedure to generate a starting point is developed in the following. Staring from the nondiscriminatory assignment matrix $\mathbf{X}^S$, where all the entries are equal to 1/N, where N is the size of the assignment problem, partial derivatives of objective function to $\{x_{ij}\}$ are evaluated. A linear assignment problem is then established with these derivatives as the coefficient matrix. The solution of the LAP problem is then taken as the initial solution.

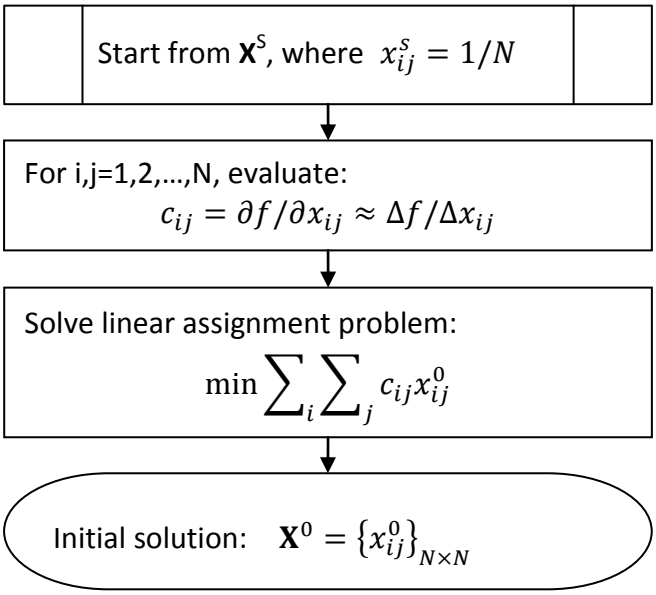


Figure 3-1: Procedure of initial solution generation

Clearly, if the problem is actually a linear assignment type problem as in Section 5.3, the initial solution is just the global optimal as desired.

3.2 Accelerated neighbor search algorithm

As stated in Section 2.2.4.1, the traditional neighbor search algorithm suffers from the problem of low speed on the improvement of design variables, and also suffers from the large computational costs of neighborhood evaluations. How to fully utilize the information obtained in neighborhood evaluation so as to create solutions that may change multiple design variables at the same time is a potential direction of improving the performance of the neighbor search algorithm.

Following this idea, an accelerated neighbor search algorithm is developed. The flowchart of the algorithm is depicted in Figure 3-2.

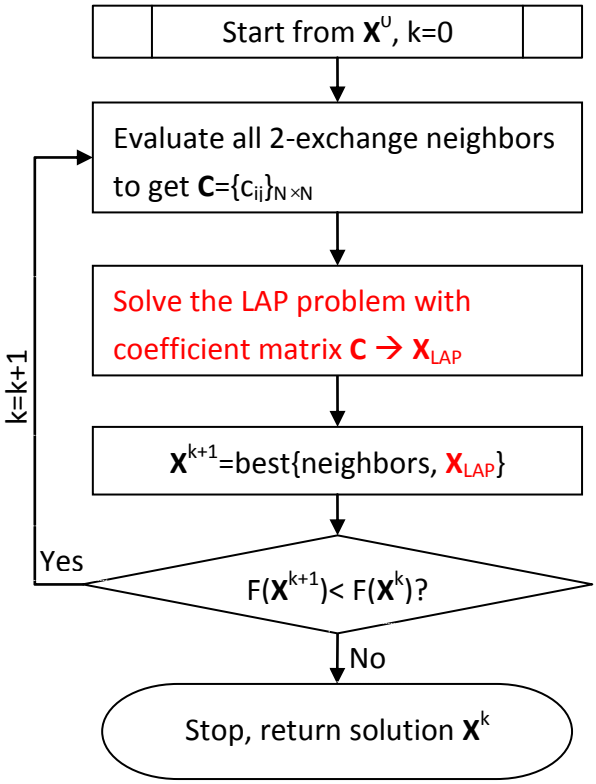


Figure 3-2: Accelerated neighbour search algorithm

Different to the traditional 2-exchange local search algorithm, a LAP problem is established after all neighbors are evaluated, where the coefficient matrix $\{c_{ij}\}$ of the LAP is calculated as follows:

$$c_{ij} = f(\mathbf{T}_{ij}(\mathbf{X})) - f(\mathbf{X}) \tag{3.1}$$

where $\mathbf{T}_{ij}(\cdot)$ is 2-exchange transformation function by which the (i,j) entry of the new design matrix $\mathbf{T}_{ij}(\mathbf{X})$ is equal to 1 through one time two entries exchange of **X**.

The solution of LAP is compared with all the neighbors, and the best one is selected as the result of the current iteration. This improvement provides a possibility to change more than two entries of the assignment matrix in a single iteration, while the increasing of number of function evaluation is only 1.

3.3 Computational complexity analysis

In generating the initial solution, one time function evaluation is required to calculate each partial derivate by finite difference method. The solution of LAP problem requires no objective evaluation. Therefore, $N^2 + 1$ function evaluation is needed in this step, where 1 refers to the evaluation of the obtained initial solution.

In each iteration, every 2-exchange neighbor requires one time function evaluation, which means $C_N^2 = N(N-1)/2$ function evaluations are needed. The evaluation of LAP result needs an additional one time evaluation. Therefore, the number of function evaluation is $N(N-1)/2$+1.

It concludes that the total number of function evaluation of the procedure is:

$$\text{Total}_{\text{func}} = N^2 + 1 + \left[\frac{N(N-1)}{2} + 1\right] \cdot n_{iteration} \tag{3.2}$$

3.4 Test results on Quadratic Assignment Problem Library (QAPLIB)

In this section, the presented optimization procedure is applied on a library of the quadratic assignment problems to demonstrate its applicability and limitations in solving NAP problems. Meanwhile, the computational cost and the effects of the acceleration technique are discussed.

3.4.1 Test suites and numerical results

QAPLIB is a library of the quadratic assignment problems that is firstly published in 1991 [Burkard 1997], which provides a unified test suite for QAP. It contains 133 mathematical QAP problems of size from 10 to 256. To demonstrate the developed optimization procedure is applicable to solve nonlinear assignment problems, it is applied onto this test suite. The objective of the found solution is compared with the global optimum or the best solution so far reported in QAPLIB. The largest 11 problems that of size above 100 are listed in the following table, and the detailed results of all the problems are presented in Appendix A.1

Table 3-1: Results of the largest 11 problems in QAPLIB

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Ite	FuncEva
tai100b	100	1185996137	1200605210	1.23	34	178336
wil100	100	273038	274946	0.70	35	183287
sko100a	100	152002	155898	2.56	33	173385
sko100b	100	153890	156136	1.46	40	208042
sko100c	100	147862	150234	1.60	34	178336
sko100d	100	149576	151860	1.53	34	178336
sko100e	100	149150	153048	2.61	27	143679
sko100f	100	149036	151062	1.36	30	158532
tai150b	150	498896643	509220418	2.07	54	626006
tho150	150	8133398	8293640	1.97	52	603654
tai256c	256	44759294	44879440	0.27	72	2415690

In the following table, the number of problems that can be solved with a relative objective difference less than the specific error levels is presented.

Table 3-2: Problems that are solved with different error levels

Error level (%)	Solved Problem (accumulated)
0	17
<1	57
<5	82
<10	106
<20	117

It shows that the developed procedure is not able to find the global optimal of the nonlinear assignment problems for most of the problems. But it is able to find a high quality solution which is close to the global or best found ones in the measure of relative error for 82 problems which is over 62% of the total problems.

3.4.2 Computational cost analysis

In Section 3.3, the computational complexity of the procedure has been discussed. The only vague point in Eq.(3.2) is the number of iterations. It is common sense to say more iteration will be needed for a larger size of problem.

To quantify the relation, pairs of the size of problems (N) and the number of iterations needed (Ite) are scattered by circled dot in 错误!未找到引用源。. A linear regression in the least-squares sense results in a relation:

$$\mathrm{Ite} = 0.31 \cdot \mathrm{N} + 1.58 \tag{3.3}$$

While a second degree polynomial curve fitting results in:

$$\mathrm{Ite} = -0.0004 \cdot \mathrm{N}^2 + 0.3732 \cdot \mathrm{N} + 0.1005 \tag{3.4}$$

The coefficient of the second-order term is approximately zero, which shows a linear relation between the number of iteration and the size of a problem as in Eq.(3.3) is proper. The relation is depicted also in 错误!未找到引用源。 with a straight red line.

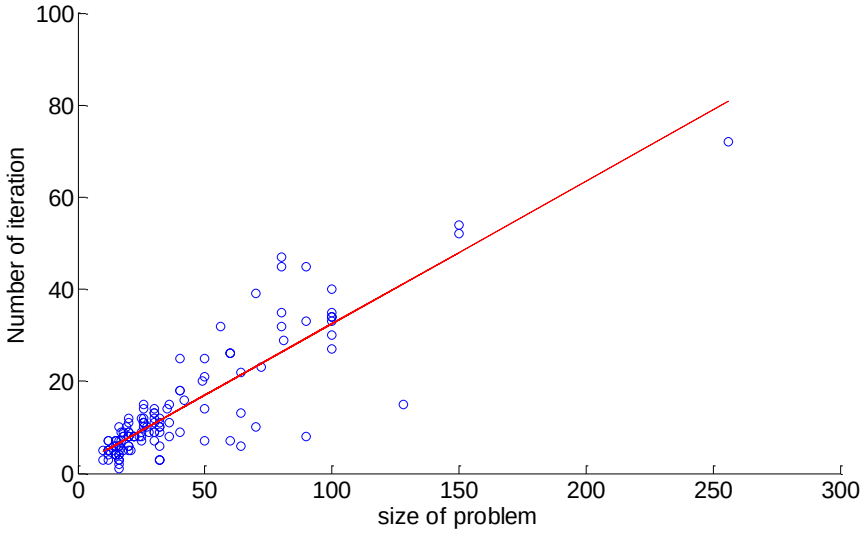


Figure 3-3: Number of iteration V.S. size of problem

Thus the total number of function evaluations is:

$$\mathrm{Total}_{\mathrm{func}} \approx N^2 + 1 + \left[\frac{N(N-1)}{2} + 1\right] \cdot N \cdot (0.31N + 1.58) \quad \sim \mathrm{O}(N^3) \tag{3.5}$$

It proves that the developed procedure of third-order polynomial time complexity.

3.4.3 Effects of the developed acceleration technique through comparison with a traditional neighbor-search algorithm

To demonstrate the advantage of the acceleration technique that developed in Section 3.2, a comparison is made between the results of traditional neighbor-search and the accelerated neighbor-search method. They both start from the starting points generated in Section 3.1.

Table 3-3: Results of the largest 11 problems in QAPLIB

Problem	ANS			NS		
	Error(%)	Ite	FuncEva	Error(%)	Ite	FuncEva
tai100b	1.23	34	178336	4.04	121	608952
wil100	0.70	35	183287	1.14	103	519852
sko100a	2.56	33	173385	2.71	94	475302
sko100b	1.46	40	208042	1.91	132	663402
sko100c	1.60	34	178336	1.99	106	534702
sko100d	1.53	34	178336	1.64	131	658452
sko100e	2.61	27	143679	2.47	96	485202
sko100f	1.36	30	158532	1.49	119	599052
tai150b	2.07	54	626006	3.23	208	2346902
tho150	1.97	52	603654	2.58	166	1877552
tai256c	0.27	72	2415690	0.45	69	2317698

From the quality of solution point of view, the table shows that the ANS is able to reduce the relative error of the final solutions. It means that the solution of LAP which is formed by the linear approximation of the neighborhood structure helps to jump out of local optima.

From the computational cost point of view, it is obvious to see that the acceleration technique can largely reduce the number of iterations required and converge to a final solution much quicker.

In the following figure, the number of iterations for both NS and ANS methods for each test problem is depicted. The linear fitting lines are also presented. For the traditional NS method, the number of iterations approximately following the equation:

$$\mathrm{Ite} = 0.80 \cdot \mathrm{N} - 3.88 \tag{3.6}$$

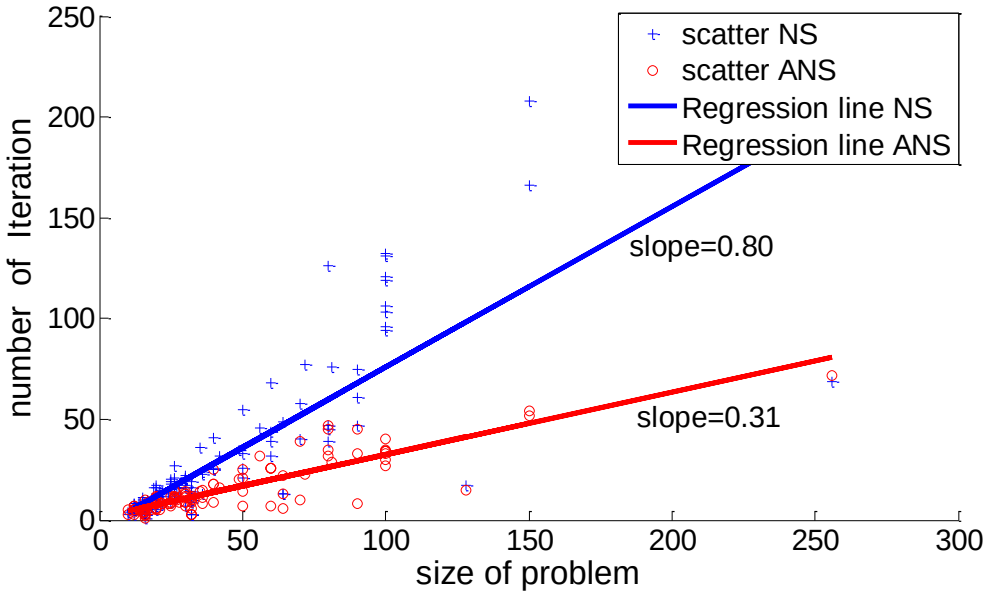


Figure 3-4: Linear regression of number of iteration with size of problem

Therefore, roughly speaking, the number of iteration could be reduced by over 60% with the acceleration technique.

4. An efficient algorithm for MINLP with binary assignment

In this chapter, an algorithm is developed for MINLP problem with binary assignment. The general formulation of this type of problem is introduced at first, and then the algorithm is presented followed by computational cost analysis and demonstration with small-scale examples.

4.1 MINLP with binary assignment

The binary assignment here refers to a MINLP problem with only binary-type integer variables and as the same time, there are assignment constraints on these binaries. The general formulation of such problems is:

$$\begin{gathered} \min f(\boldsymbol{B}, \boldsymbol{x}) \\ \text{s.t.: } g_i(\boldsymbol{B}, \boldsymbol{x}) \leq 0 \quad \text{i=1,2,...,P} \\ \boldsymbol{B} \subseteq \{0,1\}^{M_1+M_2+\cdots+M_N} \\ \boldsymbol{x} \in \boldsymbol{X} \subseteq \boldsymbol{\mathcal{R}}^S \end{gathered} \tag{4.1}$$

where the binary variables are organized into N groups, and for each group, a binary assignment constraint should be satisfied:

$$\sum_{j=1}^{M_i} b_{ij} = 1 \ (i = 1,2, \dots, N) \tag{4.2}$$

The formulation of the integer part of this problem is somehow similar to previously discussed assignment problems. However, it differs in two points: firstly, the number of binary variables in each group may be different. Secondly, there is only row-wise assignment constraints (i.e. constraints on variables of the same group), and no column-wise assignment constraints (i.e. constraints on variables across different groups).

4.2 Constraints-directed binary assignment algorithm

To solve the MINLP problems, a constraints-directed binary assignment algorithm as depicted in Figure 4-1 is developed. The algorithm aims at reducing the number of function evaluations while obtaining a good quality result.

Initially, the algorithm starts from the point where binaries occur in the same equality constraints are equal. At the first step, the binary part is fixed, and a nonlinear programming problem is solved to get a continuous part solution.

Then at step 2, a series of new problems are solved. In each of these new problems, one and only one of the binary variables is changed either from 0 to 1 or from a fractional number to 1. The other binary variables are properly adjusted so that the assignment constraints still hold. The "corresponding" continuous variables to these binaries whose value are changed are also adjusted correspondingly so that the objective value doesn't change. The constraints violation of each solution of these new problems is measured.

The word "corresponding" here refers to the continuous variables that occur in the same terms with the binary variables in the objective function. For instance, in the objective function, if the terms $f_1(b_1)f_2(x_1)+f_3(b_2)f_4(x_2)$ exist, where f_1~f_4 are general functions, then x_1 is a corresponding continuous variable of b_1 and so is x_2 to b_2. If the term $f_1(b_1)f_2(x_1,x_2)$ exists, then both x_1 and x_2 are corresponding variables to b_1. Normally, if there is only one corresponding continuous variables, then the solution for each of the new problem in step 2 is unique, and the constraints violation is defined as the maximum value of all the inequality constraints at the solution point. If there is more than one corresponding variables, then the solutions would be numerous on a super surface, and the constraints violation is defined as the least constraints violation among all solutions.

At step 3, one binary variable among the variables in a same equality constraints is selected to be set to 1 with the others are set to 0. This variable is chosen as it has the minimum constraints violations in step 2 comparing to others binaries belong to the same group.

If the integer part is unchanged after step 2 and step 3, then the procedure will stop. Otherwise, a nonlinear programming will be solved with the fixed new binary values. If the objective of the new optimal solution is larger than previous iteration, the procedure also stops. In a straightforward way to speak, the procedure moves to the next iteration only when the integer variables are changed and lead to a better solution.

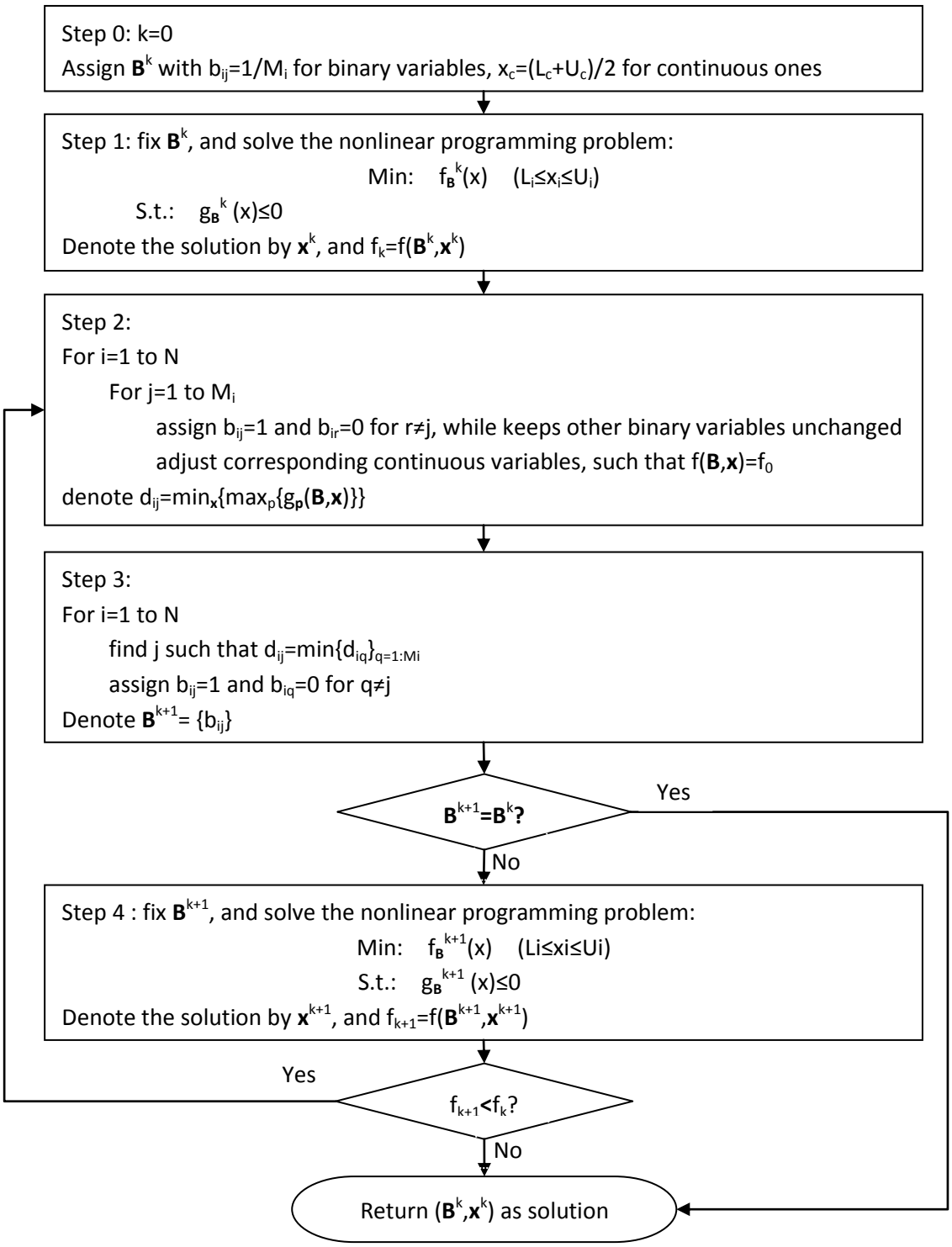


Figure 4-1: Procedure to generate initial solution for the problem

It should be noted that, this algorithm requires the objective and constraints are continuous functions of both continuous and binary design variables. Under the continuity of objective and constraints functions, the evaluation with continuous-relaxed binary variables is mathematically meaningful in Step 1. Without this condition, the evaluation of design points with fractional value may

not be possible. Although this restriction limits the applicability of developed algorithms, most of practical problems in the real world are fit into this category actually.

4.3 Computational complexity analysis

In step 1, the procedure requires one time solution of a nonlinear programming problem. In step 2 of each iteration, one time constraints evaluation is required for each pair of (i,j), therefore $\sum M_i$ function evaluations are needed. And in step 4, one time solution of a nonlinear programming problem is carried out. The total number of computational cost of the procedure is $\mathrm{num}_{\mathrm{Iteration}} + 1$ solution of nonlinear programming problem with N continuous variables plus $\sum M_i \cdot \mathrm{num}_{\mathrm{Iteration}}$ constraints evaluations.

4.4 Benchmark mathematical examples and comparison of several different algorithms

The performance of the designed algorithm is evaluated firstly with mathematical examples in this section.

In the first example, N=1, M=5, **B**=(b_{1j}) ∈ $\{0,1\}_{1\times 5}$, $1 \le x \le 100$. The objective function is:

$$\min\ \mathrm{f}(\mathrm{B},\mathrm{x}) = (2.7\mathrm{b}_{11} + 7.85\mathrm{b}_{12} + 4.5\mathrm{b}_{13} + 8.9\mathrm{b}_{14} + 1.8\mathrm{b}_{15}) \cdot \mathrm{x} \qquad (4.3)$$

The row-wise assignment constraint:

$$\sum_{\mathrm{j}=1}^{5} \mathrm{b}_{1\mathrm{j}} = 1 \qquad (4.4)$$

And an inequality constraint:

$$2000 - \mathrm{x} \cdot (143\mathrm{b}_{11} + 235\mathrm{b}_{12} + 539\mathrm{b}_{13} + 294\mathrm{b}_{14} + 210\mathrm{b}_{15}) \le 0 \qquad (4.5)$$

The global optimal of the problem could be obtained by enumerate all possible value of $\mathbf{B}$ where only one entry equals to 1 and others equals to 0. The global optimal is reached at b_{13}=1 , x=3.7106, with the objective of 16.70.

By applying developed procedure on this problem, firstly a linear programming problem is to be solved:

$$\min\ \mathrm{f}(\mathrm{B},\mathrm{x}) = 5.15\mathrm{x} \qquad (4.6)$$

$$\text{s.t. } 2000 - 284.2\text{x} \leq 0 \tag{4.7}$$

, where the optimal is $x_0 \approx 7.0373$ with $f_0 \approx 36.24$. In step 2, the binary variables are changed with their results listed in following Table 4-1.

Table 4-1: Results of the demonstration example at step 2

	b_{11}=1	b_{12}=1	b_{13}=1	b_{14}=1	b_{15}=1
$x_{adjusted}$	13.42	4.62	8.05	4.07	20.13
f_0	36.24	36.24	36.24	36.24	36.24
d_{1i}	80.51	915	-2341	803	-2228

d_{11}, d_{12} and d_{14} is larger than zero. It means that if the objective is kept unchanged, the corresponding solutions violate the constraint. The third and the fifth results show that while keeping the objective the same, feasible solutions could be found by adjusting the continuous variable. In addition, since $d_{13}<d_{15}$, which could be imagined as an evidence that the third result lies farther away from the boundary of the feasible domain, and thus it has larger potential to improve the objective than the fifth one. Therefore in Step 3, b_{13}=1 is obtained. With this binary assignment, in step 4, the following linear programming problem is to be solved:

$$\min \ \text{f}(\mathbf{B}, \mathbf{x}) = 4.5\text{x} \tag{4.8}$$

$$\text{s.t. } 2000 - 539\text{x} \leq 0 \tag{4.9}$$

The optimal is x≈3.7106 with objective 16.70. Since it is less than f_0, the algorithm returns to Step 2 to start a new iteration. And the results are listed in Table 4-2.

Table 4-2: Results of step 2 in new iteration

	b_{11}=1	b_{12}=1	b_{13}=1	b_{14}=1	b_{15}=1
$x_{adjusted}$	6.18	2.13	3.71	1.88	9.28
f_0	16.70	16.70	16.70	16.70	16.70
d_{1i}	1116	1500	0	1448	51.95

Since the all the other selection of binary variables lead to infeasible solutions, the result of obtained in step 3 is still b_{13}=1 which equals to the previous iteration. Thus one of the stopping criteria is met and returns the final solution with b_{13}=1, x=3.7106. This result is just the global optimal of the problem.

In a second example, the other conditions are the same as the first one, except that the inequality constraint is more highly nonlinear:

$$\frac{2000}{\text{e}^{\text{x}} \cdot (5\text{x}{\text{b}_{11}}^5 + 4\text{x}^2{\text{b}_{12}}^4 + 3\text{x}^3{\text{b}_{13}}^3 + 2\text{x}^4{\text{b}_{14}}^2 + \text{x}^5\text{b}_{15} + 1)} - 1 \leq 0 \tag{4.10}$$

The global optimal of the problem could be also obtained by enumerate all possible value of **B**. The global optimal is b_{15}=1, x=2.675 with the objective of 4.814.

The solution of nonlinear programming problem is carried out SQP algorithm provided by MATLAB. The comparison with state of the art deterministic mixed integer sequential quadratic programming algorithm (MISQP), OA based branch and cut algorithm, branch and bound algorithm is presented in Table 4-3.

Table 4-3: Results of different algorithms

Algorithm	Optimal point	Objective	error(%)	num_{func}
Constr.-directed	b_{15}=1 x=2.675	4.814	0	53
MISQP	b_{15}=1 x=2.675	4.814	0	26
OA based Branch&Cut	b_{15}=1 x=2.675	4.814	0	333
Branch&Bound	b_{15}=1 x=2.675	4.814	0	646

It could be seen that the developed algorithm requires much less total number of function evaluations than the branch and bound and QA based branch and cut algorithm.

It should be noted that, although in above two examples the developed algorithm reaches global optimal by all the algorithms, this should not be expected in much more complicated cases due to the nonlinear properties of the problem. Therefore, the comparison should still focus on the computational cost to find a good enough solution.

5. Optimal parts allocation for structural systems

In this chapter, the mathematical formulation of optimal parts allocation problem for structural systems is established. The optimization procedure developed in section 3 is applied. The efficiency of the procedure is demonstrated with small-scale to large-scale examples.

5.1 Introduction to optimal parts allocation problem

In a mechanical structure, it is often the case that many of the parts are nominally identical. As shown in Figure 5-1 of a 10-bar truss structure, all of the ten bars are designed to have the same length, the same cross-section area, made of the same material and thus mutually exchangeable.

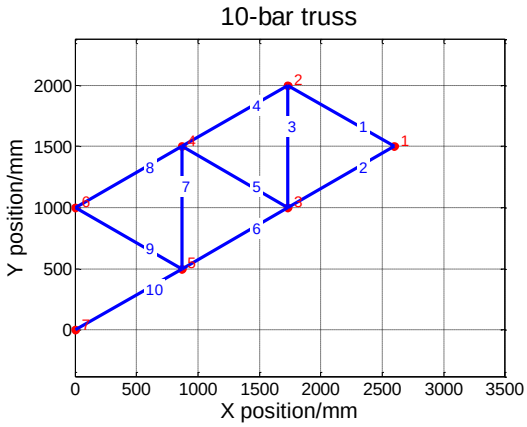


Figure 5-1: A 10-bar truss structure

However, in reality, there are always slight differences in physical and geometrical properties due to variation of material and manufacturing errors in real manufactured bars. The problem of optimal parts allocation for a structural system is to optimize performance of the structure to be manufactured, for example to minimize deformation or stress under certain loads, by assigning parts at hand to their most proper positions in the structure during the assembling period. In this way, the best structure performance is achieved without demanding any extra work in manufacture process, and thus is of great value in practice [Zhang 2011].

5.2 Mathematical modeling into an assignment problem

In a static structural system, taking bar truss structures as an example, the displacements of all degrees of freedom under certain external loads is the solution of the master equations of a finite element system:

$$\mathbf{K} \cdot \mathbf{U} = \mathbf{F} \tag{5.1}$$

, where $\mathbf{K}$ is the system stiffness matrix, which is dependent on area of cross-section (A) and material properties such as Elastic Modulus (E) of each bar. $\mathbf{U}$ is the unknown displacements vector of all degrees of freedom to be solved. $\mathbf{F}$ is the external loading vector, which could also be related to properties of each bar, for example related to the coefficient of thermal expansion (α) of each bar if the structure is under thermal loads.

Assume there are N mutually exchangeable bars are to be assembled into N different positions of a structure system. Number the N parts at hand from 1 to N, and each with properties $\mathrm{A_j},\ \mathrm{E_j},\ \alpha_\mathrm{j}\ (\mathrm{j} = 1,2,\dots,\mathrm{N})$. Also, number the N position in the structure from 1 to N, and denote properties of the bar allocated at each position by $\mathrm{A(i)},\ \mathrm{E(i)},\ \alpha(\mathrm{i})$. A binary matrix $\mathbf{X} = (\mathrm{x_{ij}}) \in \{0,1\}_{\mathrm{N\times N}}$ is employed to represent the allocation of the N parts, where

$$\mathrm{x_{ij}} = \begin{cases} 1 & \text{if jth part is allocated to position i} \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

Clearly $\mathbf{X}$ should satisfy the so-called assignment constraints:

$$\sum_{j=1}^{N} x_{ij} = 1\ (i = 1,2,\dots,N) \tag{5.3}$$

$$\sum_{i=1}^{N} x_{ij} = 1\ (j = 1,2,\dots,N) \tag{5.4}$$

Since $\mathrm{A(i)} = \mathrm{A_j}$ if and only if jth bar is allocated to position i, therefore, areas of cross-section at each position follows the interpolation equation:

$$A(i) = \sum_{j=1}^{N} x_{ij} A_j \quad (i = 1,2, \dots, N) \tag{5.5}$$

The same schemes are also applied for other properties like Young's modulus E and coefficient of thermal expansion. With these equations, the stiffness matrix $\mathbf{K}$ and the load vector $\mathbf{F}$ are both expressed as a function of $\mathbf{X}$, and the unknown displacement components are

$$\mathbf{U}(\mathbf{X}) = \mathbf{K}^{-1}(\mathbf{X}) \cdot \mathbf{F}(\mathbf{X}) \tag{5.6}$$

, which is usually a highly nonlinear functions of $\{x_{ij}\}$. Other system responses such as stresses in the structure that could further be derived from **U**, and thus are also nonlinear functions of $\{x_{ij}\}$. Therefore, the parts allocation problem is formulated as a nonlinear assignment problem.

For a dynamic structural system, there is no essential difference with the static case. The only differences lie in the velocity and acceleration terms in the master equation and corresponding increase in the time of each finite element analysis.

5.3 Examples and numerical results of parts allocation w.r.t. coefficient of thermal expansions

Firstly, a special case of the problem is investigated, where two assumptions to simplify the problem are made: the first one is that, all properties of bars are assumed to be the same except CTEs; the second one is that the objective is a linear function of the displacement vector of the finite element system.

For a single bar element with area of cross-section A, elastic modulus E, coefficient of thermal expansion α, the thermal force load along the bar due to a change of temperature ΔT is:

$$F = EA\alpha\Delta T \tag{5.7}$$

which is a linear function of α. And thus the thermal force vector F_T could be assembled as:

$$\mathrm{F_T}(\mathbf{X}) = \mathbf{T} \cdot \begin{bmatrix} \alpha(1) \\ \alpha(2) \\ \vdots \\ \alpha(\mathrm{N}) \end{bmatrix} = \mathbf{T} \cdot \mathbf{X} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_\mathrm{N} \end{bmatrix} \tag{5.8}$$

, where $\mathbf{T}=\{t_{ij}\}$ is a transformation matrix which depends only on the configuration and field of temperature of the structure. The total load vector, which is composed of mechanical force load $\mathbf{F_M}$ that is independent on the allocation of parts and thermal load $\mathbf{F_T}$ is:

$$\mathbf{F}(\mathbf{X}) = \mathbf{F_T}(\mathbf{X}) + \mathbf{F_M} \tag{5.9}$$

According to the first assumption, the stiffness matrix **K** is independent on the design variables. And due to the second assumption objective function f is in form of:

$$f(\mathbf{U}(\mathbf{X})) = \mathbf{L} \cdot \mathbf{U}(\mathbf{X}) \tag{5.10}$$

where **L** is a linear operator. According to Eq.(5.6), the objective function f could be formulated as:

$$f(\mathbf{X}) = f_m + \widetilde{\mathbf{T}} \cdot \mathbf{X} \cdot \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_\mathrm{N} \end{bmatrix} \tag{5.11}$$

where

$$f_m = \mathbf{L} \cdot \mathbf{K^{-1}} \cdot \mathbf{F_M} \tag{5.12}$$

is irrelevant to **X.** And

$$\widetilde{\mathbf{T}} = \{\tilde{\mathrm{t}}_\mathrm{i}\} = \mathbf{L} \cdot \mathbf{K^{-1}} \cdot \mathbf{T} \tag{5.13}$$

Therefore,

$$f(\mathbf{X}) = f_m + \sum_{\mathrm{i}=1}^{\mathrm{N}} \tilde{\mathrm{t}}_\mathrm{i} \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{x_{ij}} \alpha_\mathrm{j} = f_m + \sum_{\mathrm{i}=1}^{\mathrm{N}} \sum_{\mathrm{j}=1}^{\mathrm{N}} (\tilde{\mathrm{t}}_\mathrm{i} \alpha_\mathrm{j}) \mathrm{x_{ij}} \tag{5.14}$$

Denote:

$$\mathrm{c_{ij}} = \tilde{\mathrm{t}}_\mathrm{i} \alpha_\mathrm{j} \tag{5.15}$$

Then:

$$f(\mathbf{X}) = \sum_{i=1}^{N}\sum_{j=1}^{N} c_{ij}x_{ij} \tag{5.16}$$

which falls into a linear assignment problem with respect to design variables $\{x_{ij}\}$, and could be directly solved by algorithms mentioned in Section 2.2.2.

The difficulty remains to be solved is how to calculate coefficients c_{im}. In structural systems, the inverse of stiffness matrix $\mathbf{K}^{-1}$, and transformation matrix **T** is usually hard to achieve explicitly. Therefore, a direct evaluation of c_{ij} is not possible. Notice that the linearity of objective with respect to $\{x_{ij}\}$, combinatorial coefficient $\{c_{ij}\}$ could be derived accurately with the finite differencing scheme:

$$c_{ij} = \frac{\partial f(\mathbf{X})}{\partial x_{ij}} = \frac{f(\mathbf{X} + \Delta_{ij}) - f(\mathbf{X})}{\Delta_{ij}} \tag{5.17}$$

, where Δ_{ij} is a small increment only at the (i,j) entry of **X**. Due to the continuity of objective function with respect to design variables, the evaluation of objective function at $\mathbf{X} + \Delta_{ij}$ is mathematically meaningful, though it does mean anything physically.

A numerical example with the 10-bar truss structure is presented as follows.

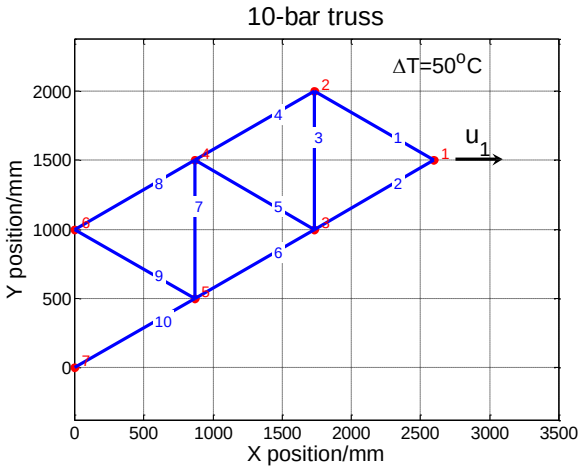


Figure 5-2: 10-bar truss under thermal load (L=1000mm, E=68.95Gpa, $A=1000mm^2$)

As depicted in Figure 5-2, a thermal load of $\Delta T = 50℃$ is applied on the 10-bar truss structure, under which the structure would deform to expand. There are ten bars with the same length, the same Young's modulus, and the same areas of cross-section. But their coefficients of thermal expansion are different as shown in Table 5-1 . The problem is to find the optimal allocation of these ten bars so as to minimize the horizontal displacement at node 1.

Table 5-1: CTEs of ten bars available

Bar No.	1	2	3	4	5
CTE ($\times 10^{-6}/℃$)	1.475	1.116	1.303	1.243	1.446
Bar No.	6	7	8	9	10
CTE($\times 10^{-6}/℃$)	1.381	1.228	1.009	1.411	1.222

To derive $\{c_{ij}\}$, let $\mathbf{X}=\{1/10\}_{10\times10}$, and $\Delta_{ij}= 10^{-1}$. The coefficient matrix then is calculated following Eq.(5.17).

$$\{c_{ij}\} = \begin{bmatrix} +0.043 & +0.032 & +0.038 & +0.036 & +0.042 & +0.040 & +0.035 & +0.029 & +0.041 & +0.035 \\ +0.043 & +0.032 & +0.038 & +0.036 & +0.042 & +0.040 & +0.035 & +0.029 & +0.041 & +0.035 \\ -0.043 & -0.032 & -0.038 & -0.036 & -0.042 & -0.040 & -0.035 & -0.029 & -0.041 & -0.035 \\ +0.043 & +0.032 & +0.038 & +0.036 & +0.042 & +0.040 & +0.035 & +0.029 & +0.041 & +0.035 \\ +0.043 & +0.032 & +0.038 & +0.036 & +0.042 & +0.040 & +0.035 & +0.029 & +0.041 & +0.035 \\ +0.000 & +0.000 & +0.000 & +0.000 & +0.000 & +0.000 & +0.000 & +0.000 & +0.000 & +0.000 \\ -0.043 & -0.032 & -0.038 & -0.036 & -0.042 & -0.040 & -0.035 & -0.029 & -0.041 & -0.035 \\ +0.085 & +0.064 & +0.075 & +0.072 & +0.083 & +0.080 & +0.071 & -0.058 & +0.081 & +0.071 \\ +0.043 & +0.032 & +0.038 & +0.036 & +0.042 & +0.040 & +0.035 & +0.029 & +0.041 & +0.035 \\ -0.043 & -0.032 & -0.038 & -0.036 & -0.042 & -0.040 & -0.035 & -0.029 & -0.041 & -0.035 \end{bmatrix}$$

And the optimal result return by Hungarian algorithm is

$$\{x_{ij}\} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

which corresponds to the allocation of ten bars as shown in Table 5-2. The optimum objective value is 0.1096mm.

Table 5-2: Optimal allocation of bars

Position No.	1	2	3	4	5
Allocated bar No.	7	2	1	3	4
CTE ($\times 10^{-6}/℃$)	1.228	1.116	1.475	1.303	1.243
Position No.	6	7	8	9	10

Allocated bar No.	6	5	8	10	9
CTE ($\times 10^{-6}$/℃)	1.381	1.446	1.009	1.222	1.411

By enumerating all the possible combinations which has a total number of 10!=3628800, it proves that the result obtained by above procedure is the global optimum of the problem. The total number of function evaluations required is just 10^2+1=101, which is significantly fewer than the factorial number.

5.4 Examples of parts allocation w.r.t. CTE, E and area of cross-section and comparison of different algorithms

Three representative parts allocation problems are presented in this section, with which the performance of previous established procedure is evaluated.

5.4.1 A 10-bar truss structure (a small scale single group example)

In this case, both a mechanical force F_M= 29.4kN and a thermal load $\Delta T = 42.37℃$ are applied on the 10-bar structure. The properties of all the bars are design to be the same with E=68.95Gpa, A=1000mm^2, and CTE=23.6$\times 10^{-6}$/℃. But actually, all three properties of the bars are different to each other slightly. The displacement of node 1 is to be minimized under the loading condition.

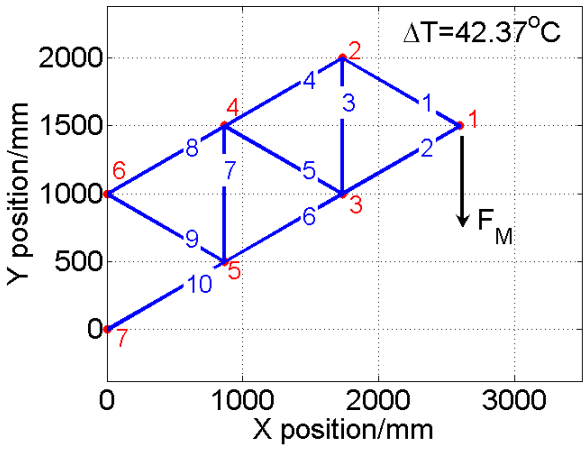


Figure 5-3: 10-bar truss under both mechanical forces and a uniform thermal load

Three different strategies are compared to solve the problems: “Random start + NS” stands for the method of 2-exchange neighbor search algorithm starting from randomly generated initial solutions; “LAP start + NS” stands for the method of 2-exchange neighbor search algorithm starting from the point that is generated by procedure in Section 3.1; and “LAP start + ANS” stands for the method of accelerated neighbor search algorithm and starting from the point that is generated in Section 3.1.

To well demonstrate the performance of developed procedures, results of three different levels of deviation: 5%, 10%, 50% are separately presented. For each of these deviation levels, 10 instances are generated, and their average results are shown in Table 5-3 to Table 5-5.

The following notations are used in tables: $error_{initial}$ is the average relative error of the objective of the initial solution with respect to that of the global optimum; $error_{final}$ is the average relative error of the objective of the final solution; $P_{success}$ is the percentage of successful runs, in which the relative error of the final solution is less than 0.5%; num_{ite} is the average number of iterations and num_{func} is the average total number of function evaluations. For comparison, the global optimum of the problem is found by enumeration of all the possible combinations.

Table 5-3: Result with 5% properties deviation of 10-bar truss example

Method	$error_{initial}$	$error_{final}$	$P_{success}$	num_{ite}	num_{func}
Random start + NS	6.69%	0.04%	95.9%	8.1	365
LAP start + NS	0.00%	0.00%	100%	3.1	241
LAP start + ANS		0.00%	100%	2.6	221

Table 5-4: Result with 10% properties deviation of 10-bar truss example

Method	$error_{initial}$	$error_{final}$	$P_{success}$	num_{ite}	num_{func}
Random start + NS	11.6%	0.06%	97.9%	8.3	376
LAP start + NS	0.00%	0.00%	100%	3.3	250
LAP start + ANS		0.00%	100%	2.7	225

Table 5-5: Result with 50% properties deviation of 10-bar truss example

Method	$error_{initial}$	$error_{final}$	$P_{success}$	num_{ite}	num_{func}
Random start + NS	122.5%	0.56%	84.1%	7.7	347
LAP start + NS	0.98%	0.00%	100%	2.8	227
LAP start + ANS		0.00%	100%	2.9	234

The results show that the initial solution generated by presented procedure is of very high quality in all three deviation levels. With the deviation level increases, the initial error also increases, but is still much smaller than random generated ones. The reduction on number of iteration and number of function evaluations of the accelerated neighbor search procedure could slightly be seen from these small scale problems.

The necessities of parts allocation optimization when deviations or imperfections exist are worth to be mentioned here with this example. Assume all the bars in the structure is perfect, which means that they have the exactly the same modulus, area of cross-section and CTE as they are designed to be. The structure analysis result shows that the displacement at node 1 is 9.77mm under the given loads. It means that the structure is design to have a maximum 9.77mm displacement when the loads are applied. If deviations exist, the optimal allocation of the parts with the best objective (the lease displacement in this example) can be obtained. The worst allocation (with the largest displacement in this example) could also be obtained by minimizing the negative objective function with the developed optimization procedure.

The average best objective and the worst objective values of 10 instances with 5%, 10%, 50% deviations are listed in Table 5-6. It could be seen that with the large level of deviation, the performance of the worst allocation deteriorate significantly. With a 50% deviation, the performance of the structure may fail the design by as large as 126% if the manufactured parts are unluckily inappropriately allocated. Therefore, the parts allocation optimization is suggested to be taken into consideration during the assembly especially when the manufacturing error is lack of control.

Table 5-6: Influence of deviations on the 10-bar truss structure

Deviation level	Obj. best/mm	Obj. worst/mm
5%	9.15 (-6.37%)	10.4 (+6.4%)
10%	8.87 (-9.23%)	11.1 (+13.6%)
50%	6.20 (-36.6%)	22.1 (+126%)

5.4.2 A 25-bar truss structure with stress constraints (a medium scale multiple groups example)

Further, a parts allocation problem of a 3D 25-bar truss structure is presented here. The 25 bars are divided into 8 groups, and each group has 1, 4, 4, 2, 2, 4, 4, 4 bars respectively as colored in Figure 5-4, the members of each group is listed in Table 5-7. Bars of the same group could be exchanged with each other and

they differ in E, CTE and A. The idea values of these properties are designed to be identical as in Section 5.4.1.

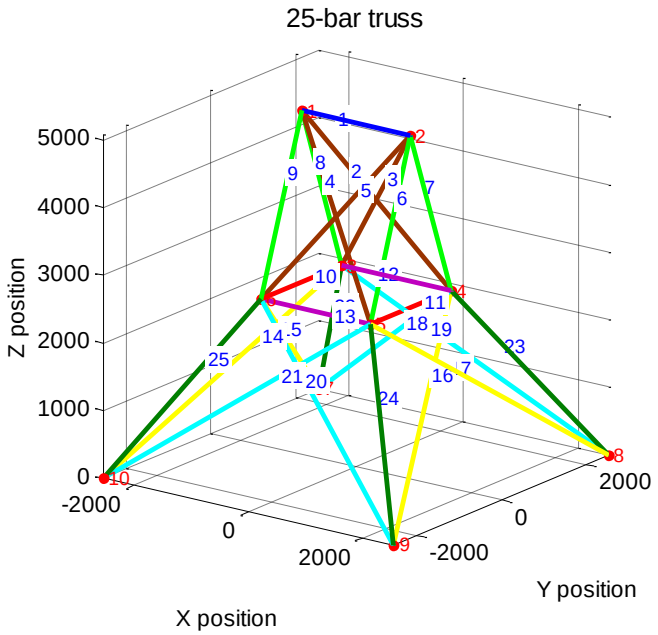


Figure 5-4: 25-bar truss structure

Table 5-7: Composition of groups

Group No.	Bar No.
1	1
2	2,3,4,5
3	6,7,8,9
4	10,11
5	12,13
6	14,15,16,17
7	18,19,20,21
8	22,23,24,25

Together with a uniform thermal load of $\Delta T = 42.37℃$, three different load cases are applied onto the structure respectively, where mechanical forces are different as listed in Table 5-8. The objective is to minimize the displacement of node 1 under each loading condition, while maximum allowable stress constraints are taken into consideration. The stress limit of each load case is artificially set to be equal to the maximal stress of the structure when all components are prefect.

Table 5-8: Load cases for 25-bar structure

Load case	Nodes	Mechanical forces			Allowable Stress
		Fx/kN	Fy/kN	Fz/kN	/Mpa
1	1	4.45	-44.5	-44.5	117
	2	0	-44.5	-44.5	
	3	2.22	0	0	
	6	2.67	0	0	
2	1	0	89.0	-22.2	135
	2	0	-89.0	-22.2	
3	1	4.45	44.5	-22.2	125
	2	0	44.5	-22.2	
	3	2.22	0	0	
	6	2.22	0	0	

The penalty method is applied to take constraints into consideration, where the objective function is modified as:

$$\text{objective} = disp_{node1} + \text{p} \cdot \max(0, \frac{\sigma_{\max}}{\sigma_{\text{allowable}}} - 1) \tag{5.18}$$

p is the penalty factor and equals to 10^4 in this example.

For a multiple-group problem, where total parts are divided into several groups and parts among each group are mutually exchangeable, the developed starting point generation procedure is applied on each group separately, while keeping the assignment of other group nondiscriminatory.

Due to the formulation of the problem and properties of interpolation scheme, the objective function is continuous with respect to the entries of assignment matrix. The partial derivatives of objective function could be approximated by finite difference method or other sensitivity analysis technique, which make the procedure mathematical meaningful.

After obtaining a starting point, the accelerated local search algorithm is employed to solve the problems. Still, for a multiple-group problem, the LAP-improvement is applied on each group separately.

The total number of function evaluations required by above procedure is the same as in Section 3.3. And for a multiple groups problem, where total N parts are divided into M groups with each have $\{\text{N}_\text{i}\}$ parts. (i=1,2,…,M, $\sum \text{N}_\text{i} = \text{N}$), total number of function evaluations is expressed:

$$\text{Totalfunc} = \sum_{i=1}^{M} N_i^2 + 1 + \left[\sum_{i=1}^{M} \frac{N_i(N_i-1)}{2} + 1\right] \cdot n_{iteration} \sim O(N^2) \quad (5.19)$$

10 instances are generated with maximum properties deviation of 10% for each load case. The global optimal is still found by enumerating all the possible combinations with total number of $1!\,4!\,4!\,2!\,2!\,4!\,4!\,4! = 31\,850\,496$. Average results of all three cases are presented in Table 5-9, where $vio_{initial}$ is the average of stress violation of initial solutions.

Table 5-9: Average results of 25-bar truss structure

Method	$error_{initial}$	$vio_{initial}$	$error_{final}$	$P_{success}$	num_{ite}	num_{func}
Random start + NS	4.73%	1.07%	0.22%	88.5%	14.3	460
LAP start + NS	1.82%	0.27%	0.03%	100%	5.5	265
LAP start + ANS			0.03%	100%	3.4	202

It could be seen, the procedure could return a starting point with both better objective and less violation of the stress constraint. The quality of final solution is also higher with a reduction in total number of function evaluations. The LAP-improvement strategy could well reduce the number of iterations needed, and thus less total number of function evaluations is required.

5.4.3 A 72-bar truss structure (a large scale multiple groups example)

Finally, a large scale 72-bar truss allocation problem is employed to demonstrate the procedures. The 72-bar truss structure is shown in Figure 5-5, where bars are divided into 4 groups with 8, 16, 16, 32 exchangeable bars respectively and makes total number of combinations of $8!\,16!\,16!\,32! \approx 4.6 \times 10^{66}$.

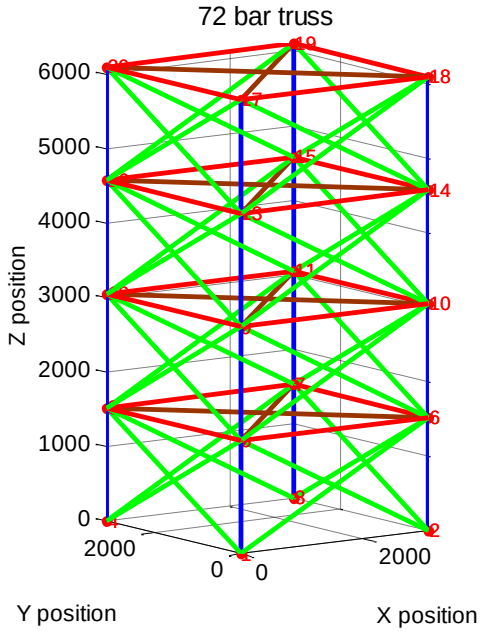


Figure 5-5: 72-bar truss structure

Together with a uniform thermal load of $\Delta T = 42.37℃$, two different mechanical load cases are applied onto the structure respectively as listed in Table 5-10. The objective is to minimize the displacement of node 20. Cases both with and without stress constraints are discussed.

Table 5-10: Load cases for 72-bar structure

Load case	Nodes	Mechanical forces			Allowable Stress /Mpa
		Fx/kN	Fy/kN	Fz/kN	
1	17	22241	22241	-22241	109
2	17	0	0	-22241	81
	18	0	0	-22241	
	19	0	0	-22241	
	20	0	0	-22241	

10 instances are generated with maximum properties deviation of 5% for each load case. The average results are presented in Table 5-11 and

Table 5-12 respectively. Due to innumerous total combinations, there is no way to find the global optimum in this problem. So for each instance, the best solution obtained by all the three methods is taken as the reference solution and the relative error are calculated with respect to it.

Table 5-11: Average results of 72-bar truss structure without stress constraints

Method	$error_{initial}$	$error_{final}$	$P_{success}$	num_{ite}	num_{func}
Random start + NS	11.6%	0.03%	100%	122.5	93571

LAP start + NS	0.16%	0.01%	100%	26.9	22114
LAP start + ANS		0.01%	100%	10.7	9748

Table 5-12: Average results of 72-bar truss structure with stress constraints

Method	$error_{initial}$	$vio_{initial}$	$error_{final}$	$P_{success}$	num_{ite}	num_{func}
Random start + NS	11.5%	2.62%	0.20%	84.8%	131.6	100546
LAP start + NS	4.79%	0.36%	0.03%	100%	72.9	57258
LAP start + ANS		0.36%	0.02%	100%	18.8	15945

Numerical results show that presented procedures could create a high quality starting point from both view of objective and constraints. Although the procedure requires larger number of function evaluations at the beginning compared with other initial solution generation methods, the reduction of total computational cost could more than compensate for it in the whole optimization process.

The results also show that accelerated neighbor search algorithm could reduce the relative error of final solution with respect to the global optimum. It significantly increases the speed of improvement of the intermediate step solutions and thus largely reduces number of iterations it needed. From small scale to large scale problems, it could be seen that, the larger the scale of problem is, the more significant the reduction will be.

6. Stacking sequence optimization (SSO) in fibre-reinforced laminate structure design

In this section, stacking sequence optimization in fibre-reinforced laminate structure design is investigated. The problem is formulated as a nonlinear assignment problem. The developed optimization procedure presented in Chapter 3 is transplanted to solve this problem, where necessary adjustments are made regarding the starting point generation. To improve the feasibility of solutions with respect to the stacking sequence constraints, a so-called repair operator is developed. It adjusts the design of the stacking sequence properly so as to reduce the degree of constraints violations. The operator is proven to be able to obtain high quality feasible solutions with much less computational cost.

6.1 Introduction to stacking sequence optimization

Fibre-reinforced laminates are composite materials constructed by stacking multiple fiber-reinforced layers in a specific sequence. Each layer or ply consists of parallel fibers embedded in a matrix (Figure 6-1 left). The orientation of fibers of different layers could be varied with respect to a common reference axes (Figure 6-1 right).

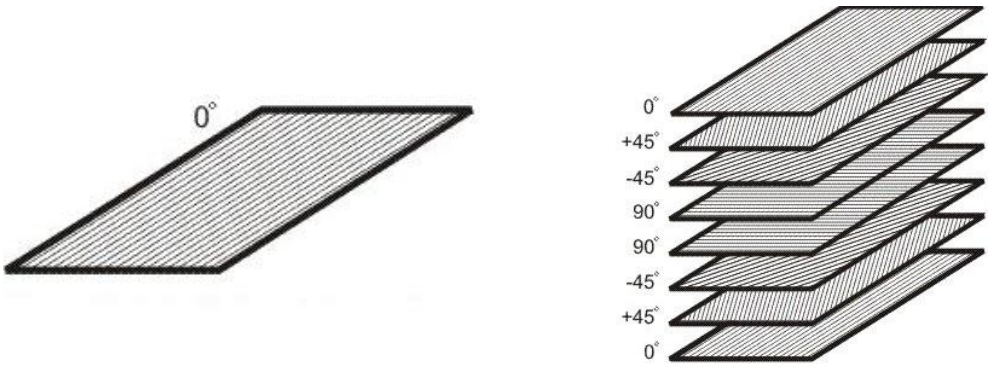


Figure 6-1: An individual fibre-reinforced ply (left) and a 8-layer fibre-reinforced laminate stacking with plies in four different orientations (right)

There are mainly three aspects in designing a laminate composite material: ply material selection, ply thickness and orientation optimization, and finally the stacking sequence optimization. In the first two steps, the composition of each ply and the number of plies in each direction are to be determined. The stacking

sequence optimization determines in which order all these plies should be combined together in the end so that mechanical properties are optimized.

6.2 Theory of fibre-reinforced laminate analysis

For better understanding the optimization approach, the theory of classical fibre-reinforced laminate analysis is briefly introduced here as background knowledge.

Given the properties of layers and the stacking sequence of the layers, the properties of a laminate material are determined through classical laminate theory. Further, the mechanical performance of the structure could be evaluated by closed-formed formula for simple supported 2D panel or by finite element analysis for more complicated structure.

6.2.1 Material properties matrices (A B D matrix)

Given the layer material properties: two in-plane Young's modulus E_1, E_2 in longitudinal and transverse directions, Poisson's ratio ν_1, ν_2 and modulus E_3 in vertical direction, they have the following relations:

$$\nu_2 = \frac{E_2 \nu_1}{E_1} \tag{6.1}$$

Then material invariants Q, U and V are calculated as [Tsai1968]:

$$\begin{aligned} Q_{11} &= \frac{E_1}{1-\nu_1\nu_2} \\ Q_{22} &= \frac{E_2}{1-\nu_1\nu_2} \\ Q_{12} &= \frac{E_2\nu_1}{1-\nu_1\nu_2} \\ Q_{66} &= E_3 \end{aligned} \tag{6.2}$$

And U

$$\begin{aligned} U_1 &= \frac{1}{8}(3Q_{11} + 3Q_{22} + 2Q_{12} + 4Q_{66}) \\ U_2 &= \frac{1}{2}(Q_{11} - Q_{22}) \\ U_3 &= \frac{1}{8}(Q_{11} + Q_{22} - 2Q_{12} - 4Q_{66}) \\ U_4 &= \frac{1}{8}(Q_{11} + Q_{22} + 6Q_{12} - 4Q_{66}) \end{aligned} \tag{6.3}$$

$$U_5 = \frac{1}{8}(Q_{11} + Q_{22} - 2Q_{12} + 4Q_{66})$$

And V_A

$$V_{A0} = \sum_k (h_{k+1} - h_k)$$

$$V_{A1} = \sum_k (h_{k+1} - h_k)\cos 2\theta_k$$

$$V_{A2} = \sum_k (h_{k+1} - h_k)\sin 2\theta_k \qquad (6.4)$$

$$V_{A3} = \sum_k (h_{k+1} - h_k)\cos 4\theta_k$$

$$V_{A4} = \sum_k (h_{k+1} - h_k)\sin 4\theta_k$$

In these formulas, the variable h_k represents the coordinate of the layer to the midplane of the laminate.

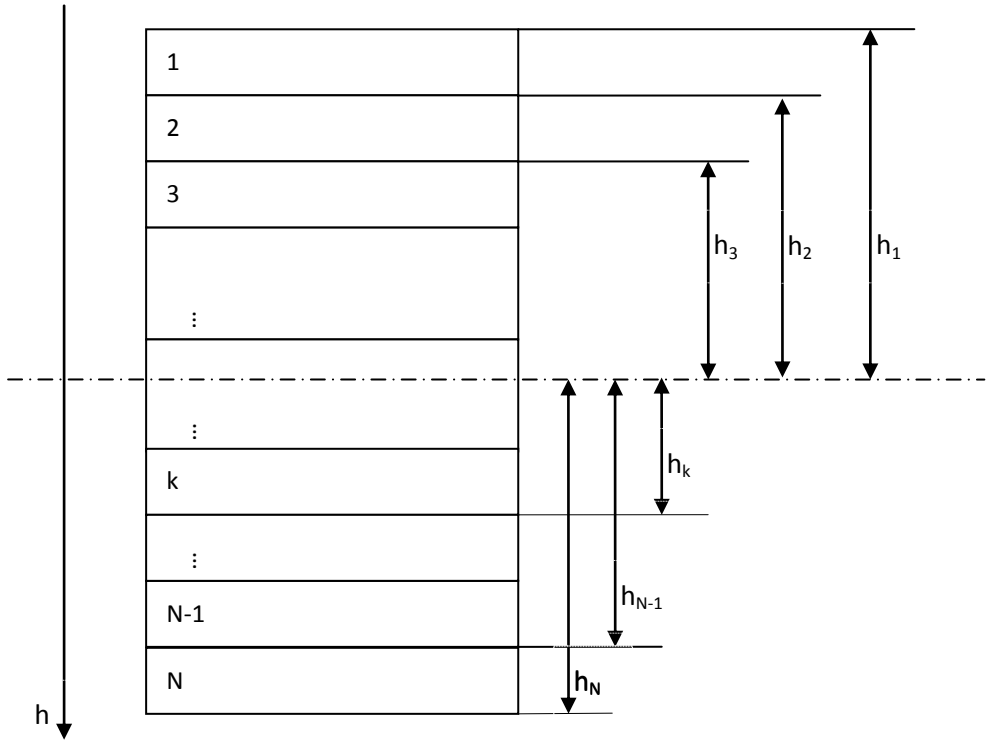


Figure 6-2: Distance of the layer to the midplane

And V_D

$$V_{D0} = \frac{1}{3}\sum_{k}\left(h_{k+1}^{3} - h_{k}^{3}\right)$$

$$V_{D1} = \frac{1}{3}\sum_{k}\left(h_{k+1}^{3} - h_{k}^{3}\right)\cos 2\theta_{k}$$

$$V_{D2} = -\frac{1}{3}\sum_{k}\left(h_{k+1}^{3} - h_{k}^{3}\right)\sin 2\theta_{k} \qquad (6.5)$$

$$V_{D3} = \frac{1}{3}\sum_{k}\left(h_{k+1}^{3} - h_{k}^{3}\right)\cos 4\theta_{k}$$

$$V_{D4} = -\frac{1}{3}\sum_{k}\left(h_{k+1}^{3} - h_{k}^{3}\right)\sin 4\theta_{k}$$

Then, stiffnesses A_{ij} are obtained by:

$$\begin{aligned} A_{11} &= U_1 V_{A0} + U_2 V_{A1} + U_3 V_{A3} \\ A_{22} &= U_1 V_{A0} - U_2 V_{A1} + U_3 V_{A3} \\ A_{12} &= U_4 V_{A0} - U_3 V_{A3} \\ A_{66} &= U_5 V_{A0} - U_3 V_{A3} \\ A_{16} &= -0.5 \cdot (U_2 V_{A2} + 2U_3 V_{A4}) \\ A_{66} &= -0.5 \cdot (U_2 V_{A2} - 2U_3 V_{A4}) \end{aligned} \qquad (6.6)$$

flexural stiffnesses D_{ij}

$$\begin{aligned} D_{11} &= U_1 V_{D0} + U_2 V_{D1} + U_3 V_{D3} \\ D_{22} &= U_1 V_{D0} - U_2 V_{D1} + U_3 V_{D3} \\ D_{12} &= U_4 V_{D0} - U_3 V_{D3} \\ D_{66} &= U_5 V_{D0} - U_3 V_{D3} \\ D_{16} &= -0.5 \cdot (U_2 V_{D2} + 2U_3 V_{D4}) \\ D_{66} &= -0.5 \cdot (U_2 V_{D2} - 2U_3 V_{D4}) \end{aligned} \qquad (6.7)$$

The coupling stiffnesses B_{ij} are equal to zero when the laminate is symmetric. The symmetry of a composite material is defined as layers are distributed symmetrically to the midplane. The symmetry assumption ensures a balanced distribution of layer directions and usually provides a better load bearing abilities under complicated cases. Therefore it is usually true and practical in real world manufacturing and design.

The material stiffness matrices are assembled as:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{16} \\ A_{12} & A_{22} & A_{26} \\ A_{16} & A_{26} & A_{66} \end{bmatrix} \tag{6.8}$$

$$B = \mathbf{0}_{3\times 3} \tag{6.9}$$

$$D = \begin{bmatrix} D_{11} & D_{12} & D_{16} \\ D_{12} & D_{22} & D_{26} \\ D_{16} & D_{26} & D_{66} \end{bmatrix} \tag{6.10}$$

With these stiffness matrices, plate constitutive equation can be established:

$$\begin{bmatrix} N_x \\ N_y \\ N_{xy} \\ M_x \\ M_y \\ M_{xy} \end{bmatrix} = \begin{bmatrix} A & B \\ B & D \end{bmatrix} \cdot \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_{xy} \\ \kappa_x \\ \kappa_y \\ \kappa_{xy} \end{bmatrix} \tag{6.11}$$

6.2.2 Strain failure load factor

The maximum strain theory states that if any of strains in the principal material directions exceeds allowable values, then the material will fail. The relation of strains in the longitudinal direction ε_L, transverse direction ε_T and shear strain ε_{LT} and that in common-reference coordinate system are:

$$\begin{Bmatrix} \varepsilon_L \\ \varepsilon_T \\ \frac{1}{2}\varepsilon_{LT} \end{Bmatrix} = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 2\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & -2\sin\theta\cos\theta \\ -\sin\theta\cos\theta & \sin\theta\cos\theta & \cos^2\theta - \sin^2\theta \end{bmatrix} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \frac{1}{2}\varepsilon_{xy} \end{Bmatrix} \tag{6.12}$$

the strain failure load factor is:

$$\lambda_{strain} = \min\{\frac{\varepsilon_{LU}}{\varepsilon_L}, \frac{\varepsilon_{TU}}{\varepsilon_T}, \frac{\varepsilon_{LTU}}{\varepsilon_{LT}}\} \tag{6.13}$$

6.2.3 Tsai-Wu failure criterion load factor

There are several material failure theories regarding stresses. Tsai–Wu failure criterion is commonly employed for anisotropic composite material due to different strengths in tension and compression. The criterion states that the stresses of any point in a plane stress should satisfy:

$$F_{11}\sigma_L^2 + 2F_{12}\sigma_L\,\sigma_T + F_{22}\sigma_T^2 + F_{66}\tau_{LT}^2 + F_1\sigma_L + F_2\sigma_T \leq 1 \tag{6.14}$$

where $F_{11} = \frac{1}{\sigma_{Lt}\sigma_{Lc}}$, $F_{22} = \frac{1}{\sigma_{Tt}\sigma_{Tc}}$, $F_{66} = \frac{1}{\tau_s^2}$, $F_{12} = -\frac{1}{2}\sqrt{F_{11}F_{22}}$, $F_1 = \frac{1}{\sigma_{Lt}} - \frac{1}{\sigma_{Lc}}$, $F_2 = \frac{1}{\sigma_{Tt}} - \frac{1}{\sigma_{Tc}}$. σ_{Lt} and σ_{Lc} are longitudinal tensile and compressive strengths respectively, σ_{Tt} and σ_{Tc} are transverse tensile and compressive strengths respectively, τ_S is shear strength. The stresses components in principal material axes are given:

$$\begin{Bmatrix} \sigma_L \\ \sigma_T \\ \tau_{LT} \end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{12} & Q_{22} & 0 \\ 0 & 0 & 2Q_{66} \end{bmatrix} \begin{Bmatrix} \varepsilon_L \\ \varepsilon_T \\ \frac{1}{2}\gamma_{LT} \end{Bmatrix} \tag{6.15}$$

Given a load condition $\mathbf{F}$, the strength load factor is defined as

$$\mathbf{F}^{\mathbf{critical}} = \lambda_\sigma \mathbf{F} \tag{6.16}$$

For a linear structure system, the stresses under the critical load and given load have following relations:

$$\sigma_*{}^{\text{critical}} = \lambda_\sigma \sigma_* \tag{6.17}$$

Replace equation (6.17) into (6.14) and get:

$$\lambda_\sigma^2 \Sigma_1 + \lambda_\sigma \Sigma_2 - 1 \leq 0 \tag{6.18}$$

where $\Sigma_1 = F_{11}\sigma_L^2 + 2F_{12}\sigma_L\sigma_T + F_{22}\sigma_T^2 + F_{66}\tau_{LT}^2$, $\Sigma_2 = F_1\sigma_L + F_2\sigma_T$.

By solving the inequality, the maximum load factor regarding Tsai-Wu criterion could be obtained.

6.2.4 Critical buckling load analysis

Laminate composites are widely used in the manufacturing of structural parts in the aerospace industries. They are often operated under compression loadings and thus buckling behavior has to be taken into consideration during the design. Critical buckling load analysis is a method used to determine critical loads at which structural parts become unstable and their corresponding buckling modal shapes.

Finite element method is used for the analysis of complex composite laminate structures. However, for simple composite plate under simple boundary conditions, there are strict theoretical analyses or approximations to predict the instabilities. These closed-form predictions provide ways of highly efficient structural evaluation for practice. In the following, an approximation of the critical buckling load factors for orthotropic composite laminated plates subject to uniformly distributed in-plane loads are introduced as an example. The

structure is depicted in Figure 6-3. It will be further employed in the demonstration of the developed optimization procedure.

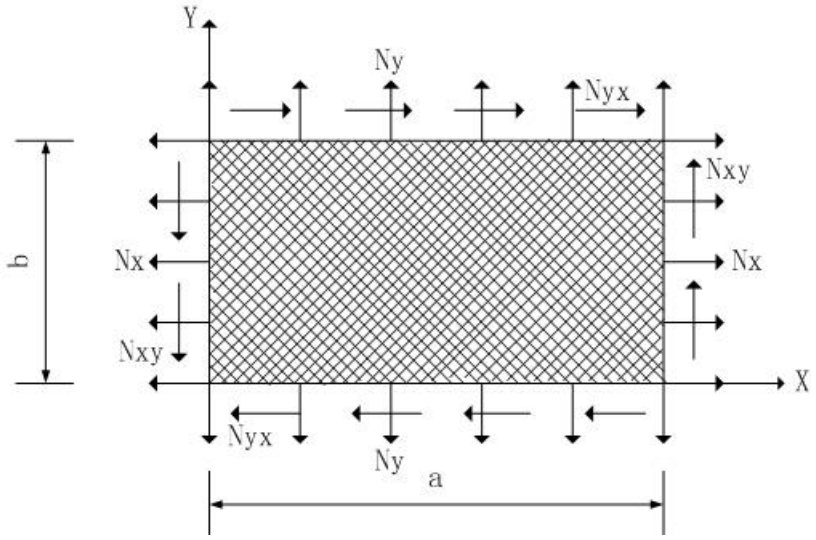


Figure 6-3: Composite laminated plate under in-plane loads

Theoretical critical buckling load factor of the plate under only normal loads N_x and N_y is

$$\lambda_{nc} = \pi^2 \cdot \frac{D_{11}(\frac{m}{a})^4 + 2(D_{12} + 2D_{66})(\frac{m}{a})^2(\frac{n}{b})^2 + D_{22}(\frac{n}{b})^4}{(\frac{m}{a})^2 N_x + (\frac{n}{b})^2 N_y} \tag{6.19}$$

where (m, n) is the pair of number of half-waves that minimizes λ_{nc}.

If the plate is under only shear loads, a finite element analysis is normally required to get the buckling load factor. When the plate is assumed to have an infinite length in x direction, an analytical solution is given:

$$\lambda_{sc} = \begin{cases} \dfrac{4\beta(D_{11}D_{22}^3)^{1/4}}{b^2 N_{xy}} & 1 \le \Gamma \le \infty \\ \dfrac{4\beta\sqrt{D_{22}(D_{12} + 2D_{66})}}{b^2 N_{xy}} & 0 \le \Gamma < 1 \end{cases} \tag{6.20}$$

where

$$\Gamma = \frac{\sqrt{D_{11}D_{22}}}{D_{12} + 2D_{66}} \tag{6.21}$$

and β is given in Table 6-1.

Table 6-1: Empirical value of coefficient β regarding shear buckling load factor

Γ	β
0.0	11.71

Γ	β
0.2	11.80
0.5	12.20
1.0	13.17
2.0	10.80
3.0	9.95
5.0	9.25
10.0	8.70
20.0	8.40
40.0	8.25
∞	8.13

When normal and shear loads are applied at the same time to the plate, their interaction should be considered. And the normal buckling load factor is modified by:

$$\frac{1}{\lambda_n} = \frac{1}{\lambda_{nc}} + \frac{1}{\lambda_{sc}{}^2} \tag{6.22}$$

To prevent the buckling under both shear and normal loads, both λ_n and λ_{sc} should be larger than one. Therefore the critical buckling load factor is:

$$\lambda_c = \min\{|\lambda_{sc}|, \lambda_n\} \tag{6.23}$$

6.3 Mathematical formulation of stacking sequence optimization into a nonlinear assignment problem

In stacking sequence optimization problem, all the plies are given, and the number of plies in each orientation is fixed. Assume total number of N plies in M different orientations $\theta_1, \theta_2, \ldots, \theta_M$ are given, and in direction $\theta_m$there are N_m plies (m=1, 2,..., M). A binary matrix $X = \{x_{ij}\} \in \{0,1\}^{N\times N}$ could be used to represent the sequence order from top surface to bottom, where $x_{ij} = 1$ means that the ith layer counting from the top is arranged with the jth ply. If j≤N_1, then it means a ply in orientation θ_1 should be arranged in layer i. In general, if $N_1 + N_2 + \cdots + N_{k-1} < j \leq N_1 + N_2 + \cdots + N_k$, then it means a ply in direction θ_k should be arranged. Denote:

$$\boldsymbol{\Theta} = [\underbrace{\theta_1, \theta_1, \ldots, \theta_1}_{N_1}, \underbrace{\theta_2, \theta_2, \ldots, \theta_2}_{N_2}, \ldots, \underbrace{\theta_M, \theta_M, \ldots \theta_M}_{N_M}]^{\mathbf{T}} \tag{6.24}$$

Then ply direction could be expressed:

$$\boldsymbol{\Theta}(\mathrm{k}) = \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{x}_{\mathrm{kj}}\theta_j \tag{6.25}$$

Taking entries of **X** as design variables, it is obvious that they should satisfy the assignment constraints:

$$\sum_{j=1}^{N} x_{ij} = 1\ (i = 1,2,\dots,N) \tag{6.26}$$

$$\sum_{i=1}^{N} x_{ij} = 1\ (j = 1,2,\dots,N) \tag{6.27}$$

Once a stacking sequence is chosen, mechanical performance of the structure that is made of this material could be usually calculated through finite element codes for a complex structure or by classical laminate composites theory for a laminate plate. These results then go into the objective or constraint function evaluations.

In practice, there is always a contiguity constraint which helps to prevent cracks going through many layers. This constraint requests that contiguous plies of the same orientation should not exceed a specific number. A function named violation number (V_{num} for short) is defined to measure in which degree the sequence violates the contiguity condition. Assume the number of maximal allowable consecutive layers in the same direction is N_c, the value of V_{num} is calculated as follows: initially, set V_{num} equals to zero. Then check from the topmost layer till the bottom layer, if more than N_c layers of the same orientation are detected, the violation number is added by the number of consecutive layers of same direction minus N_c. So the contiguity constraint could be formulated as:

$$\mathrm{V}_{\mathrm{num}}(\mathbf{X}) = 0 \tag{6.28}$$

A penalty is added to the objective if this constraint is violated. Thus the optimal stacking sequence problem is formulated into a constrained nonlinear assignment problem. The total number of possible combinations equal to $N!/(\prod_i^M N_i!)$, which is always a huge number due to the existence of the factorial operator.

6.4 Improvements on the optimization procedure

6.4.1 Linear assignment based starting point generation

Since the problem is also formulated into a nonlinear assignment problem, the procedure depicted in Figure.3-1 is employed here to generate the starting point. The key point to apply the procedure lies in how to define the "partial derivatives" of objective with respect to $\{x_{ij}\}$ and evaluate with fraction in the assignment matrix.

Two methods are presented in the following to tackle this problem. When the structural evaluation codes allows for external input of **A**, **B**, **D** matrices, the first method is suitable. Otherwise, the second technique could be used instead. The ideas of these methods are identical: firstly, "homogenized" the laminate structure, secondly, assign one and only one layer to a specific angle while keeping other layers under homogenized state, and then the objective function value of the "perturbed" structure is defined as the corresponding entry in the **C** matrix of a linear assignment problem.

6.4.1.1 Method with interpolation of material stiffness matrices

There are finite element analysis codes such as ANSYS that allow input of material stiffness matrices **A**, **B**, **D** as material properties, with which the structural performance could be evaluated.

To evaluate the objective function at continuous-relaxed design points, an interpolation scheme is defined:

$$f(\boldsymbol{\Theta}(\mathrm{k})) = \sum_{\mathrm{j}=1}^{\mathrm{N}} \mathrm{x_{kj}} \cdot f(\theta_j) \tag{6.29}$$

where $f(\cdot)$ is a general function of ply angle. Apply this scheme to trigonometric function in Eq.4.4 and Eq.4.5. Then the **A**, **B**, **D** matrices are as follows:

$$\begin{aligned} \mathrm{V_{A0}} &= \sum_{\mathrm{k}} (\mathrm{h_{k+1}} - \mathrm{h_k}) \\ \mathrm{V_{A1}} &= \sum_{\mathrm{k}} (\mathrm{h_{k+1}} - \mathrm{h_k}) \sum_{\mathrm{j}} \mathrm{x_{kj}} \cos 2\theta_j \end{aligned} \tag{6.30}$$

$$V_{A2} = \sum_k (h_{k+1} - h_k) \sum_j x_{kj} \sin 2\theta_j$$

$$V_{A3} = \sum_k (h_{k+1} - h_k) \sum_j x_{kj} \cos 4\theta_j$$

$$V_{A4} = \sum_k (h_{k+1} - h_k) \sum_j x_{kj} \sin 4\theta_j$$

$$V_{D0} = \frac{1}{3} \sum_k \left(h_{k+1}^3 - h_k^3\right)$$

$$V_{D1} = \frac{1}{3} \sum_k \left(h_{k+1}^3 - h_k^3\right) \sum_j x_{kj} \cos 2\theta_j$$

$$V_{D2} = -\frac{1}{3} \sum_k \left(h_{k+1}^3 - h_k^3\right) \sum_j x_{kj} \sin 2\theta_j \qquad (6.31)$$

$$V_{D3} = \frac{1}{3} \sum_k \left(h_{k+1}^3 - h_k^3\right) \sum_j x_{kj} \cos 4\theta_j$$

$$V_{D4} = -\frac{1}{3} \sum_k \left(h_{k+1}^3 - h_k^3\right) \sum_j x_{kj} \sin 4\theta_j$$

The **A**, **B**, **D** matrices are presented as continuous functions of design matrix **X**, thus the structural evaluations at continuous-relaxed design points is mathematical meaningful and the procedure developed in Section 3.1 could be directly applied here.

6.4.1.2 Method with quasi-homogenization concept

In many composite laminate finite element analysis codes, such as MSC.Nastran, only the stacking sequence and layer material properties are accepted as inputs to the finite element model. Therefore, interpolations of **A**, **B**, **D** are not useful in this case. To overcome this difficulty, a method with quasi-homogenization concept is presented, with which the initial solution generation procedure could be carried out.

Homogenization is a process to make a mixture of the composite so that the mechanical properties are the same throughout the material. In the first step of the algorithm presented in Section 3.1, an evaluation of **X** with all entries are equal is required. It could be seen as the material is “homogenized”. Therefore, a

procedure to “homogenize” a laminate material so that it is able to be analysed by finite element laminate analysis code, and at the same time its performance is influenced as little as possible by assigned stacking sequence is required.

Given a composite laminate with N layers, the thickness t of each layer is t and stacking sequence is θ_1~ θ_N as shown in the left of Figure 6-4. A corresponding quasi-homogenized laminate is created as depicted in the right of Figure 6-4.

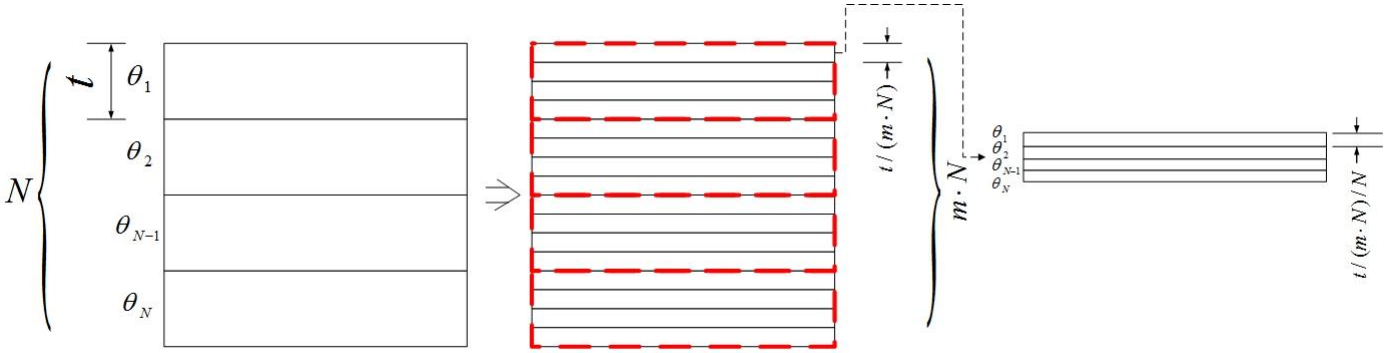


Figure 6-4: Quasi-homogenization of a composite laminate material

The quasi-homogenized laminate is composed of N homogenized layers, where each layer is composed of m sub-layers with thickness of t/(mN). Here m is a preselected integer, the larger the m is, the more close the laminate properties approximate that of the real homogenized laminate.

Each sub-layer is composed again with N ply with thickness t/mN/N, where the stacking sequence of the plies keeps the same as the original laminate. To evaluate the “partial derivative” of design variables, assume it is c_{ij}, the quasi-homogenized material is modified as depicted in Figure 6-5 to reflect the influence of a layer in direction θ_j locates in the i^{th} position: replace the i^{th} homogenized layer with a layer in direction θ_j, while keep the other layers still homogenized.

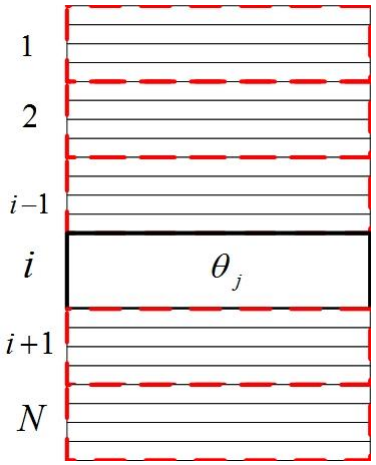


Figure 6-5: Modified quasi-homogenized material to evaluate c_{ij}

6.4.2 Repair operators for contiguity constraints

To improve the efficiency of stacking sequence optimization, so called repair operators are presented in literature, which repair the stacking sequence to force it satisfy the contiguity constraint. A state of the art repair operator is firstly introduced in this section, and then a new repair operator with consideration of maximize degree of improvement on violations is presented. Their efficiencies are compared with numerical examples thereafter.

6.4.2.1 Introduction to Baldwinian repair operator

Todoroki & Haftka [TH1998] introduced a Baldwinian repair strategy for dealing with contiguity constraints for standard GA. The idea of the operator is based on repairing stacking sequence with the nearest different one. The operator checks from the outermost layer to innermost layer, and once a contiguity violation is found in the sequence, the inner ply will be changed with the nearest layer in other directions in the inner side. Then it checks further violation and repair any violation it meets. The advantage of this operator is that it only changes the nearest possible layer each time, which minimize the change of stacking sequence and thus minimize the influence on the material properties. However, the disadvantage of the operator is that it focuses on one violation with each exchange, and thus requires many times of repair work.

6.4.2.2 Maximum improvement repair operator

Applying repair operator to found solutions will change the stacking sequence and thus the objective function value. To limit this influence on system responses as much as possible, one natural idea is to reduce the changes on stacking sequences. Based on the idea to repair as many as possible contiguity violations with the least number of changes of stacking sequence, the following repair operator is developed.

Firstly, a function to measure the degree of improvement is defined as follows:

$$\mathrm{d}_{\mathrm{improve}} = \mathrm{N} \cdot \left(\mathrm{vio}_{\mathrm{original}} - \mathrm{vio}_{\mathrm{repair}}\right) - \left|\mathrm{dist}_{\mathrm{repair}}\right| \tag{6.32}$$

where N is total number of stacks, $vio_{original}$ is number of violation before repair, vio_{repair} is number of violation after the repair, and $dist_{repair}$ is the difference of positions of interchanged stacks in the repair.

The maximum improvement repair operator works as shown in Figure 6-6. When the stacking sequence violates the contiguity constraint, it checks each pairwise exchange of the sequence and carries out the exchange that has the largest degree of improvement. Then the operator applies on the new stacking sequence again until exchanged sequence fulfils the contiguity condition.

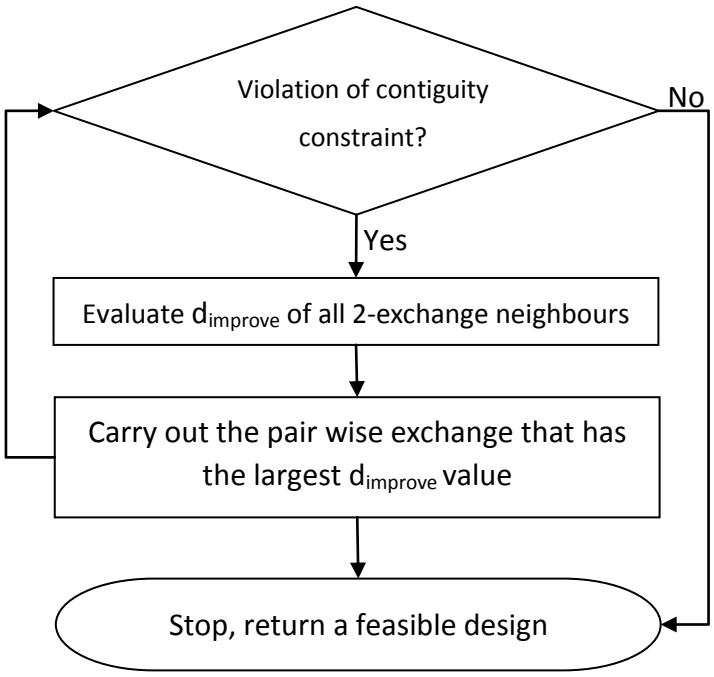


Figure 6-6: Flowchart of maximum improvement repair operator

From the definition of degree of improvement, it is easily seen it has the following properties:

1. If a 2-exchange of sequence repairs at least one contiguity violation, then $d_{improve}$ is positive;

2. The 2-exchange that repairs the most number of violations has the largest degree of improvement, no matter what the distance between the exchanged two layers is;

3. If two 2-exchange repairs the same number of violations, the exchange that involve closer layers has larger degree of improvement.

Therefore, on one hand, this repair operator focuses on reducing the number of violations as many as possible within one 2-exchange. And on the other hand, it succeeds the advantage of Baldwinian operator which chooses exchange that involves layers as near as possible.

The difference of above two repair strategies is better presented with the following example. Given a stacking sequence of $[90_2/90_2/90_2/\pm45/0_2/0_2/0_2]$, and contiguity constraint equals to 4. Therefore, the given sequence violation the constraints twice with 3 successive 90_2 and 0_2 layer respectively.

The Baldwinian operator will first repair the violation caused by $90°$ stacks by exchanging the third 90_2 with ±45 next to it. And then it repairs the violation due to $0°$ stacks by exchanging the first 0_2 with 90_2 next to it. The repair procedure is depicted in the following:

$[90_2/90_2/\underline{90_2/\pm45}/0_2/0_2/0_2]$→$[90_2/90_2/\pm45/\underline{90_2/0_2}/0_2/0_2]$→$[90_2/90_2/\pm45/0_2/90_2/0_2/0_2]$

Evaluating all possible 2-exchange of stacks with respect to degree of improvement, the results are shown in Table 6-2.

Table 6-2: Example of evaluation of degree of improvement ($\mathrm{vio}_{\mathrm{original}} = 2$, N=7)

Pair to exchange	Resulted stacking sequence	$\mathrm{Vio}_{\mathrm{repair}}$	$\mathrm{disp}_{\mathrm{repair}}$	$\mathrm{d}_{\mathrm{improve}}$
$[\underline{90_2}/90_2/90_2/\underline{\pm45}/0_2/0_2/0_2]$	$[\pm45/90_2/90_2/90_2/0_2/0_2/0_2]$	2	3	-3
$[\underline{90_2}/90_2/90_2/\pm45/\underline{0_2}/0_2/0_2]$	$[0_2/90_2/90_2/\pm45/90_2/0_2/0_2]$	0	4	10
$[\underline{90_2}/90_2/90_2/\pm45/0_2/\underline{0_2}/0_2]$	$[0_2/90_2/90_2/\pm45/0_2/90_2/0_2]$	0	5	9
$[\underline{90_2}/90_2/90_2/\pm45/0_2/0_2/\underline{0_2}]$	$[0_2/90_2/90_2/\pm45/0_2/0_2/90_2]$	0	6	8
$[90_2/\underline{90_2}/90_2/\underline{\pm45}/0_2/0_2/0_2]$	$[90_2/\pm45/90_2/90_2/0_2/0_2/0_2]$	1	2	5
$[90_2/\underline{90_2}/90_2/\pm45/\underline{0_2}/0_2/0_2]$	$[90_2/0_2/90_2/\pm45/90_2/0_2/0_2]$	0	3	11
$[90_2/\underline{90_2}/90_2/\pm45/0_2/\underline{0_2}/0_2]$	$[90_2/0_2/90_2/\pm45/0_2/90_2/0_2]$	0	4	10
$[90_2/\underline{90_2}/90_2/\pm45/0_2/0_2/\underline{0_2}]$	$[90_2/0_2/90_2/\pm45/0_2/0_2/90_2]$	0	5	9
$[90_2/90_2/\underline{90_2}/\underline{\pm45}/0_2/0_2/0_2]$	$[90_2/90_2/\pm45/90_2/0_2/0_2/0_2]$	1	1	6
$[90_2/90_2/\underline{90_2}/\pm45/0_2/0_2/0_2]$	**$[90_2/90_2/0_2/\pm45/90_2/0_2/0_2]$**	**0**	**2**	**12**
$[90_2/90_2/\underline{90_2}/\pm45/0_2/\underline{0_2}/0_2]$	$[90_2/90_2/0_2/\pm45/0_2/90_2/0_2]$	0	3	11
$[90_2/90_2/\underline{90_2}/\pm45/0_2/0_2/\underline{0_2}]$	$[90_2/90_2/0_2/\pm45/0_2/0_2/90_2]$	0	4	10
$[90_2/90_2/90_2/\underline{\pm45}/\underline{0_2}/0_2/0_2]$	$[90_2/90_2/90_2/0_2/\pm45/0_2/0_2]$	1	1	6
$[90_2/90_2/90_2/\underline{\pm45}/0_2/\underline{0_2}/0_2]$	$[90_2/90_2/90_2/0_2/0_2/\pm45/0_2]$	1	2	5
$[90_2/90_2/90_2/\underline{\pm45}/0_2/0_2/\underline{0_2}]$	$[90_2/90_2/90_2/0_2/0_2/0_2/\pm45/]$	2	3	11

Therefore, the 2-exchange of the third 90_2 and the first $0_2$is selected. It could also be seen, that Baldwinian operator leads to a result that is different to original one in three bits (the third, the fourth and the fifth), while the result of developed operator differs in only two bits (the third and the fifth) with less number of change of stacking sequences.

6.4.3 Optimization procedure

The accelerated neighbor search algorithm presented in Section 3.2 can be directly applied to solve this problem. Integrating the repair operator, the following initial solution generation procedure in Figure 6-7 and accelerated neighbor-search algorithm depicted in Figure 6-8 are employed.

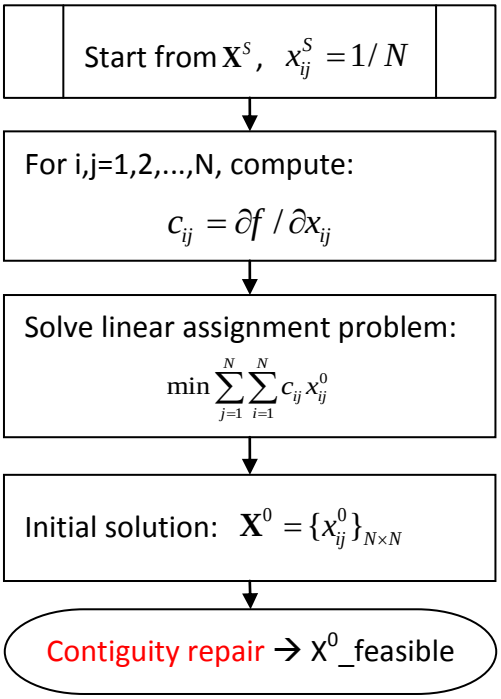


Figure 6-7: Initial generation procedure with repair operator embedded

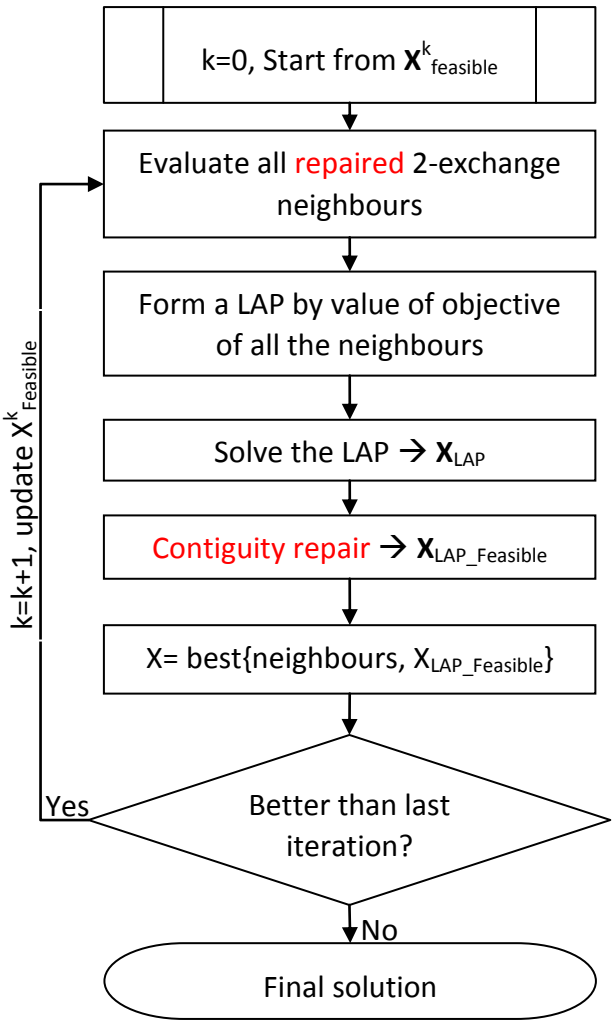


Figure 6-8: Deterministic algorithm with repair operator embedded

In generating the starting point, it requires NM+1 function evaluations. In each iteration, it needs only to evaluate the pairwise exchange between two plies in different directions, and thus calls for $\sum_{i=1}^{M}\frac{N_i(\sum_{j\neq i}N_j)}{2}+1=\sum_{i=1}^{M}\sum_{j=i+1}^{M}N_iN_j+$ 1function evaluations. Therefore, total number of function evaluation is:

$$\text{Totalfunc}= NM+1 +\left[\sum_{i=1}^{M}\sum_{j=i+1}^{M}N_iN_j+1\right]\cdot n_{iteration} \qquad (6.33)$$

It can be derived that Totalfunc$< NM+1+N^2/2$, which means number of function evaluations of the approach is in order of $\mathcal{O}(N^2)$.

6.5 Demonstration through test problems and comparison of different algorithms

6.5.1 A thin panel design examples

The procedure is tested with a simply supported 0.61m square laminar plate as in Figure 6-3, which is widely employed in literature. All the plies are in direction $0°$, $\pm 45°$, $90°$. The laminate is assumed to be in symmetry.

To simplify the problem, the laminate is further assume to be made up of stacks that has two plies of the same orientation, i.e. plies in $0°$ and $90°$ always show up with two together, and each $+45°$ ply is followed by a $-45°$ ply (as shown in Figure 6-9). This assumption is adopted in practice to reduce the complexity in manufacturing.

The material property is listed in Table 6-3. The maximum number of consecutive layers in a same direction is 4. The objective is to maximize the critical buckling load factors under given loads and give number of layers in each direction.

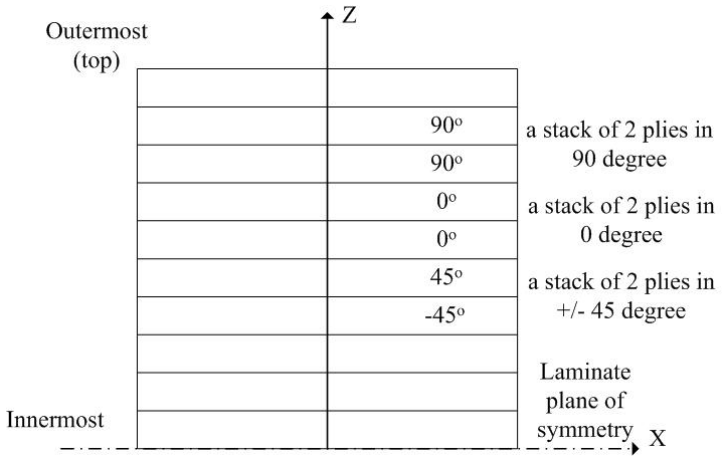


Figure 6-9: Layout of stacking sequence

Table 6-3: Material properties

E_1/Gpa	E_2/Gpa	G_{12}/Gpa	v_{12}
128	13.0	6.4	0.3

With reference to literature, 11 cases shown in

Table 6-4 are employed, where both the composition of plies and total number of stacks are different.

Table 6-4: Test cases

case	Loading(N/mm)			Number of stacks				Total combinations
	Nx	Ny	Nxy	$(0°)_2$	(±45°)	$(90°)_2$	Total	
1	-3500	-350	175	9	18	9	36	4.4×10^{14}
2	-2625	-350	175	8	17	8	33	1.5×10^{13}
3	-1750	-350	175	7	15	7	29	2.7×10^{11}
4	-875	-350	175	6	12	6	24	2.5×10^{9}
5	0	-350	175	4	8	4	16	9.0×10^{5}
6	0	-2800	1400	8	16	8	32	7.7×10^{12}
7	2797	-2584	1778	9	8	13	30	2.9×10^{12}
8	-2915	344	145	13	7	15	35	2.5×10^{14}
9	1865	-2599	1750	14	10	15	39	4.9×10^{16}
10	1865	-2599	1750	14	8	17	39	1.6×10^{16}
11	1973	-3318	1400	10	8	12	30	3.8×10^{12}

6.5.1.1 Optimization without repair operators on the stacking sequence

Firstly, the focus is on comparing different optimization algorithms. Therefore, the repair operators are not involved, and the penalty method is employed to deal with the contiguity constraint.

Results are listed in Table 6-5, where Error is the relative error of objective with respect to that of the known optimal so far. For random start, it is evaluated by the average performance of 100 runs.

Table 6-5: Test results of different deterministic procedure

case	Random start + NS		Random start+ ANS		LAP start + ANS	
	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)
1	7178	0.00	5640	0.00	1433	0.00
2	5162	0.01	3823	0.01	1448	0.00
3	3795	0.01	2455	0.01	868	0.00
4	2093	0.00	1386	0.00	616	0.00
5	577	0.00	331	0.00	211	0.00
6	4635	0.06	3307	0.06	1381	0.12
7	3810	0.00	2844	0.00	1561	0.00
8	7973	0.10	5458	0.10	2458	0.00
9	5976	0.94	5672	0.94	7132	0.35
10	5882	1.20	5309	1.20	4501	2.02

case	Random start + NS		Random start+ ANS		LAP start + ANS	
	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)
11	5104	0.13	3859	0.13	2764	0.00
average	4744	0.22	3644	0.22	2216	0.23

It can be seen that, the accelerated neighbor search procedure significantly outperforms the traditional local-search strategy with a random start in all the cases. Starting from generated high quality initial solution, the procedure requires less function evaluations to reach the final solution.

In addition, a genetic algorithm (GA) and a permutation discrete particle swarm optimization algorithm (PDPSO) which are used to solve these problems in literature [Chen 2008, Chang 2010] are also implemented. The stopping criteria of the heuristic algorithms are set as relative error of the objective with respect to that of the optimal known so far is smaller than 0.5%. And results are compared in Table 6-6.

Table 6-6: Comparison of optimum and number of function evaluations with heuristic algorithms

Case	GA with GR[1]		GA with PMX[1]		PDPSO[2]		LAP start + ANS	
	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)
1	1777	0.42	3950	0.37	1081	0.33	1433	0.00
2	1617	0.43	3281	0.38	836	0.32	1448	0.00
3	1283	0.33	2881	0.29	686	0.29	868	0.00
4	1038	0.33	2462	0.28	457	0.28	616	0.00
5	621	0.37	1148	0.37	301	0.29	211	0.00
6	1626	0.48	3735	0.45	1290	0.41	1381	0.12
7	2487	0.37	3436	0.36	1470	0.39	1561	0.00
8	2429	0.29	15597	0.41	3281	0.31	2458	0.00
9	15684	0.36	22452	0.41	9995	0.36	7132	0.35
10	21024	0.41	23942	0.46	10474	0.38	4501	2.02
11	4196	0.48	9814	0.48	6634	0.47	2764	0.00
average	4889	0.39	8427	0.39	3319	0.35	2216	0.23

1. GA with GR refers to genetic algorithm using gene-rank crossover, GA with PMX refers to genetic algorithm using partially mapped crossover, which are presented in literature. The results are evaluated by taking the average of 100 runs.

2. The results are evaluated also by taking the average of 100 runs.

It shows that the developed deterministic approach can still greatly reduce the computational cost comparing with the genetic algorithm and particle swarm optimization method.

6.5.1.2 Optimization with repair operators on the stacking sequence

The performance of above two repair operators are then evaluated again with test cases presented in

Table 6-4. The repair operators are embedded into all the four algorithms to be compared. For heuristic algorithms, the repair operator is invoked for each generated solution before evaluation. For accelerated neighbor search algorithm, the repair operator is invoked in three different steps as shown in Figure 6-7 and Figure 6-8. The results of Baldwinian operator and operator with maximum degree of improvement (MI) are listed in Table 6-7 and Table 6-8 separately.

Table 6-7: Results with Baldwinian repair operator

	GA+GR		GA+PMX		PDPSO		LAP start + ANS	
	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)
1	1247	0.34	2984	0.33	580	0.31	504	0.00
2	1112	0.33	2148	0.32	511	0.30	429	0.00
3	958	0.34	2146	0.29	392	0.30	336	0.00
4	760	0.32	1708	0.27	299	0.28	245	0.00
5	425	0.35	592	0.25	187	0.25	206	0.00
6	1086	0.38	1619	0.32	485	0.32	721	0.12
7	1126	0.37	2089	0.34	470	0.39	607	0.00
8	1164	0.33	3008	0.28	785	0.27	460	0.00
9	1494	0.39	3710	0.34	382	0.38	1000	0.30
10	2301	0.39	7852	0.37	550	0.39	1848	0.12
11	677	0.32	954	0.28	249	0.37	617	0.00
average	1123	0.35	2619	0.31	445	0.32	633	0.05

Table 6-8: Results with repair considering maximum degree of improvement

	GA+GR		GA+PMX		PDPSO		LAP start + ANS	
	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)
1	1255	0.34	2838	0.27	596	0.31	499	0.00
2	1061	0.34	2574	0.27	554	0.29	417	0.00

	GA+GR		GA+PMX		PDPSO		LAP start + ANS	
	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)	num_{func}	Error (%)
3	933	0.36	2487	0.27	445	0.26	334	0.00
4	732	0.30	1585	0.25	307	0.28	249	0.00
5	487	0.36	684	0.29	191	0.26	126	0.00
6	1201	0.38	1948	0.35	555	0.40	698	0.12
7	1106	0.39	1961	0.31	568	0.38	583	0.00
8	965	0.30	4279	0.25	751	0.28	419	0.18
9	4704	0.44	12439	0.41	2287	0.45	1402	0.93
10	7879	0.44	17990	0.45	3059	0.48	1720	0.00
11	1545	0.38	4346	0.35	1392	0.42	1330	0.00
average	1988	0.37	4830	0.32	973	0.35	707	0.12

In comparison with results in Table 6-6, both repair operators could significantly reduce the number of function evaluations needed. Applying on the developed algorithm, repair operator with maximum degree of improvement could achieve comparable performance as Baldwinian operator, although its performance is worse in collaboration with heuristic algorithms. And still, LAP based deterministic procedure well outperform heuristic ones with both repair strategies.

6.5.2 Problems with large number of layers: thick composite panels design

Further, one large scale example is employed to demonstrate the whole procedure. The composite panel structure is the same as in Figure 6-3. However, two more angles of ply direction $\pm 30°$ and $\pm 60°$ are taken into account. At the same time, total number of stacks increases to 150~200, and contiguity constraint is 10 instead of 4. 11 test cases with different composition of stacks are generated and employed as listed in Table 6-9.

Table 6-9: Test cases of a very thick composite panel

case	Number of stacks						Total combinations
	$0°_2$	$\pm 30°$	$\pm 45°$	$\pm 60°$	$90°_2$	Total	
1	36	72	36	24	24	192	1.1×10^{122}

case	Number of stacks						Total combinations
	$0°_2$	$\pm 30°$	$\pm 45°$	$\pm 60°$	$90°_2$	Total	
2	32	68	32	22	22	176	9.1×10^{110}
3	28	60	28	20	18	154	2.6×10^{96}
4	24	48	24	16	16	128	1.8×10^{80}
5	16	32	16	12	10	86	1.2×10^{53}
6	32	64	32	22	20	170	3.0×10^{107}
7	36	32	52	20	20	160	1.0×10^{103}
8	52	28	60	24	22	186	5.4×10^{118}
9	56	40	60	26	26	208	3.1×10^{135}
10	56	32	68	26	26	208	3.2×10^{133}
11	40	32	48	20	20	160	3.0×10^{103}

The same loading conditions as in

Table 6-4 are applied on corresponding case. And still the critical buckling load factor is taken as the objective to be maximized. Since it has been proved that the accelerated neighbor search algorithm outperforms the heuristic algorithms largely, therefore, only results with this algorithm are discussed here. Results of the two different repair operators are compared in Table 6-10.

Table 6-10: Test cases with different repair operators

	Baldwinian repair		Maximum improvement repair	
	num_{func}	Load factor	num_{func}	Load factor
1	41521	4.04	14727	4.04
2	34590	4.03	12365	4.04
3	26592	3.83	9568	3.83
4	18599	3.75	6780	3.75
5	8510	3.83	3216	3.83
6	32382	3.69	11603	3.69
7	103950	4.83	97062	4.83
8	63723	5.01	13561	5.03
9	48557	14.17	98728	14.17
10	173427	14.16	80918	14.03
11	106010	5.07	145307	5.10
average	59806	----	44894	----

It shows that, for most of test cases, the repair operator with maximum degree of improvement requires fewer number of function evaluations than using Baldwinian repair operator. Comparing with results of small scale problems in Section 6.5.1.1 and 6.5.1.2, the reduction on computational cost is much more significant for larger scale problem. This is because the number of changes required by Baldwinian repair operator is usually larger than that by maximum improvement repair strategy in a large scale problem.

6.6 Application in engineering problems

6.6.1 Composite panel with stiffeners

In this section, the stacking sequence optimizing of a composite panel with stiffeners is presented. The finite element model of the structure and the cross-section of the stiffener are shown in Figure 6-10 and Figure 6-11. Both panel and stiffeners are made of fiber-reinforced composite laminate layers. Each layer is of thickness 0.132mm, the material properties of each layer are listed in Table 6-11, and the composition of layers for stiffener and panel are the same as shown in Table 6-12.

Table 6-11: Material properties

E_1/GPa	E_2/GPa	G_{12}/GPa	v_{12}
128	11.3	6.0	0.3

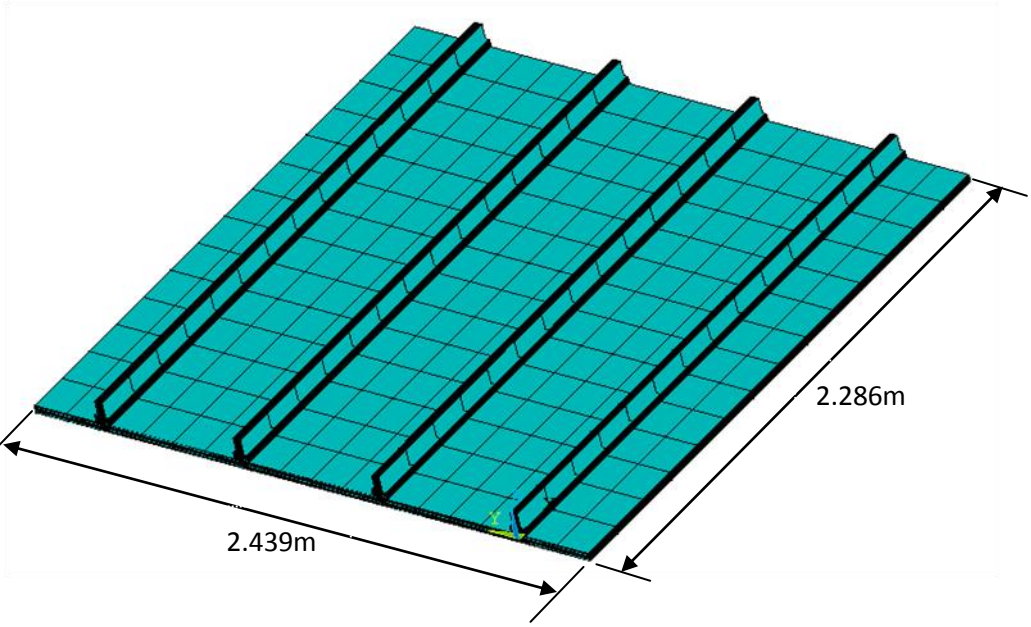


Figure 6-10: Composite panel with stiffeners

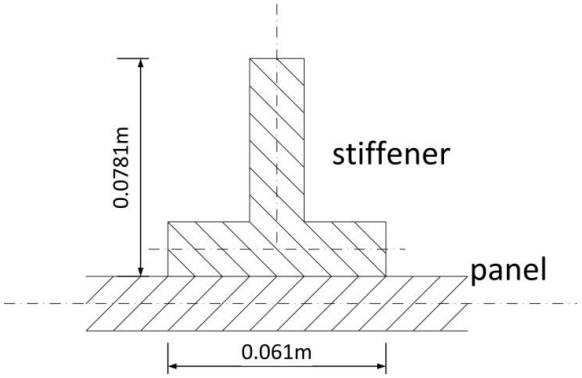


Figure 6-11: Geometry of the cross-section

Table 6-12: Composition of layers

	Number of stacks						Total combinations
	$0°_2$	$\pm 30°$	$\pm 45°$	$\pm 60°$	$90°_2$	Total	
Panel	16	10	10	10	16	62	2.6×10^{78}
Stiffener	16	10	10	10	16	62	

Firstly, only the critical buckling load factor $\lambda_{buckling}$ is taken into account and to be maximized. This factor is calculated directly by finite element analysis. The boundary and loading conditions are presented in Figure 6-12. The plate is simply supported and a uniformly distributed normal load N_x of total 2.45×10^5N is applied on edges that are vertical to the stiffener direction. And the contiguity constraint is 10.

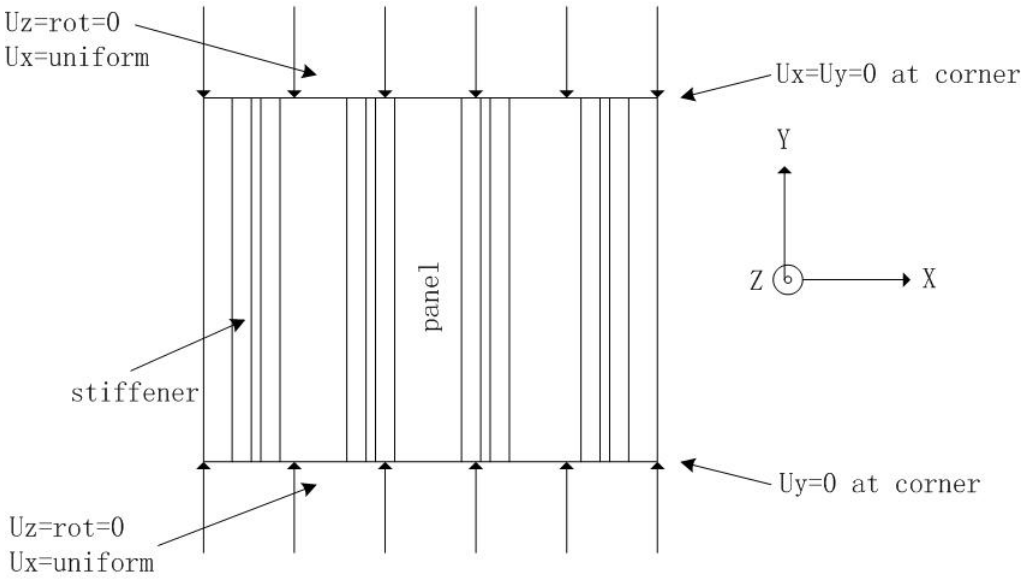


Figure 6-12: Boundary conditions and normal loads

The optimization procedure terminates after 10 iterations with total 30548 number of finite element analyses. The critical load factor of the optimal is 1.8579. The first-order buckling mode of the structure is depicted in Figure 6-13.

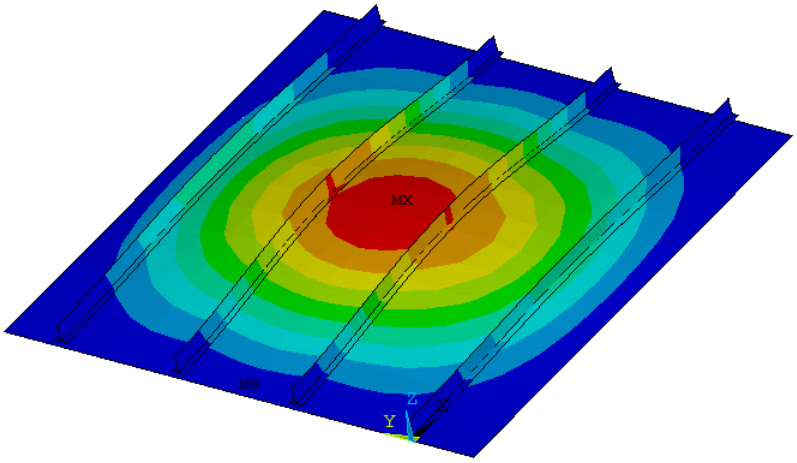


Figure 6-13: First-order buckling mode of the structure with optimized stacking sequence

For comparison, 20000 feasible solutions are generated randomly, where the largest critical load factors obtained is 1.805. In Figure 6-14, the optimization history is depicted, where the best objective value from random enumeration is also presented as reference.

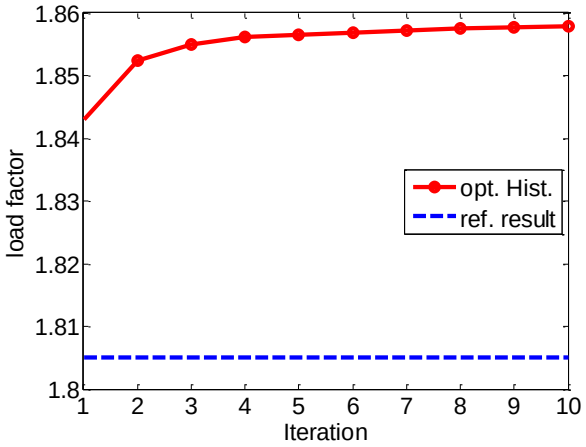


Figure 6-14: Optimization history

It could be seen that, the procedure is able to find an extremely high quality initial solutions while the number of function calls to generate the initial solution in this problem is just 621. In the following table, the relevant optimization result is summarized.

Table 6-13: Critical load factors results

	Load factor	Number of function calls
Average random	1.721	----
Best random	1.805	20000
Initial solution	1.843	621
Found solution	1.858	30548

The histogram of the critical load factors of random solutions are depicted in the following histogram. It shows that the distribution of the load factor is bell-shaped and it is very difficult to obtain a good solution with random enumeration.

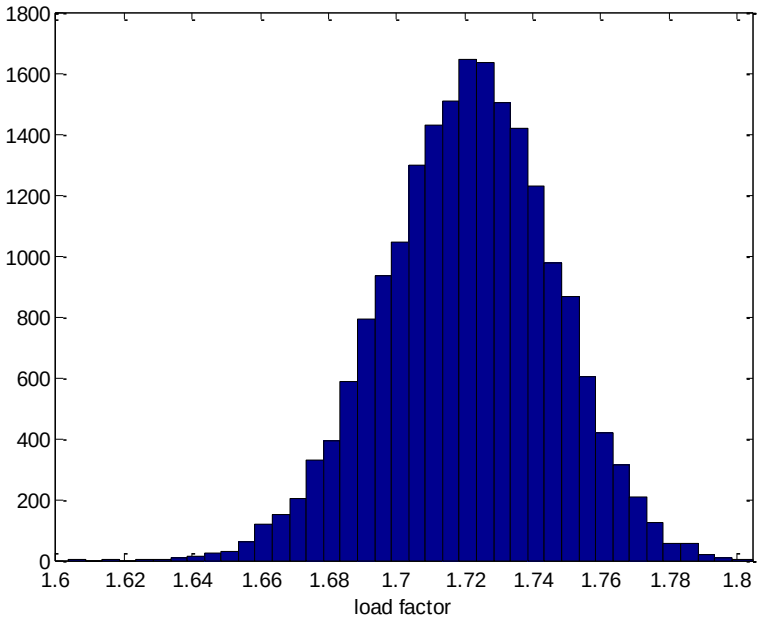


Figure 6-15: Distribution of critical load factors for buckling case

In the second case, the critical load factors with respect to critical buckling, stress failure criteria and strain failure criteria are considered simultaneously. The structure is under two load cases. The first load case is the same as the above example. And in the second load case, a pressure of one standard atmosphere is applied on top surface of the simply supported panel. Stress failure load factor λ_{stress} and strain failure load factor λ_{strain} are calculated according to Section 6.2.2 and 6.2.3 based on the results from static finite element analysis. Corresponding strain and stress limit of the material employed are stated in Table 6-14.

Table 6-14: Relevant strain and stress limits

strain	$\varepsilon_{\mathrm{LU}} = 0.008,\ \varepsilon_{\mathrm{TU}} = 0.029,\ \gamma_{\mathrm{LTU}} = 0.015$
Stress	$\sigma_{\mathrm{Lt}} = 1154\mathrm{Mpa},\ \sigma_{\mathrm{Lc}} = 1154\mathrm{Mpa},\ \sigma_{\mathrm{Tt}} = 51.2\mathrm{Mpa}, \sigma_{\mathrm{Tc}} = 210\mathrm{Mpa},\ \tau_s = 96.5Mpa$

The critical load factor λ of the problem which is to be maximized is defined as follows,

$$\lambda = \min\{\lambda_c, \lambda_\varepsilon, \lambda_\sigma\} \tag{6.34}$$

The optimization procedure terminates after 7 iterations, with 21449 number of function evaluations. The critical load factor of the solution is 1.833. The 20000 randomly generated feasible solutions are evaluated where the largest load factor obtained is 1.793. In the following figure, the optimization history is depicted.

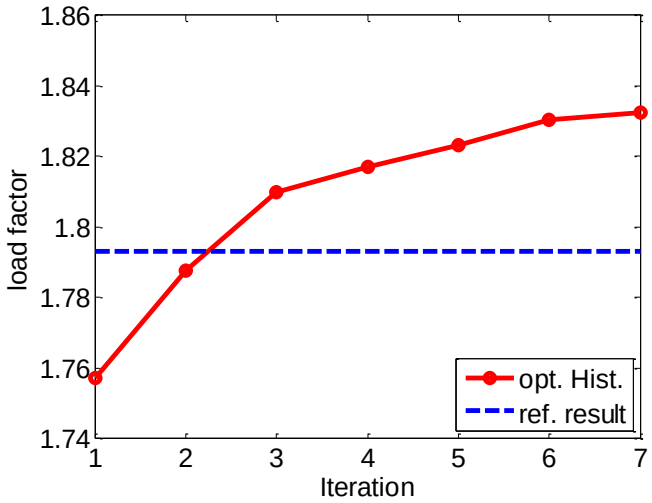


Figure 6-16: Optimization history

In Table 6-15, the relevant optimization result is summarized.

Table 6-15: Critical load factors results

	Load factor	Number of function calls
Average random	1.712	----
Best random	1.793	20000
Initial solution	1.757	621
After Iteration 2	1.810	6687
Found solution	1.833	21449

It could be seen that, the algorithm is also able to find a very high quality initial solution. After two iterations, the algorithm is able to find a solution that better than all the random ones.

6.6.2 Hybrid-stiffness laminated fuselage panels with window cutouts

6.6.2.1 Introduction to hybrid-stiffness laminated fuselage panels with window cutouts

Hybrid-stiffness laminates consists of both straight and curvilinear fiber layers. As the name stated, the composite fibers direction in a curvilinear fiber layers varies with respect to their spatial position. The advantage of hybrid-stiffness laminates has been investigated in recent years. It has been demonstrated that it is capable to increase both the structure's buckling load and tensile strength by applying the hybrid-stiffness laminated in plate with cut-outs. The application of this concept in the design of aircraft structures has been carried out at the Institute of Lightweight Structures of TU München [Ungwattanapanit 2012, 2013].

Figure 6-17 depicts the geometry of a hybrid-stiffness laminated fuselage panels with window cutouts that is being developed. The panel is composed of four segments: the plate, the outer flange which is overlapped with the plate, the vertical flange and the inner flange. All the four segments are manufactured with carbon fiber reinforced polymers (CFRP), among which the outer flange and the plate contains curvilinear fiber layers to improve the performance of the structure.

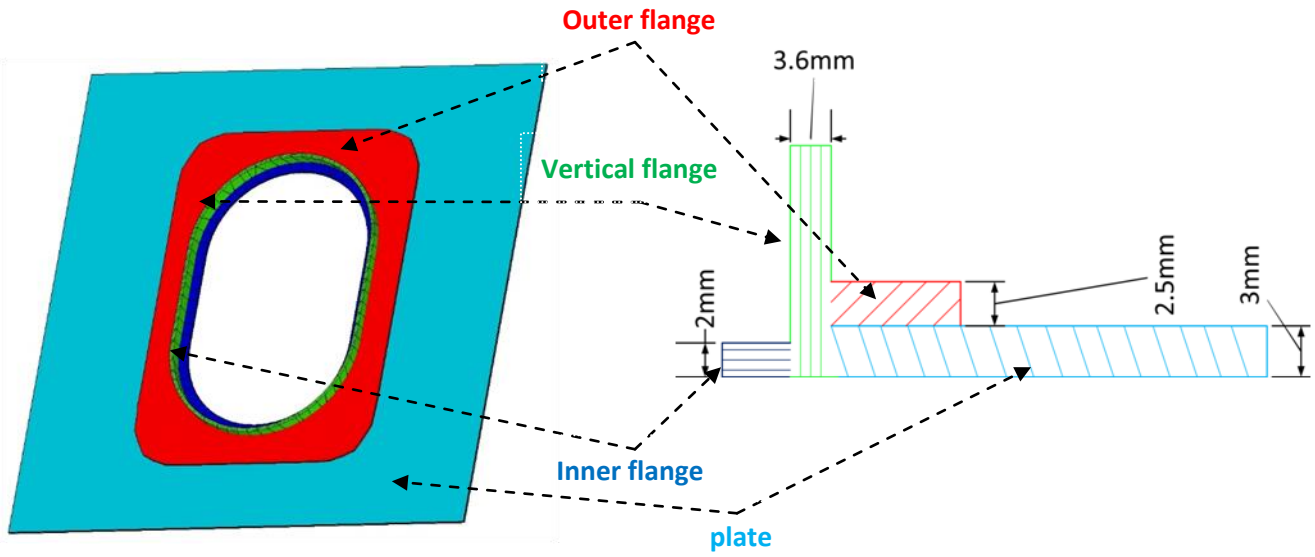
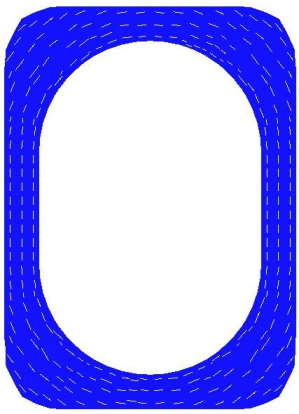
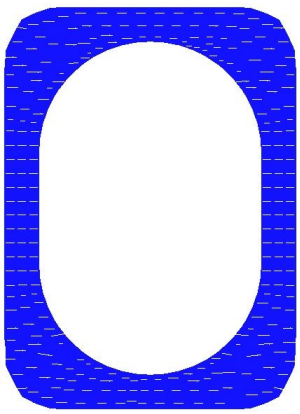


Figure 6-17: A fuselage panel with window cutouts to be reinforced

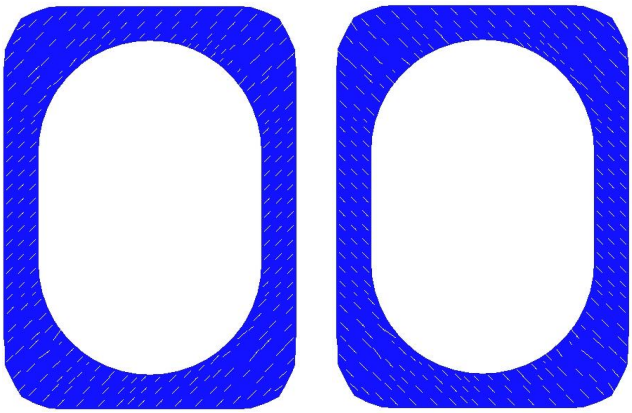
The compositions of the outer flange layers and plate layers have both been optimized. The outer flange contains eight circular layers in which the fibre direction is parallel to the outline of the window cut-out, four $0°$-layers, four pair of $\pm 45°$ layers as depicted in Figure 6-18. The stacking sequence is required to be symmetric.



(a) Circular layer ×8



(b) $0°$ layer ×4



(c) $\pm 45°$ layer ×4

Figure 6-18: Composition of layers for outer flange

The plate is composed of linearly varied fibre layers, where the fibre direction varies along y-axis according to Eq.(4.35). The layer is denoted $\left\langle \theta(0) \middle| \theta\left(\frac{b}{2}\right) \right\rangle$ to show its angle at y=0 and y=b/2. An illustration of the layer is depicted in Figure 6-19.

$$\theta(y) = \theta(0) + \frac{2|y|}{b} \cdot \left[\theta\left(\frac{b}{2}\right) - \theta(0)\right] \tag{6.35}$$

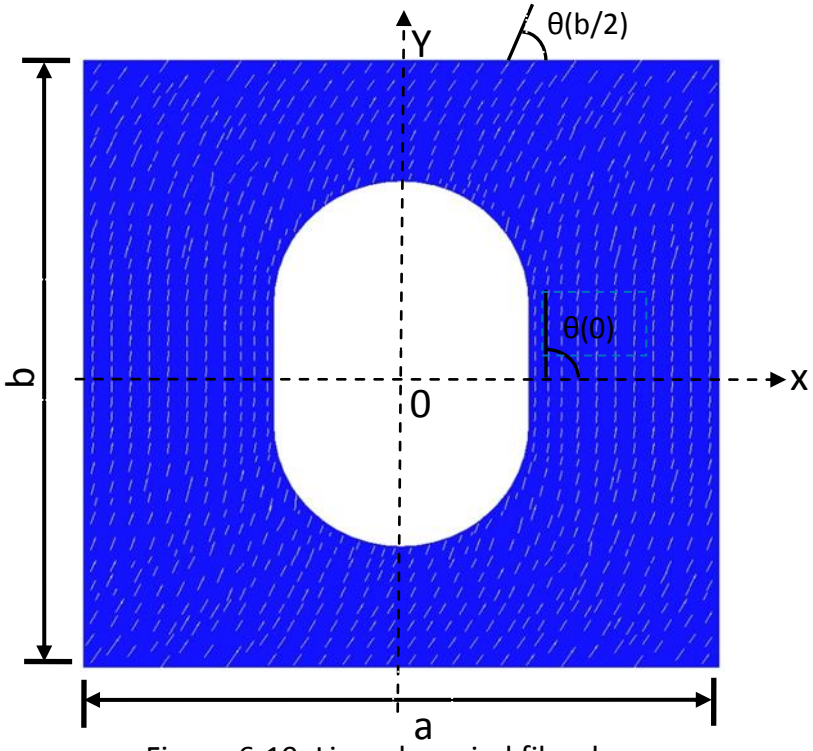
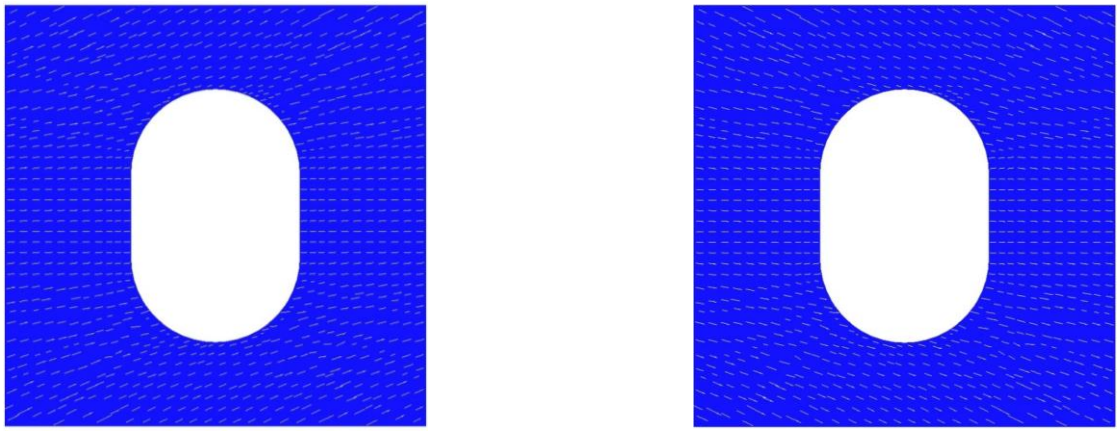
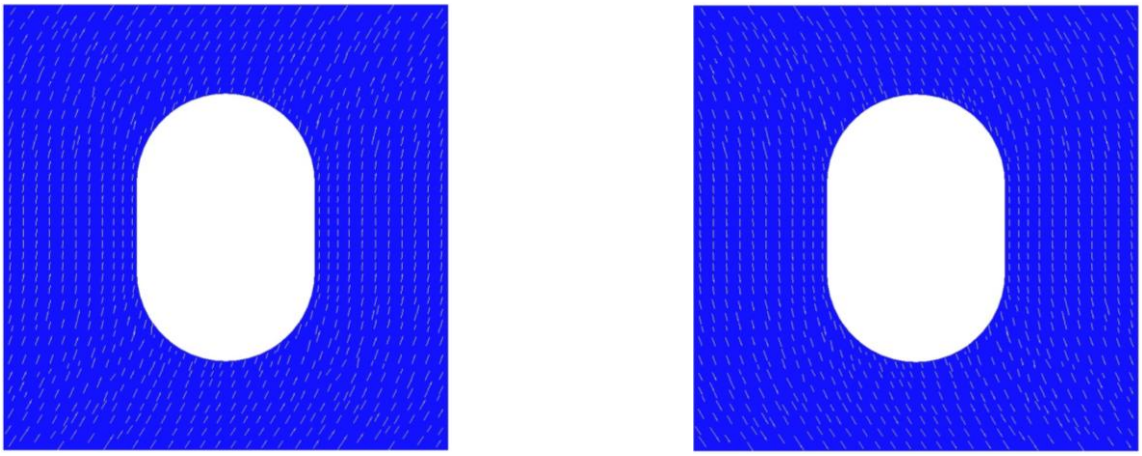


Figure 6-19: Linearly varied fiber layer

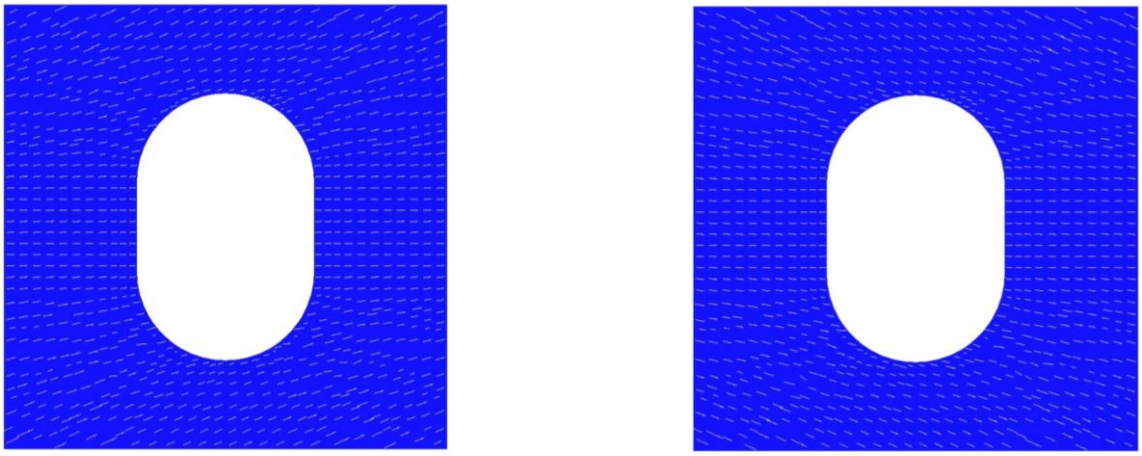
The plate is composed of twelve pairs of linearly varied fibre layers as shown in Figure 6-20.



(a) $\pm\langle 1.3°|25.0°\rangle$ ×4



(b) $\pm\langle 88.2°|56.3°\rangle$ ×4



(d) $\pm\langle 1.4°|26.0°\rangle$ ×4

Figure 6-20: Composition of plate layers

The thickness of each layer is 0.125mm, and the material is CFRP IM7/E8550. The stacking sequence of the structure is symmetric and the positive and negative angle layers are arranged consecutively. The total number of possible stacking sequence combination is $\binom{8}{4} \cdot \binom{4}{2} \cdot \binom{6}{2} \cdot \binom{4}{2} = 37800$.

The finite element analysis is carried out through Msc.Nastran, which doesn't accept interpolation of **A**, **B**, **D** matrix. Therefore, the initial solution generation method presented in Section 6.4.1.2 is employed here.

6.6.2.2 Stacking sequence optimization with respect to compression buckling

Firstly, the structure is under a compressive load of F=72822.5N as depicted in Figure 6-21. The compression critical buckling load factor is to be maximized.

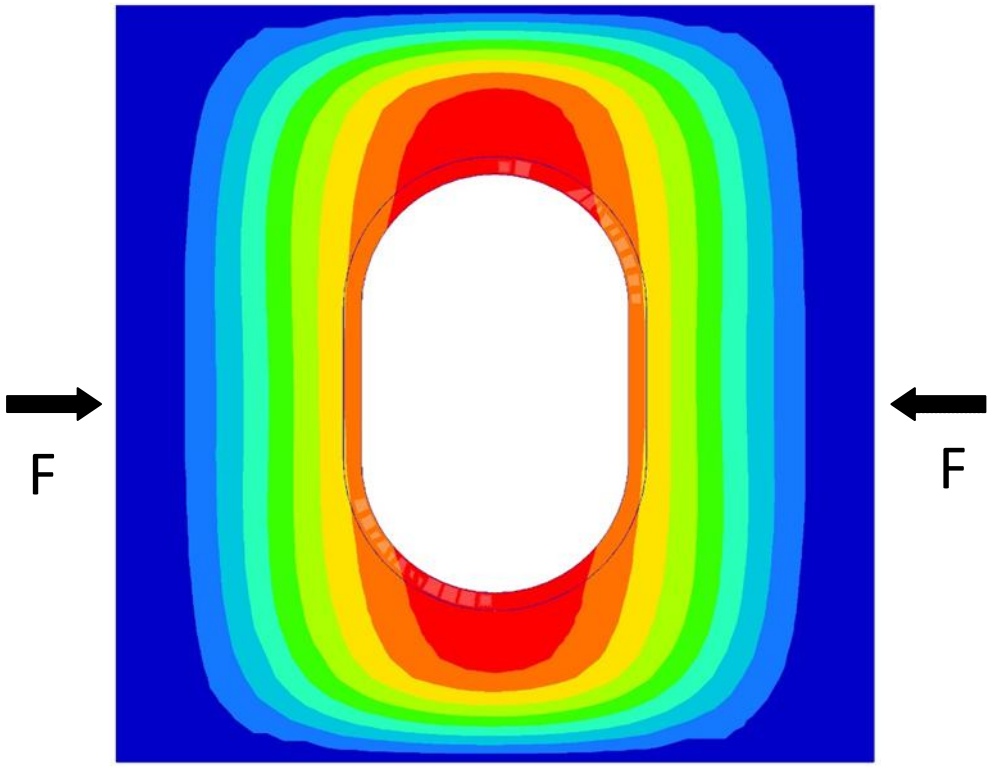


Figure 6-21: Compressive load and buckling mode

The optimization result is listed in Table 6-16. All the feasible solutions are evaluated and their average is listed in the table as reference. The optimized stacking sequence presented in Table 6-17.

Table 6-16: Stacking sequence optimization result with compression buckling

numFuncEva	Initial_obj.	Optimized obj.	Average	Global
175	1.01	1.07	0.991	1.07

Table 6-17: Optimum stacking sequence

	Outer flange	Plate
global	$[(\pm 45)_2/(\text{Circular})_4/0_2]_S$	$[(\pm\langle 88.2°\|56.3°\rangle/\pm\langle 1.4°\|26.0°\rangle)_2/(\pm\langle 1.3°\|25.0°\rangle)_2]_S$
Found	$[(\pm 45)_2/(\text{Circular})_4/0_2]_S$	$[(\pm\langle 88.2°\|56.3°\rangle/\pm\langle 1.4°\|26.0°\rangle)_2/(\pm\langle 1.3°\|25.0°\rangle)_2]_S$

In Figure 4-21, the histogram of the objective of all possible stacking sequences is presented. Different to a single bell-shaped distribution as in Figure 4-15, it is composed of many separated smaller bell-shaped histograms.

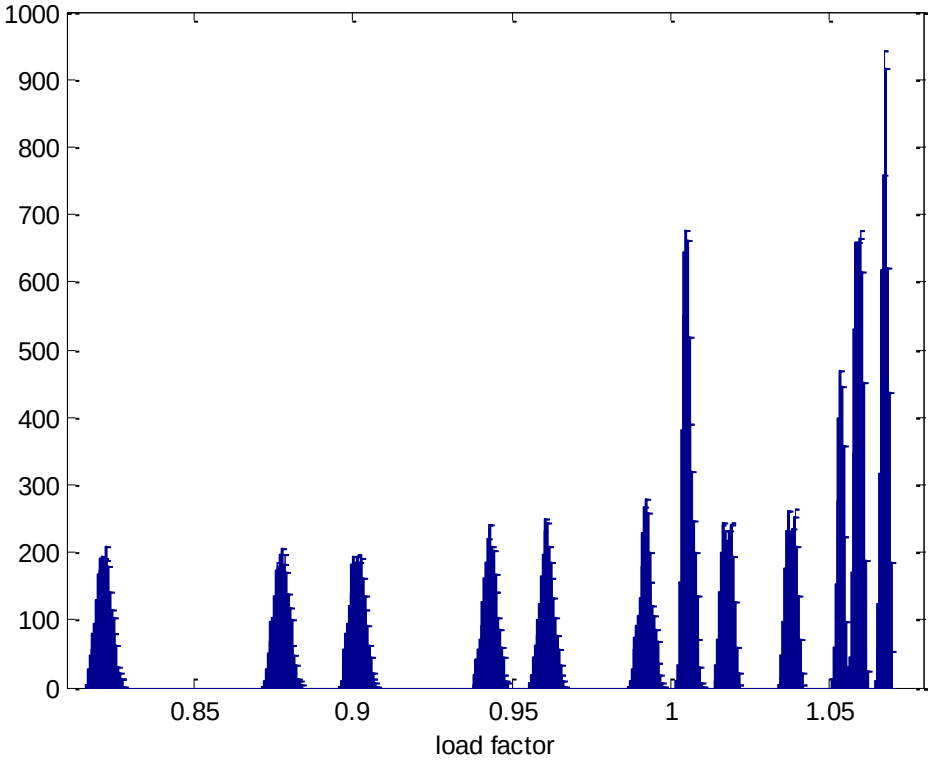
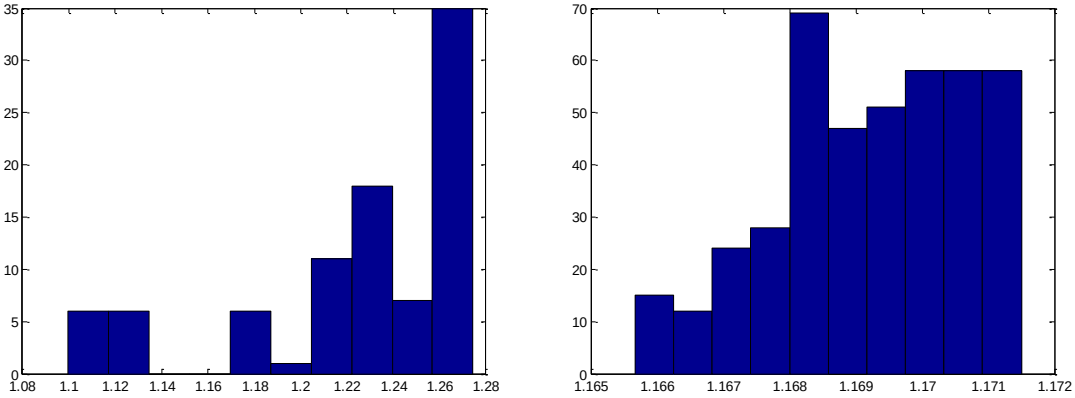


Figure 6-22: Distribution of load factor

Further looking into the distribution of the result, it turns out that this phenomenon is caused by the hybrid-stiffness layers: fixing the stacking sequence of the plate, the histogram of the load factor with changing the sequence of the outer flange is continuous. And the other way around, the histogram of the load factor with changing only the sequence of the plate is still separated.



(a) Fix flange, change plate (b) Fix plate, change flange

Figure 6-23: Histogram of load factors when the stacking sequence of only one component changes

The range of load factor of changing the stacking sequence of the plate is much larger than that when change only the flange. This fact demonstrates that the stacking sequence of the structure, especially the components that contain hybrid-stiffness laminates, should be properly arranged in order to obtain a satisfying structural performance.

6.6.2.3 Stacking sequence optimization with respect to shear buckling

In this case, the structure is under a uniformly shear load as presented in Figure 6-24, and the shear critical buckling load factor is the objective to be maximized.

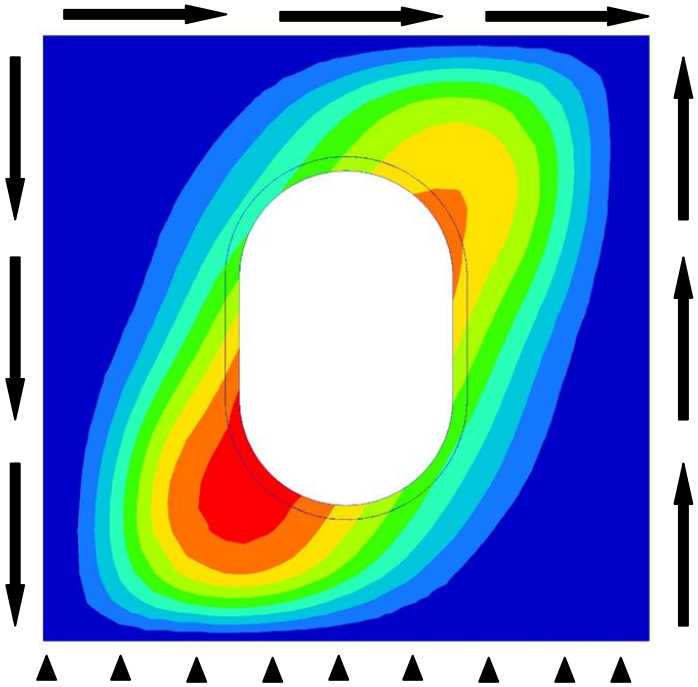


Figure 6-24: Shear load and buckling mode

The optimization result is listed in Table 6-18. The total combination of feasible stacking sequences is 37800. All the feasible solutions are evaluated and their average is listed in the table as reference. The optimized stacking sequence is presented in Table 6-19.

Table 6-18: Stacking sequence optimization result with shear buckling

numFuncEva	Initial_obj	Optimized obj.	Average	Global
131	1.271	1.273	1.224	1.274

Table 6-19: Optimum stacking sequence

	Outer flange	Plate
global	$[(\text{Circular})_4/0_2/(\pm45)_2]s$	$[(\pm\langle 1.4°\vert 26.0°\rangle)_2/(\pm\langle 88.2°\vert 56.3°\rangle)_2/(\pm\langle 1.3°\vert 25.0°\rangle)_2]s$
Found	$[(\text{Circular})_4/0_2/(\pm45)_2]s$	$[\pm\langle 1.4°\vert 26.0°\rangle/\pm\langle 88.2°\vert 56.3°\rangle/\pm\langle 1.4°\vert 26.0°\rangle/(\pm\langle 1.3°\vert 25.0°\rangle)_2/\pm\langle 88.2°\vert 56.3°\rangle]s$

7. Simultaneous material selection and size optimization in structural design

Optimal materials selection and materials combination are important topics in modern structural design, when more and more new types of material are developed and available. In this chapter, the optimal material selection is taken into account together with traditional size optimization. The problem is formulated into a mixed-integer nonlinear programming (MINLP) problem, the algorithm developed in section 4.2 is applied to find a good solution for problems with specific characteristics. A strategy to improve the quality of solutions is also presented. Several numerical examples are employed to demonstrate the procedure.

7.1 Mathematical formulation into MINLP

Assume there are M types of material available for a structure which is composed of N components. Each of the components is made of one type of material, and different components could use different kinds of material. Additionally, there is one continuous size parameters that needs to be optimized for each component. The objective and constraints are functions related to the structural performance, such as buckling load factor, stresses and deflections.

Number the M material by 1 to M, and N components by 1 to N. The geometric configuration of a structure is determined by vector $\mathbf{X}=[\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_M]$, where $\mathbf{x}_i$ is the size parameter for component i (i=1,2,…,N).

A binary matrix $\mathbf{B} = (b_{ij}) \in \{0,1\}_{N\times M}$ is employed to represent the selection of materials, where:

$$b_{ij} = \begin{cases} 1, & if\ material\ j\ is\ selected\ for\ component\ i \\ 0, & otherwiese \end{cases} \tag{7.1}$$

According to assumption that each component is made of one material, the row-wise assignment constraints should be satisfied:

$$\sum_{j=1}^{M} b_{ij} = 1 \ (i = 1,2, \dots, N) \tag{7.2}$$

Denote $\rho_j$the density of material j (j=1,2,….,M), then the density of component i could be interpolated by:

$$\rho(i) = \sum_{j=1}^{M} \rho_j b_{ij} \tag{7.3}$$

This interpolation scheme is applied also to other material properties, with which properties of each component are expressed as functions of $\mathbf{B}$.

Therefore, general structural performance and further constraints, which are fully determined by structural configuration and material properties, could be formulated implicitly as functions of $\mathbf{B}$ and $\mathbf{x}$:

$$g_k(\mathbf{B}, \mathbf{x}) \leq 0 \ (\mathrm{k} = 1,2, \dots) \tag{7.4}$$

Normally objective and constraints are highly nonlinear functions in a structural system. Thus, the simultaneous geometric and material optimization problem is formulated into a mixed-integer nonlinear programming problem.

7.2 Illustration with a single beam optimization problem

To illustrate the formulation procedure, a cantilever beam example as depicted in Figure 7-1 is employed. The cantilever beam, which is of circular cross-section with length L=1m, is under an axial load of F=2000N. The beam is to be light weighted with a constraint on stress that should not be larger than the maximum allowable stress.

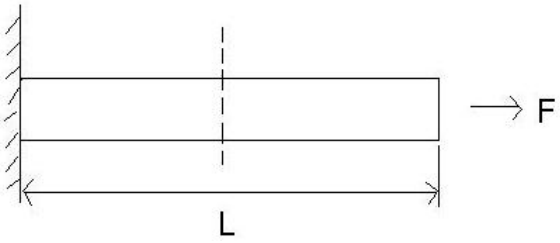


Figure 7-1: Cantilever beam under axial force

There are five kinds of material available for manufacturing the beam with their properties listed in Table 7-1.

Table 7-1: Available material and related properties for the beam example

	Aluminum	Steel	Titanium	Copper	Magnesium
Density ρ (10^3kg/m^3)	2.7	7.85	4.5	8.9	1.8
Maximum allowable stress σ_A (MPa)	143	235	539	294	210

The area of cross-section of the beam is the size variable to be optimized, and five binary variables $b_{11}\sim b_{15}$ represent the selection of the five materials. Then the density of the beam through the interpolation scheme is formulated as:

$$\rho = \sum_{j=1}^{5} \rho_j b_{1j} \tag{7.5}$$

The maximum allowable stress σ_A is interpolated as

$$\sigma_A = \sum_{j=1}^{5} \sigma_{A_j} b_{1j} \tag{7.6}$$

The weight of structure to be minimized is:

$$f = \rho x L = \sum_{j=1}^{5} \rho_j L x b_{1j} \tag{7.7}$$

With the row-wise assignment constraint:

$$\sum_{j=1}^{5} b_{1j} = 1 \tag{7.8}$$

The stress of beam is calculated by force divided by area of cross-section, and thus the constraint related to structure performance is:

$$\frac{F}{x} \leq \sigma_A \tag{7.9}$$

i.e.:

$$\frac{F}{x \cdot \sum_{j=1}^{5} \sigma_{A_j} b_{1j}} - 1 \leq 0 \tag{7.10}$$

Equation (7.7), (7.8) and (7.10) is the mathematical formulation of the problem. The solution procedure of this problem has been presented in section 4.4.

7.3 Demonstration with bar truss structures optimization through comparison of algorithms at hand

7.3.1 A small scale 3-bar truss example

In this section, performance of the algorithm is evaluated through a three bar truss example. The structure is presented in Figure 7-2, with two load cases listed in Table 7-2, lengths of bar 1 to bar 3 are equal to 1732mm, 1000mm, and 1000mm respectively.

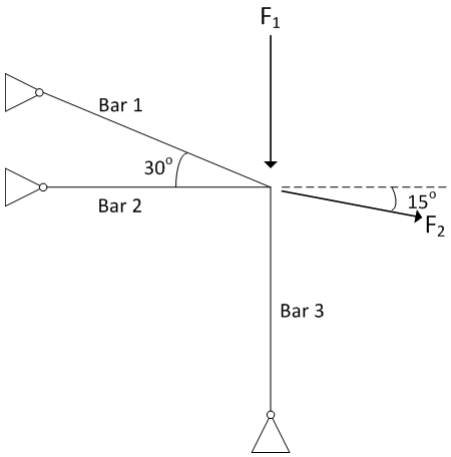


Figure 7-2: Three bar truss structure and loads

Table 7-2: Load cases of three bar truss example

Load case	Force (kN)	Direction
1	25	Vertical downwards

2	150	15^o to horizontal downward

Seven kinds of material are available with their properties listed in Table 7-3.

Table 7-3: Available material and related properties for three-bar truss example

	Al	Steel	Ti	Cu	Ni	Zn	Mg
Density ρ (10^3kg/m^3)	2.7	7.85	4.5	8.9	8.8	7.14	1.8
Young's modulus E(GPa)	70	210	102	119	207	70	45
Maximum allowable stress σ_A(MPa)	143	235	539	294	450	30	210

The size variables are area of cross-section of each bar in the range of 0.01 to 2000 mm^2. Under each load case, all three bars are under constraints of maximum stress and buckling conditions:

$$\frac{\sigma_i}{\sigma_{A_i}} - 1 \leq 0 \ (\mathrm{i} = 1,2,3) \tag{7.11}$$

$$\frac{\sigma_i}{\sigma_{cr_i}} - 1 \leq 0 \ (\mathrm{i} = 1,2,3) \tag{7.12}$$

, where the critical buckling stress is calculated as:

$$\sigma_{cr} = \frac{\pi EA}{L^2} \tag{7.13}$$

The mathematical formulation of this problem has 21 binary variables, 3 bounded continuous variables, 3 linear equality constraints and 12 inequality constraints. The global optimal of the problem could be obtained by enumerate all possible binary combinations, which is total 7^3=343 under assignment constraints. The best material for bar 1 to bar 3 is [Mg, Ti, Mg], area of cross-section is [153.8, 222.5, 1175.6] mm^2 respectively, with objective equals to 3.437kg. The results of developed algorithm and state of the art algorithms are compared in Table 7-4.

Table 7-4: Results of different algorithms for three bar truss problem

Algorithm	obj (kg)	Error	num_{func}
Constr.-directed	3.437	0.00%	325
MISQP*	3.89	13.12%	252
Branch&Cut*	3.450	0.38%	4063
Branch&Bound*	3.437	0.00%	14884

*Since the performance of algorithms relies on the choice of the initial solution, average 343 results where initial binary variables enumerate all possible combinations is presented.

It could be seen that the developed algorithm is significantly outperform the branch and bound and branch and cut algorithm from both quality of solution and number of function evaluations. The computational cost is larger but within the same magnitude as the MISQP algorithm. However, the quality of solutions is largely improved. Since the quality of the solution is also an important consideration, therefore, in the following the performance of MISQP algorithm is omitted. It should also be noted that, the branch and bound algorithm requires much larger computational efforts than the other three algorithms.

7.3.2 A medium scale 25-bar truss examples

The same 25-bar truss structure as in Figure 5-4 is taken as examples here. All of the bars are divided into eight groups, and bars of each group are made of the same material and with the same area of cross-section. The problem is to select proper material and cross-section area of each group to minimize the weight of the structure under stress and displacement constraints. Five kinds of material (Al, Steel, Ti, Cu, Mg) as stated in Table 7-3 are available. The total number of binary variables is 40, together with 8 continuous variables. The total combination of binary variables under assignment constraints is $5^8 \approx 3.9 \times 10^5$.

Two tests are carried out with this example, where in test 1 only a single load case is considered. In test 2, two load cases are applied on the structure respectively. The loading conditions are listed in Table 7-5.

Table 7-5: Loading conditions for 25-bar truss example

Test	Load case	Node	Fx/N	Fy/N	Fz/N
1	1	1	4448	44482	44482
		2	0	44482	44482
		3	2224	0	0
		6	2669	0	0
2	2	1	0	88964	-22241
		2	0	88964	22241
	3	1	4448	44482	-22241
		2	0	44482	-22241
		3	2224	0	0
		6	2224	0	0

In the first test example, the structure is under single load case which applied forces on node 1,2,3,6 at the same time. The displacement of each degree of freedom is limited to 8.89mm, and compression and tension stresses of each bar are constrained with respect to buckling and maximum allowable stress. There are 80 nonlinear constraints and 8 linear equality constraints in all. The global

optimal obtained by enumerating all feasible binary combinations is listed in Table 7-6, with minimum weight of 196.2kg.

Table 7-6: Global optimal of test 1

Group	Material	Area of cross section (mm^2)
1	Mg	11.72
2	Mg	37.88
3	Steel	770.99
4	Mg	0.01
5	Steel	422.36
6	Steel	164.64
7	Steel	33.90
8	Steel	829.64

The developed algorithm stops after 1801 function evaluations, the result is listed in Table 7-7, with the objective of 210.8kg, which is larger than that of global optimal by 7.4%.

Table 7-7: Optimal found by developed algorithm of test 1

Group	Material	Area of cross section (mm^2)
1	Steel	0.01
2	Steel	9.44
3	Ti	1503
4	Mg	0.01
5	Steel	436.7
6	Steel	169.6
7	Mg	149.2
8	Steel	856.2

Results of different algorithms are compared in Table 7-8.

Table 7-8: Results of different algorithms for test 1 of 25-bar truss problem

Algorithm	obj (kg)	error(%)	num_{func}
Constr.-directed	210.8	7.4	1801
Branch&Cut*	223.9	14.2	3408
Branch&Bound*	no comparable returned in 50000 function calls		

*Since the performance of algorithms relies on the choice of the initial solution, average results where initial binary variables enumerate all possible combinations is presented.

In the second example, the problem is the same with above example except that two load cases needs to be considered. The total number of constraints therefore increases to 160 nonlinear inequality constraints with 8 linear equality constraints.

The global optimal obtained by enumerating all feasible binary combinations is listed in Table 7-9, with minimum weight of 230.05kg.

Table 7-9: Global optimum of test 2

Group	Material	Area of cross section (mm^2)
1	Mg	0.01
2	Steel	432.6
3	Steel	635.5
4	Mg	0.01
5	Mg	0.01
6	Steel	144.9
7	Steel	344.1
8	Steel	566.0

The developed algorithm stops after 1235 function evaluations, the result is listed in Table 7-10, with the objective of 230.05kg, which is only slightly larger than that of global optimum.

Table 7-10: Optimum found by developed algorithm of test 2

Group	Material	Area of cross section (mm^2)
1	Al	0.01
2	Steel	432.6
3	Steel	635.6
4	Steel	0.01
5	Steel	0.01
6	Steel	144.9
7	Steel	344.0
8	Steel	566.0

Results of different algorithms are compared in Table 7-11.

Table 7-11: Results of different algorithms for test 2 of 25-bar truss problem

Algorithm	obj (kg)	error(%)	num_{func}
Constr.-directed	230.05	0.00	1705
Branch&Cut*	232.76	1.18	9032
Branch&Bound*	no comparable returned in 50000 function calls		

*Since the performance of algorithms relies on the choice of the initial solution, average results where initial binary variables enumerate all possible combinations is presented.

From the above two examples, it could be seen that the Branch-and-bound algorithm is not proper in solving the material selection problem due to its large and significantly increasing computational efforts with respect to the growth of problem size. The Branch-and-cut algorithm is comparable but underperforms the developed constraint-directed algorithm in all of the cases.

7.3.3 A large scale 72-bar truss example

Finally the procedure is demonstrated with a large scale 72-bar truss structure. All the bars are divided into 16 groups as depicted in different color in Figure 7-3. It is constructed by overlaying the same structure four times from the bottom to top. The composition of each group is illustrated in Figure 7-4 with a single layer of the structure.

The structure is under a uniform thermal load with a temperature change of 40℃. Besides that, three load cases as in Table 7-13 is applied separately, where stress and deflection constraints as discussed in Section 7.3.2 are applied. Five materials listed in Table 7-12 are available. There are 80 binary variables, 16 continuous variables, 16 linear equality constraints, 612 nonlinear inequality constraints in all. The total number of feasible binary combinations is $7^{16}\approx 3.32\times 10^{13}$.

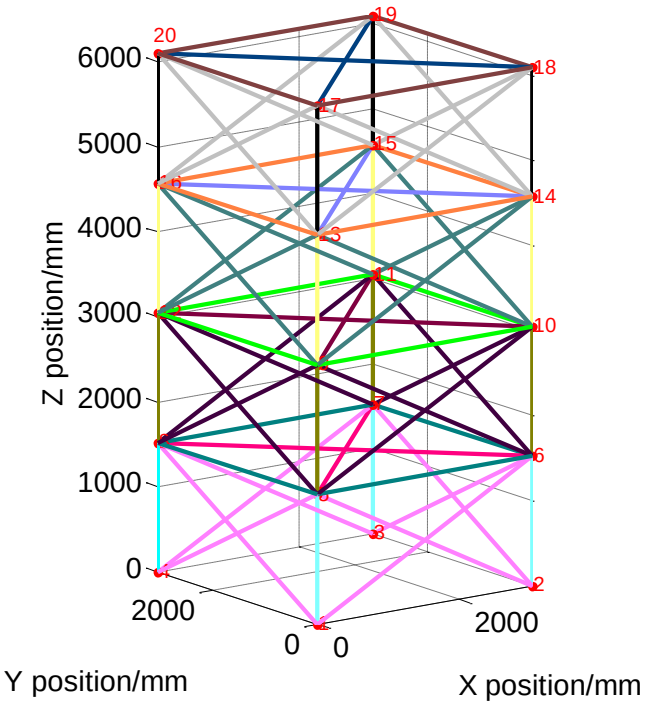


Figure 7-3: 72-bar truss structure

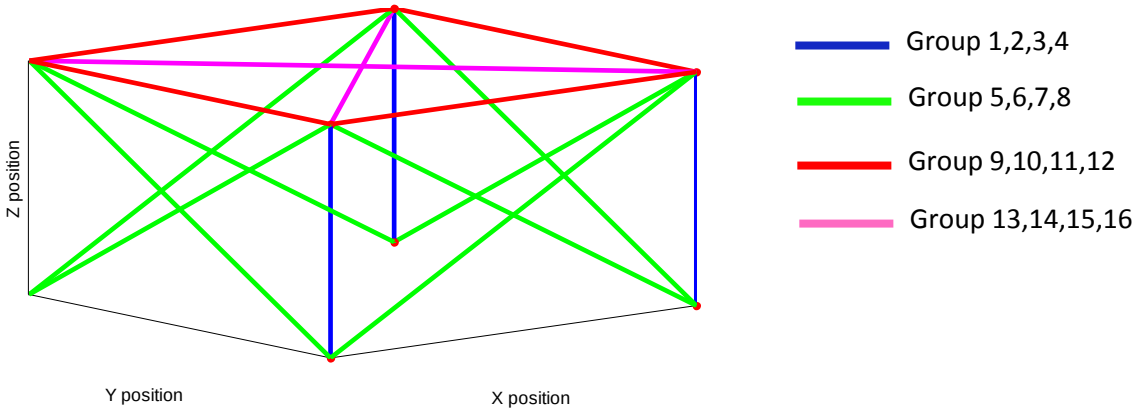


Figure 7-4: Single layer of the 72-bar structure with group illustration

Table 7-12: Available material and related properties for 72-bar truss

	Aluminum	Steel	titanium	Copper	Magnesium
Density (10^3kg/m^3)	2.768	7.85	4.5	8.9	1.8
Allowable stress (MPa)	143	216	539	294	210
E (GPa)	68.95	210	102	119	45
CTE (10^{-6}/℃)	23.6	11.8	8.6	17	26

Table 7-13: Loading conditions for 72-bar truss example

Load case	Node	Fx/N	Fy/N	Fz/N
1	17	22241	22241	-22241
2	17	0	0	-22241
	18	0	0	-22241
	19	0	0	-22241
	20	0	0	-22241
3	10	-22241	-22241	0
	12	22241	22241	0

Since the total number of combinations of material is innumerous, there is no way to know the global optimal for this problem. Instead, 1000 random combinations of material selection are generated, with which pure continuous nonlinear optimization with respect to the size variables are carried out with SQP algorithm of MATLAB optimization toolbox. The distribution of the objective of optimums for these problems is depicted in Figure 7-5. The objective of the best result achieved is 119.3kg.

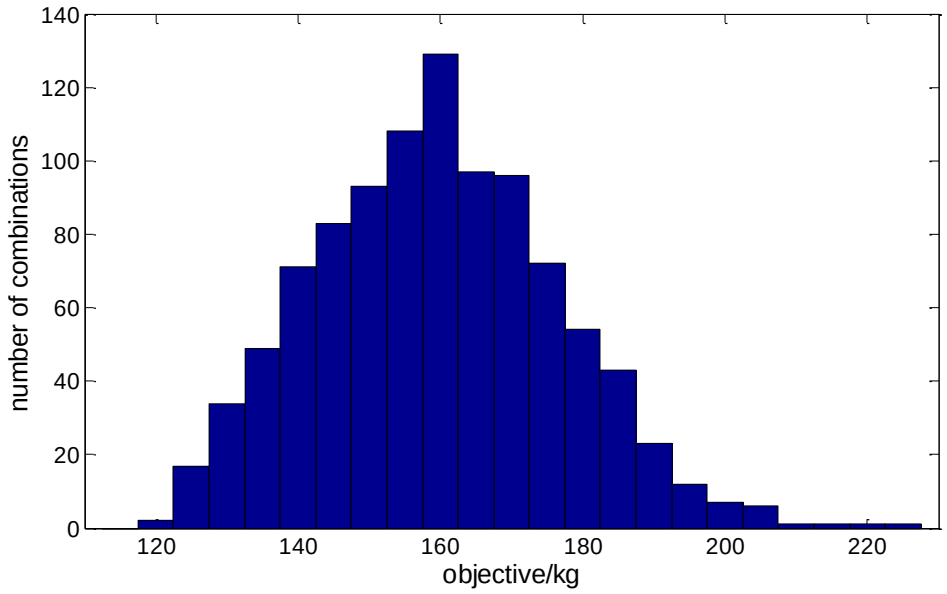


Figure 7-5: 72-bar truss structure and loads

The developed algorithm solves this problem in total 940 function evaluations with the objective of 147.3kg which is larger than the best result by 23.4%. The reason that the algorithm behaves not ideally in this case is investigated in the following and a method to cure this shortcoming is suggested.

7.4 Enforced neighbor-search strategy

7.4.1 Procedure with enforced neighbor-search strategy

Looking into the optimization procedure of the 72 bar truss example, it shows that the algorithm stops after only 2 iterations due to the same material selection in successive iterations is detected. The reason for this is that with the increasing of number of constraints, it is difficult to improvement the design by adjusting the material selection and the size of only one component. Therefore, the possibility of wrongly rejecting a better neighbor material selection (i.e. a combination of material selection that differs to the current selection in one component) increases.

To overcome this problem, an enforced neighbor-search strategy is developed and embedded into the optimization procedure as depicted in red in Figure 7-6.

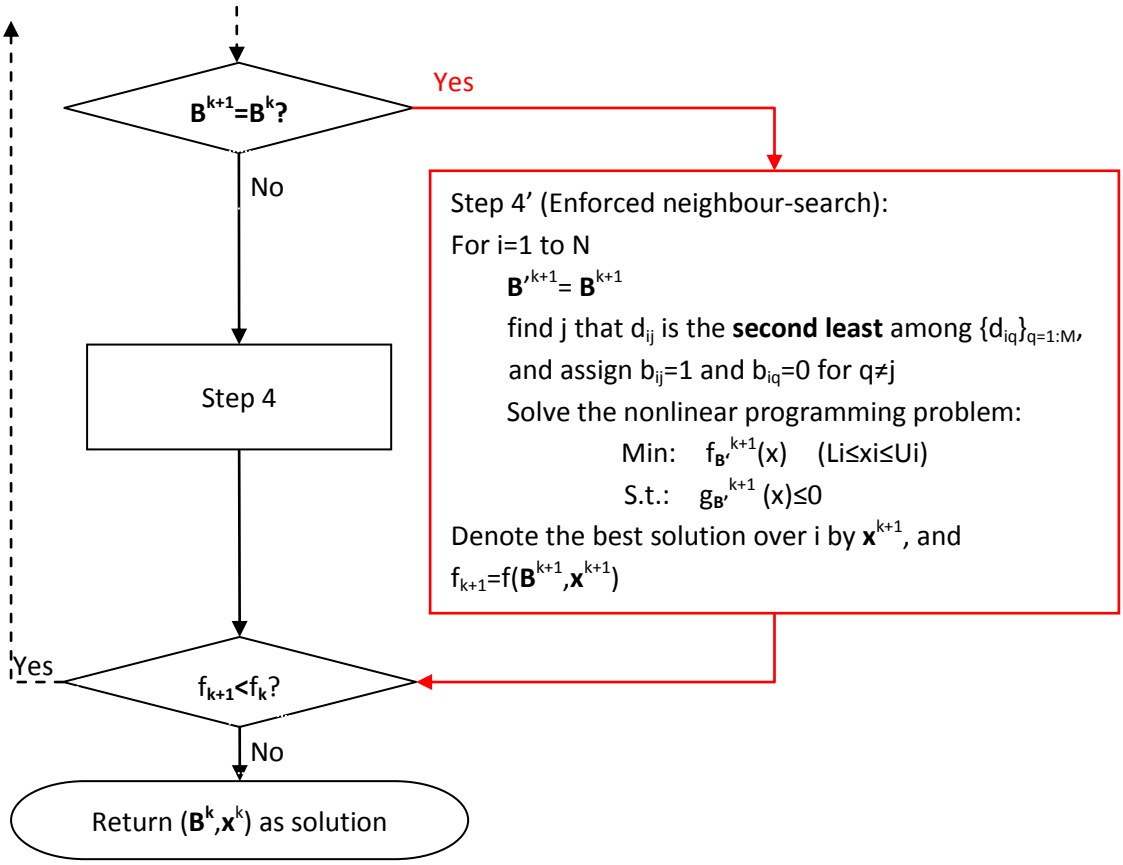


Figure 7-6: Optimization procedure with enforced neighbour-search strategy

It should be noted that, when $\mathbf{B}^{k+1}=\mathbf{B}^{k}$, it must be the case that all $\{d_{ij}\}$ are nonnegative. And for the pair (i,j) that $b_{ij}=1$, it must be $d_{ij}=0$. Therefore the smallest row-wise d_{ij} corresponds to the current material selection, and the best neighbor selection must have the second smallest value of $\{d_{ij}\}$ in a same row.

This strategy enforces the search for a better solution in a neighbor material selection for each component, which will increase the quality of found solutions. On the other hand, however, it will increase the computational costs greatly. During the enforced neighbor-search procedure, N nonlinear optimization problems needs to be solved. Comparison to the Step 4 where only one nonlinear optimization problem is to be solved, the number of function evaluations could be estimated as roughly N times as before. At the same time, the number of function calls will grow also due to the increase number of iterations.

7.4.2 Demonstration with the 72-bar truss example

The large-scale 72-bar truss problem is again employed here for the demonstration of the improved procedure. Now, the procedure stops after 5 iterations with total function evaluations of 26571. The objective of the solution found is 114.2kg, which is smaller than all the 1000 optimization solutions with random generated material selections. The material and the value of size variables of the solution are listed in Table 7-14.

Table 7-14: Optimal found by developed algorithm

Group	Material	Area of cross-section /mm^2	Group	Material	Area of cross-section /mm^2
1	Steel	280.7	9	Al	0.01
2	Steel	188.0	10	Al	112.9
3	Al	221.4	11	Ti	21.4
4	Al	269.2	12	Al	109.1
5	Al	260.1	13	Copper	0.01
6	Al	233.5	14	Al	0.01
7	Al	227.0	15	Copper	0.01
8	Steel	75.3	16	Al	181.0

The bars with area of cross-section equals to the lower bound of 0.01mm^2 are seen that these bars could be removed from the structure, and the total number of these kinds of bars is 10. By using different colors to represent the material

selection, the optimized structure is depicted in Figure 7-7, which contains 62 bars in all.

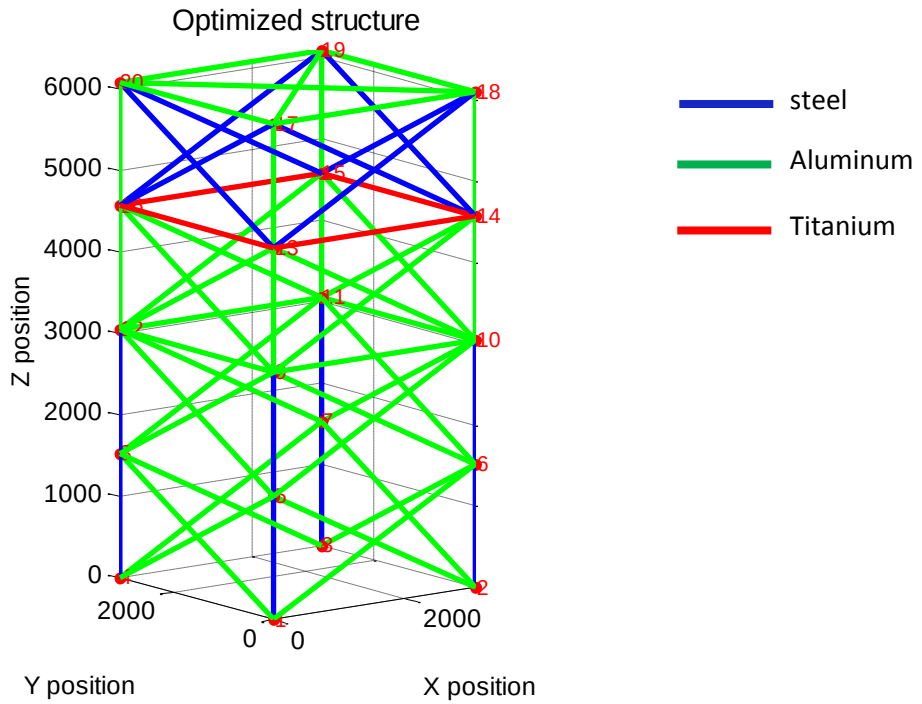


Figure 7-7: Optimization structure

This example demonstrates that with the enforced neighbour-search strategy, the quality of the solution is greatly improved. The computational cost could be estimated by N times the algorithm without this strategy. This estimation could be used to decide whether the strategy is embedded into the procedure in practice when the quality of solution and computational cost need to be compromised.

7.5 Application in the design of a panel with stiffeners

In this section, optimization the design of a panel structure with stiffeners is presented.

7.5.1 Model description

The panel structure is shown in the following figure. It contains two parts, the stiffener and the panel. The cross-section of the stiffener is depicted as below.

The thickness t is the continuous parameter that determines the geometry of the stiffeners as in Eq. 7.14.

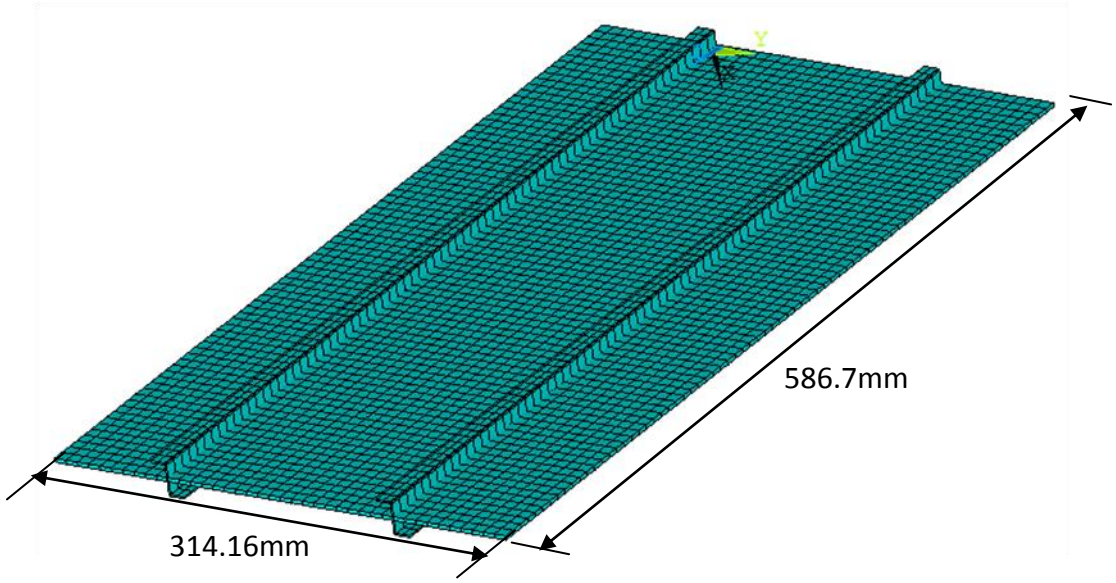


Figure 7-8: Panel structure with stiffeners

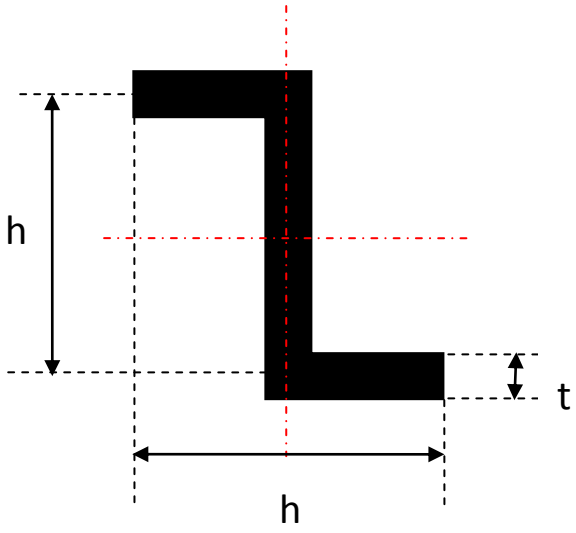


Figure 7-9: Cross-section of the stiffeners

$$h = 10t + 10 \tag{7.14}$$

Three load cases are under consideration, which are taken from typical working conditions of a fuselage panel. In the first load case, the panel is under a pure compression force 33742N, which is uniformly distributed on edges and along the

direction of the stiffener. The loading condition with the buckling mode is depicted in Figure 7-10.

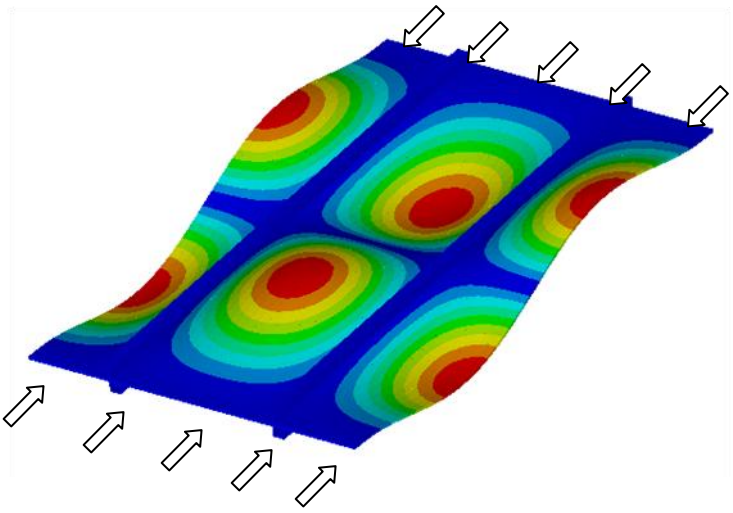


Figure 7-10: Load case 1: buckling under compression

In the second load case, the panel is under both a compression force 24994N and a shear force 12720N. They are uniformly distributed on edges. The loading condition with the buckling mode is depicted in Figure 7-11.

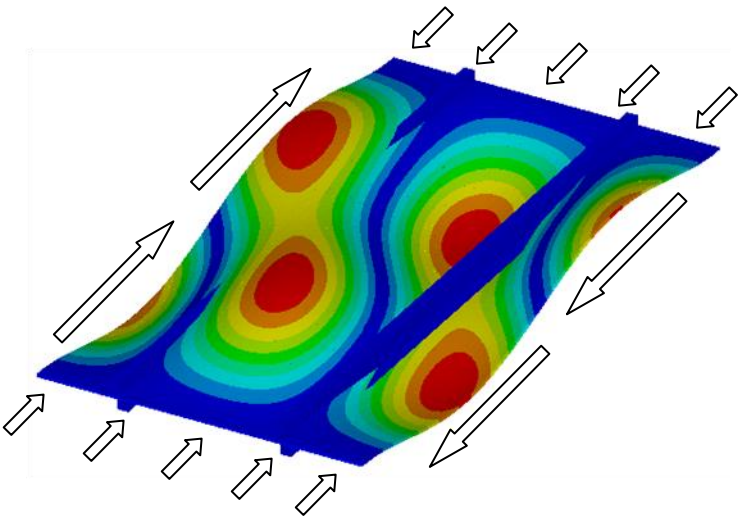
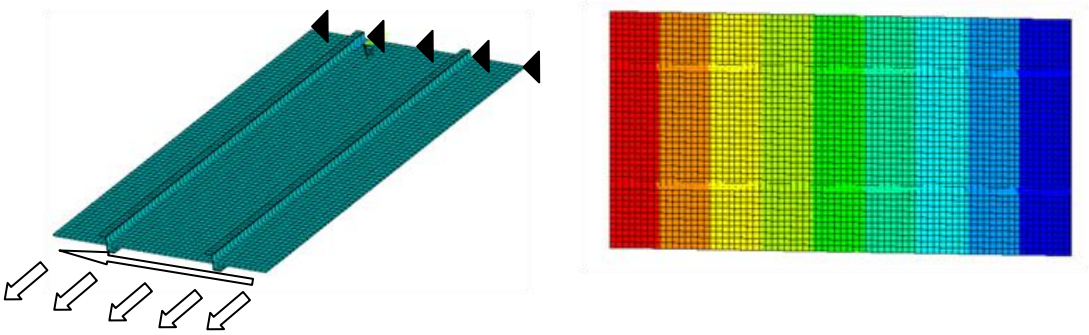


Figure 7-11: Load case 2: buckling under compression and shear forces

The third load case is a static case. The structure is under both an extensional force 60650N and a shear force 7500N on one edge.



(a) extension and shear forces (b) contour of in plane displacement

Figure 7-12: Load case 3: static analysis under extension and shear forces

7.5.2 Optimization problem and its solution

There are four kinds of aluminium alloys are available for the manufacturing of stiffeners and the panel. Some of the properties of these materials are listed in Table 7-15 [ASM 2014].

Table 7-15: Material properties of four kinds of aluminum alloys

	Al-2024 T3	Al-2014-T6	Al-6061-T6	Al-7075-T6
Density (kg/m^3)	2780	2800	2700	2810
Young's modulus (GPa)	73.1	73.1	68.9	71.7
Poisson's ratio	0.33	0.33	0.33	0.33
Fatigue Strength (MPa)	138	124	96.5	159
Cost (Euro/kg)	2.016	2.010	1.836	1.893

The design variables of optimization problem are the material selections of the stiffeners and the panel from these four materials, the height of the stiffeners and the thickness of the panel. The objective is to minimize the manufacturing cost of the structure. There are four constraints in all: the lower bounds of the buckling load factor in load case 1 and load case 2, the upper bounds of the maximum von Mises stress of both the stiffeners and the panel with respect to fatigue strength under a safety factor of 1.1.

Applying the developed algorithm, an optimum is found after 2 iterations and with 62 function evaluations. The solution shows the panel is manufactured with Al-6061 and the stiffeners are made of Al-7075. The corresponding optimal

thickness of the panel is 2.63mm and the height of the stiffener is 20.90mm. The total cost of the structure is 2.67Euro.

To verify the quality of the solution, continuous optimizations under all 16 material combinations are carried out. The optimal objective values are listed in Table 7-16. It demonstrates that the algorithm successfully found the global optimum in this case.

Table 7-16: Optimal objective value (Cost in Euro) of different material combinations

		Panel			
		Al-2024 T3	Al-2014-T6	Al-6061-T6	Al-7075-T6
Stiffener	Al-2024 T3	2.95	2.96	2.70	2.83
	Al-2014-T6	3.18	3.20	3.00	3.09
	Al-6061-T6	3.83	3.85	3.61	3.71
	Al-7075-T6	2.93	2.95	**2.67**	2.81

8. Conclusion and outlook

In this thesis, an accelerated neighbor-search algorithm and a constraint-directed algorithm are designed for pure integer nonlinear assignment problem and the mixed-integer nonlinear programming with binary assignment problem respectively. Initial solution generation procedures for both problems are also developed, with which high quality of starting points for further optimization algorithm can be generated.

The performance of these procedures are evaluated from the point of view of both the solution quality and number of function evaluations which dominates the computational cost in structural optimization with finite element analysis. Mathematical formulations of three types of structural optimization problems with pure integer and mixed-integer design variables are established. The parts allocation problem is formulated into nonlinear assignment problem, which is a type of pure combinatorial optimization. Proper interpolation schemes is established so that the function evaluation at fraction points is mathematical meaningful. Based on this property, the developed initial solution generation method and the accelerated neighbor-search algorithm is applied. It demonstrates that the whole procedure is able to reduce the number of iterations and thus the number of function evaluations effectively.

The stacking sequence optimization in laminate structure design is also formulated into nonlinear assignment problem. The same procedure as in solving the parts allocation problem is transplanted to deal with this problem. The initial solution generation method is modified in two ways: when the material stiffness quantities are acceptable by finite element solvers, an interpolation scheme is utilized. Otherwise, a quasi-homogenous technique is developed and employed. To fit the contiguity constraint, a repair operator aims at repairing as many as possible contiguity violations with the least number of changes of stacking sequence is developed. The performance of the operator is proved through large-scale examples. The developed procedure is applied on practical straight-fiber laminate sequence optimization and also achieved success in variable-stiffness laminate structure optimization.

The simultaneous material selection and size optimization problem is formulated into a mixed-integer nonlinear optimization problem, with both binary and

continuous type design variables. The developed constraint-directed binary assignment algorithm is directly applied. The efficiency of the algorithm is demonstrated with small-scale to large-scaled bar truss structures and a panel design problem. A strategy to improve the quality of final solutions is also presented.

The accelerated neighbor-search algorithm shows great application potential in structure optimization with pure integer variables. The idea of fully utilizing neighbor-search information is important in the design of neighbor-search based algorithms to solve assignment problems. The acceleration technique presented here is just an example. It could be properly adjusted to fit further assignment type problems. The constraint-directed approach shows its capability in solving mixed-integer optimization with binary assignment. Extension of this algorithm to adapt to more general types of MINLP problems, for example problems with natural integers, and improvement on the quality of the solutions by developing and integrating more advanced nonlinear optimization solvers are interesting points in future study. From the point of view of an entire solution procedure of structural optimization problems, surrogate models can be combined with the solution algorithm to further improve the efficiency.

Bibliography

[Aarts 2003] E. Aarts and J. K. Lenstra, Local Search in Combinatorial Optimization. Princeton, NJ, USA: Princeton University Press, 2003, pp. 57–214.

[Akrotirianakis 2001] I. Akrotirianakis, I. Maros and B. Rustem, An Outer Approximation based Branch and Cut Algorithm for convex 0-1 MINLP problems, Optimization Methods and Software, 16 (2001), pp. 21-47.

[Apalak 2011] Z. G. Apalak, M. K. Apalak, R. Ekici and M. Yildirim, Layer optimization for maximum fundamental frequency of rigid point-supported laminated composite plates, Polymer Composites Volume 32, Issue 12, pages 1988–2000, December 2011.

[ASM 2014] ASM Aerospace Specification Metals Inc.: Aluminum 2014-T6 2014-T651; Aluminum 2024-T3; Aluminum 6061-T6 6061-T651; Aluminum 7075-T6 7075-T651.
http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA2014T6
http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA2024T3
http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA6061T6
http://asm.matweb.com/search/SpecificMaterial.asp?bassnum=MA7075T6

[Bendsoe 2003] M. P. Bendsoe and O. Sigmund, Topology Optimization Theory, Methods, and Applications, Springer, 2003.

[Bonami 2008] Pierre Bonami, Lorenz T. Biegler, Andrew R. Conn etc., An Algorithm Framework for Convex Mixed Integer Nonlinear Programs, Discrete Optimization 5 (2008) pp. 186-204.

[Burkard 1997] R.E. Burkard, E. Çela, S.E. Karisch and F. Rendl, QAPLIB – A Quadratic Assignment Problem Library, Journal of Global Optimization, 1997, 10, pp. 391-403.

[Burkard 2009] R. Burkard, M. Dell'Amico, and S. Martello, Assignment Problems. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, ch. Linear sum assignment problem, 2009, pp.

73–144.

[Chang 2010] N. Chang, W. Wang, W. Yang and J. Wang. Ply stacking sequence optimization of composite laminate by permutation discrete particle swarm optimization. Struct Multidisc Optim, 41:179-187, 2010.

[Chen 2008] J. Chen, R. Ge, and J. Wei. Probabilistic optimal design of laminates using improved particle swarm optimization. Engineering Optimization, 40(8):695-708, August 2008.

[Duran 1986] M. Duran and I.E. Grossmann, An outer-approximation algorithm for a class of mixed-integer nonlinear programs, Mathematical Programming 36 (1986) pp.307-339.

[Feo 1991] T. Feo, K. Venkatraman and J. Bard, A grasp for difficult single machine scheduling problem, Computers&Operations Research, 18:635-643, 1991.

[Fletcher 1994] R. Fletcher and S. Leyffer, Solving mixed integer nonlinear programs by outer approximation, Mathematical Programming 66 (1994), pp. 327-349.

[Floudas 1995] C. A. Floudas, Nonlinear and Mixed-Integer Optimization—Fundamentals and Applications, New York, Oxford: OXFORD UNIVERSITY PRESS, 1995.

[Frieze 1983] A. M. Frieze, Complextiy of a 3-dimensional assignment problem, European Journal of Operation Research, vol. 13, no. 2, pp. 161–164, 1983.

[Glover 1997] F. Glover and M. Laguna, Tabu Search. Kluwer, Dordrecht, 1997.

[Gomory 1958] R. Gomory, Outline of an algorithm for integer solutions to linear problems, Bulletin of the American Mathematical Society, 64 (1958), pp. 275-278.

[Graves 1983] S. C. Graves and B. W. Lamar, An integer programming procedure

for assembly design problems, Operations Research, 31(3) 1983, 522-545.

[Gupta 1985] O.K. Gupta and V. Ravindran, Branch and bound experiments in convex nonlinear integer programming, Management Science 31 (1985), pp. 1533-1546.

[Gürdal 1999] Z. Gürdal, R. T. Haftka and P. Hajela, Design and Optimization of Laminated Composite Materials, John Wiley & Sons, 1999.

[Gutkowski 1994] W. Gutkowski, Discrete structural optimization, Springer, 1994.

[Huber 2010] M. Huber, D. Neufeld, , J. Chung,, K. Behdinan and H. Baier, Data Mining Based Mutation Function for optimization problems with mixed continuous-discrete design variables, International Journal of Structural and Multidisciplinary Optimization: Volume 41, Issue 4, 2010.

[Huber 2010] M. Huber, Structural Design Optimization Including Quantitative Manufacturing Aspects Derived from Fuzzy Knowledge, Doctor dissertation, LLB, TU München 2010.

[Kirkpatrick 1983] S. Kirkpatrick, C. G. Jr. and M. Vecchi, Optimization by simulated annealing, Science, 1983, 220:671-680.

[Kuhn 1955] H. W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly, 1955, 2:83–97.

[Lee 2012] J. Lee and S. Leyffer, Mixed Integer Nonlinear Programming, Series: The IMA Volumes in Mathematics and its Applications, Vol. 154, Springer 2012.

[Lehmann 2013] S. Lehmann, K. Schittkowski, M. Stingl, F. Wein and C. Zillober, 2013, A strictly feasible sequential convex programming method applied to free material optimization

[Li 2000] Q. Li, G. P. Steven and Y.M. Xie, Evolutionary structural

optimization for stress minimization problems by discrete thickness design, Computers & Structures, Volume 78, Issue 6, December 2000, Pages 769–780.

[Li 2006] D. Li and X. Sun, Nonlinear Integer Programming, Series: International Series in Operations Research & Management Science, Vol. 84, Springer 2006.

[Liu 2000] B. Liu, R. T. Haftka, M. A. Akgün and A Todoroki, Permutation genetic algorithm for stacking sequence design of composite laminates, Comput. Methods Appl. Mech. Engrg. 186 (2000), pp. 357-372.

[Munkres 1957] J. Munkres, Algorithms for the Assignment and Transportation Problems, Journal of the Society for Industrial and Applied Mathematics, vol. 5, no. 1, pp. 32–38, 1957.

[Papadimitriou 1998] H. Papadimitriou and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Dover Publications, 1998.

[Pardalos 2000] P. M. Pardalos and L. S. Pitsoulis, Nonlinear Assignment Problems: Algorithms and Applications (Combinatorial Optimization). Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2000, ch. Heuristics for Nonlinear Assignment Problems, pp. 175–215.

[Quesada 1992] I. Quesada and I.E. Grossmann, An LP/NLP based branched and bound algorithm for convex MINLP optimization problems, Computers and Chemical Engineering 16 (1992) pp.937-947.

[Riche 1993] R. Riche and R. T. Haftka, Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm, AIAA Journal, Vol. 31, No. 5 (1993), pp. 951-956.

[Sahni 1976] S. Sahni and T. Gonzalez, P-complete approximation problems, Journal of the Association of Computing Machinery, 23 (1976), pp. 555–565.

[Salman 2002] A. Salman, I. Ahmad and S. Al-Madani, Particle swarm optimization for task assignment problem, Microprocessors and Microsystems, Vol. 26(8): pp. 363–371, 2002.

[Schittkowski 2006] O. Exler and K. Schittkowski, A Trust Region SQP Algorithm for Mixed-Integer Nonlinear Programming, 2006.

[Schittkowski 2013] O. Exler, T. Lehmann and K. Schittkowski: A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization, Mathematical Programming Computation, Vol. 4, 383-41, 2013.

[Stegmann 2005] J. Stegmann and E. Lund, Discrete material optimization of general composite shell structures, International Journal for Numerical Methods in Engineering, Volume 62, Issue 14, pp. 2009–2027.

[Stubbs 1999] R. Stubbs and S. Mehrotra, A branc-and-cut method for 0-1 mixed convex programming, Mathematical Programming 86 (1999), pp. 515-532.

[Syslo 2006] M. M. Syslo, N. Deo and J. S. Kowalik, Discrete Optimization Algorithms: with Pascal Programs, Dover Publications, 2006.

[Tsai 1968] S. W. Tsai and N. J. Pagano, Invariant Properties of Composite Materials, in: Composite Materials Workshop, Technomic Publishing Company, Westport, Conn., 1968

[Tsutsui 2007] S. Tsutsui, Cunning Ant System for Quadratic Assignment Problem with Local Search and Parallelization, Pattern Recognition and Machine Intelligence, Lecture Notes in Computer Science Volume 4815, 2007, pp 269-278.

[Ungwattanapanit 2012] T. Ungwattanapanit and H. Baier, Strength-Stability Optimization of Hybrid-Stiffness Laminated Plates with Cut-outs subjected to Multiple Loads, RAeS Aircraft Structural Design Conference 2012, Delft.

[Ungwattanapanit T. Ungwattanapanit, Y. Zhang and H. Baier, Hybrid-Stiffness

2013] Laminated Fuselage Panels with Window Cutouts Optimized under Stability and Strength Constraints, ICCS 17 2013, Porto.

[Vavasis 1991] S. Vavasis, Nonlinear Optimization: Complexity Issues. Oxford University Press, New York, N.Y., 1991.

[Westerlund 1995] T. Westerlund and F. Pettersson, An extended cutting plane method for solving convex MINLP problems, Computers & Chemical Engineering 19 (1995), pp. 131-136.

[Zhang 2011] Y. Zhang and H. Baier, Optimal Parts Allocation for Structural Systems Via Improved Initial Solution Generation. VII. ALIO / EURO Workshop on Applied Combinatorial Optimization 2011, Porto, Portugal, 2011.

Appendix A. Full test results of QAPLIB problems

Table A-1: Test results of developed procedure on QAPLIB

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
bur26a	26	5426670	5432907	0.114932	11	4264
bur26b	26	3817852	3836581	0.490564	11	4264
bur26c	26	5426795	5427739	0.017395	12	4590
bur26d	26	3821225	3865437	1.157011	14	5242
bur26e	26	5386879	5411820	0.462995	15	5568
bur26f	26	3782044	3823154	1.086978	11	4264
bur26g	26	10117172	10244156	1.255133	10	3938
bur26h	26	7098658	7191733	1.311163	11	4264
chr12a	12	9552	13194	38.12814	5	481
chr12b	12	9742	14378	47.58776	7	615
chr12c	12	11156	13974	25.25995	5	481
chr15a	15	9896	13376	35.16572	6	863
chr15b	15	7990	10956	37.1214	7	969
chr15c	15	9504	16826	77.04125	4	651
chr18a	18	11098	17948	61.72283	8	1558
chr18b	18	1534	1938	26.33638	8	1558
chr20a	20	2192	2806	28.01095	9	2121
chr20b	20	2298	3236	40.8181	8	1930
chr20c	20	14142	23758	67.99604	6	1548
chr22a	22	6156	7266	18.03119	8	2342
chr22b	22	6194	7360	18.82467	8	2342
chr25a	25	3796	5196	36.88093	12	4239
els19	19	17212548	20508428	19.14812	10	2083
esc16a	16	68	68	0	6	984
esc16b	16	292	292	0	4	742
esc16c	16	160	168	5	4	742
esc16d	16	16	18	12.5	7	1105
esc16e	16	28	30	7.142857	3	621
esc16f	16	0	0	0	1	379
esc16g	16	26	26	0	2	500
esc16h	16	996	996	0	5	863
esc16i	16	14	14	0	3	621

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
esc16j	16	8	12	50	3	621
esc32a	32	----	152	----	10	5996
esc32b	32	----	212	----	10	5996
esc32c	32	----	646	----	6	4008
esc32d	32	----	210	----	9	5499
esc32e	32	2	2	0	3	2517
esc32f	32	2	2	0	3	2517
esc32g	32	6	8	33.33333	3	2517
esc128	128	64	64	0	15	138321
had12	12	1652	1676	1.452785	5	481
had14	14	2724	2724	0	5	658
had16	16	3720	3750	0.806452	6	984
had18	18	5358	5414	1.045166	9	1712
had20	20	6922	7044	1.762496	8	1930
kra30a	30	88900	93700	5.399325	13	6570
kra30b	30	91420	94110	2.942463	9	4826
kra32	32	17740	18664	5.208568	12	6990
lipa20a	20	3683	3782	2.688026	12	2694
lipa20b	20	27076	27076	0	6	1548
lipa30a	30	13178	13431	1.919866	9	4826
lipa30b	30	151426	151426	0	11	5698
lipa40a	40	31583	31995	1.304499	18	15660
lipa40b	40	476581	476581	0	9	8631
lipa50a	50	62093	62832	1.19015	14	19666
lipa50b	50	1210244	1210244	0	7	11084
lipa60a	60	107218	108407	1.108956	26	49648
lipa60b	60	2520135	2520135	0	7	15999
lipa70a	70	169755	171046	0.760508	39	99126
lipa70b	70	4603200	4603200	0	10	29062
lipa80a	80	253195	255388	0.866131	35	117037
lipa80b	80	7763962	9369127	20.67456	45	148647
lipa90a	90	360630	363362	0.757563	45	188372
lipa90b	90	12490441	12490441	0	8	40150
nug12	12	578	616	6.574394	7	615
nug14	14	1014	1018	0.394477	6	750
nug15	15	1150	1260	9.565217	4	651

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
nug16a	16	1610	1686	4.720497	6	984
nug16b	16	1240	1262	1.774194	10	1468
nug17	17	1732	1758	1.501155	7	1250
nug18	18	1930	1968	1.968912	5	1096
nug20	20	2570	2664	3.657588	5	1357
nug21	21	2438	2510	2.95324	5	1498
nug22	22	3596	3628	0.889878	8	2342
nug24	24	3488	3646	4.529817	8	2794
nug25	25	3744	3898	4.113248	8	3035
nug27	27	5234	5448	4.088651	10	4251
nug28	28	5166	5330	3.174603	9	4197
nug30	30	6124	6358	3.821032	14	7006
rou12	12	235528	255614	8.528073	5	481
rou15	15	354210	367966	3.883572	4	651
rou20	20	725522	751674	3.604577	8	1930
scr12	12	31410	35002	11.43585	5	481
scr15	15	51140	56124	9.745796	7	969
scr20	20	110030	116536	5.912933	8	1930
sko81	81	90998	92676	1.843997	29	100552
sko90	90	115534	117010	1.277546	33	140300
sko100a	100	152002	155898	2.563124	33	173385
sko100b	100	153890	156136	1.459484	40	208042
sko100c	100	147862	150234	1.604199	34	178336
sko100d	100	149576	151860	1.526983	34	178336
sko100e	100	149150	153048	2.613476	27	143679
sko100f	100	149036	151062	1.359403	30	158532
ste36a	36	9526	10254	7.642242	11	8239
ste36b	36	15852	19052	20.18673	15	10763
ste36c	36	8239110	9648150	17.10185	8	6346
tai12a	12	224416	244672	9.026094	3	347
tai12b	12	39464925	43975950	11.43047	4	414
tai15a	15	388214	393178	1.278676	7	969
tai15b	15	51765268	51999162	0.451836	7	969
tai17a	17	491812	506346	2.955194	9	1524
tai20a	20	703482	743604	5.703344	9	2121
tai20b	20	1.22E+08	1.37E+08	12.22154	11	2503

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
tai25b	25	3.44E+08	3.71E+08	7.646	7	2734
tai30b	30	6.37E+08	7.3E+08	14.56366	13	6570
tai35b	35	2.83E+08	3.25E+08	14.8207	14	9571
tai40b	40	6.37E+08	6.68E+08	4.801161	25	21127
tai50b	50	4.59E+08	4.77E+08	3.920445	25	33152
tai60a	60	7208572	7539198	4.586567	26	49648
tai60b	60	6.08E+08	6.77E+08	11.2575	26	49648
tai64c	64	1855928	1865494	0.515429	13	30319
tai80a	80	13557864	13959286	2.960806	47	154969
tai80b	80	8.18E+08	8.46E+08	3.416145	32	107554
tai100b	100	1.19E+09	1.2E+09	1.231798	34	178336
tai150b	150	4.99E+08	5.09E+08	2.069321	54	626006
tai256c	256	44759294	44879440	0.268427	72	2415690
tho30	30	149936	157712	5.186213	12	6134
tho150	150	8133398	8293640	1.970173	52	603654
wil100	100	273038	274946	0.698804	35	183287
esc32h	----	----	446	----	11	6493
esc64a	----	----	118	----	6	16200
sko42	42	15812	16406	3.756641	16	15558
sko49	49	23386	23864	2.043958	20	25943
sko56	56	34458	34846	1.126008	32	52450
sko64	64	48498	49252	1.554703	22	48472
sko72	72	66256	67476	1.841343	23	63997
tai10a	10	----	135828	----	3	240
tai10b	10	----	1221121	----	5	332
tai25a	25	1167256	1242244	6.424298	9	3336
tai30a	30	1818146	1950320	7.269713	7	3954
tai40a	40	3139370	3285000	4.638829	18	15660
tai50a	50	4938796	5138448	4.042524	21	28248

Table A-2: Test results of procedure without acceleration technique on QAPLIB

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
bur26a	26	5426670	5435663	0.165719	18	6528
bur26b	26	3817852	3836672	0.492947	12	4578
bur26c	26	5426795	5429054	0.041627	19	6853

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
bur26d	26	3821225	3865472	1.157927	16	5878
bur26e	26	5386879	5412611	0.477679	27	9453
bur26f	26	3782044	3827490	1.201625	16	5878
bur26g	26	10117172	10231402	1.12907	21	7503
bur26h	26	7098658	7203633	1.478801	13	4903
chr12a	12	9552	13194	38.12814	5	476
chr12b	12	9742	14378	47.58776	8	674
chr12c	12	11156	13974	25.25995	6	542
chr15a	15	9896	13376	35.16572	6	857
chr15b	15	7990	10082	26.18273	8	1067
chr15c	15	9504	13228	39.1835	6	857
chr18a	18	11098	17948	61.72283	10	1856
chr18b	18	1534	1938	26.33638	8	1550
chr20a	20	2192	3210	46.44161	8	1922
chr20b	20	2298	3108	35.24804	12	2682
chr20c	20	14142	27510	94.52694	16	3442
chr22a	22	6156	7118	15.62703	12	3258
chr22b	22	6194	7360	18.82467	8	2334
chr25a	25	3796	5196	36.88093	19	6327
els19	19	17212548	20508428	19.14812	16	3099
esc16a	16	68	70	2.941176	5	858
esc16b	16	292	292	0	4	738
esc16c	16	160	160	0	5	858
esc16d	16	16	18	12.5	7	1098
esc16e	16	28	30	7.142857	3	618
esc16f	16	0	0	0	1	378
esc16g	16	26	26	0	3	618
esc16h	16	996	996	0	5	858
esc16i	16	14	14	0	3	618
esc16j	16	8	12	50	3	618
esc32a	32	----	158	----	14	7970
esc32b	32	----	192	----	19	10450
esc32c	32	----	646	----	9	5490
esc32d	32	----	204	----	12	6978
esc32e	32	2	2	0	3	2514
esc32f	32	2	2	0	3	2514

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
esc32g	32	6	8	33.33333	3	2514
esc128	128	64	66	3.125	17	154562
had12	12	1652	1676	1.452785	6	542
had14	14	2724	2724	0	5	653
had16	16	3720	3750	0.806452	9	1338
had18	18	5358	5444	1.605077	7	1397
had20	20	6922	7056	1.935857	12	2682
kra30a	30	88900	97780	9.988751	16	7862
kra30b	30	91420	94240	3.084664	20	9602
kra32	32	17740	19494	9.88726	16	8962
lipa20a	20	3683	3782	2.688026	12	2682
lipa20b	20	27076	30957	14.33373	5	1352
lipa30a	30	13178	13443	2.010927	17	8297
lipa30b	30	151426	151426	0	22	10472
lipa40a	40	31583	31951	1.165184	25	21102
lipa40b	40	476581	476581	0	25	21102
lipa50a	50	62093	62894	1.29	26	34352
lipa50b	50	1210244	1210244	0	33	42927
lipa60a	60	107218	108363	1.067918	32	60242
lipa60b	60	2520135	2520135	0	39	72632
lipa70a	70	169755	171258	0.885394	40	101502
lipa70b	70	4603200	4603200	0	58	144972
lipa80a	80	253195	255219	0.799384	45	148602
lipa80b	80	7763962	9435946	21.53519	39	129642
lipa90a	90	360630	363458	0.784183	47	196337
lipa90b	90	12490441	12490441	0	61	252407
nug12	12	578	616	6.574394	8	674
nug14	14	1014	1018	0.394477	9	1017
nug15	15	1150	1260	9.565217	4	647
nug16a	16	1610	1686	4.720497	6	978
nug16b	16	1240	1298	4.677419	8	1218
nug17	17	1732	1752	1.154734	9	1515
nug18	18	1930	1998	3.523316	7	1397
nug20	20	2570	2636	2.568093	7	1732
nug21	21	2438	2480	1.722724	15	3593
nug22	22	3596	3690	2.614016	14	3720

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
nug24	24	3488	3694	5.905963	13	4166
nug25	25	3744	3822	2.083333	18	6027
nug27	27	5234	5446	4.050439	16	6347
nug28	28	5166	5340	3.368177	16	6834
nug30	30	6124	6388	4.310908	20	9602
rou12	12	235528	255614	8.528073	5	476
rou15	15	354210	367966	3.883572	6	857
rou20	20	725522	744514	2.617701	12	2682
scr12	12	31410	35002	11.43585	5	476
scr15	15	51140	56124	9.745796	9	1172
scr20	20	110030	115094	4.602381	17	3632
sko81	81	90998	93652	2.916548	76	252803
sko90	90	115534	117792	1.954403	75	308477
sko100a	100	152002	156120	2.709175	94	475302
sko100b	100	153890	156836	1.914354	132	663402
sko100c	100	147862	150798	1.985635	106	534702
sko100d	100	149576	152028	1.6393	131	658452
sko100e	100	149150	152838	2.472679	96	485202
sko100f	100	149036	151260	1.492257	119	599052
ste36a	36	9526	10382	8.985933	24	16418
ste36b	36	15852	18872	19.05122	23	15788
ste36c	36	8239110	9648150	17.10185	13	9488
tai12a	12	224416	244672	9.026094	3	344
tai12b	12	39464925	43975950	11.43047	7	608
tai15a	15	388214	393178	1.278676	10	1277
tai15b	15	51765268	51999162	0.451836	11	1382
tai17a	17	491812	506346	2.955194	10	1651
tai20a	20	703482	732262	4.091078	13	2872
tai20b	20	1.22E+08	1.38E+08	12.49783	11	2492
tai25b	25	3.44E+08	3.66E+08	6.252887	20	6627
tai30b	30	6.37E+08	7.3E+08	14.64143	21	10037
tai35b	35	2.83E+08	3.25E+08	14.63668	36	22647
tai40b	40	6.37E+08	6.68E+08	4.785443	41	33582
tai50b	50	4.59E+08	4.79E+08	4.42153	55	69877
tai60a	60	7208572	7496622	3.995937	44	81482
tai60b	60	6.08E+08	6.82E+08	12.17491	68	123962

Problem	Size	QAPLIB_opt	Found_opt	Error(%)	Num Ite	FuncEva
tai64c	64	1855928	1865494	0.515429	13	30306
tai80a	80	13557864	14020750	3.414151	47	154922
tai80b	80	8.18E+08	8.48E+08	3.55567	126	404562
tai100b	100	1.19E+09	1.23E+09	4.04011	121	608952
tai150b	150	4.99E+08	5.15E+08	3.225353	208	2346902
tai256c	256	44759294	44961636	0.452067	69	2317698
tho30	30	149936	157954	5.347615	21	10037
tho150	150	8133398	8343436	2.582414	166	1877552
wil100	100	273038	276140	1.136106	103	519852
esc32h	----	----	476	----	10	5986
esc64a	----	----	116	----	13	30306
sko42	42	15812	16370	3.528965	32	29318
sko49	49	23386	23910	2.240657	35	43563
sko56	56	34458	34958	1.451042	46	73978
sko64	64	48498	49792	2.668151	49	102882
sko72	72	66256	67164	1.370442	77	201998
tai10a	10	----	135828	----	3	237
tai10b	10	----	1228910	----	4	282
tai25a	25	1167256	1242244	6.424298	9	3327
tai30a	30	1818146	1950320	7.269713	7	3947
tai40a	40	3139370	3284656	4.627871	26	21882
tai50a	50	4938796	5158174	4.441933	21	28227

Appendix B. Optimal stacking sequence

B.1. Thin composite panel examples

Table B-1: Optimal stacking sequence for the thin composite panel

Load case	Obj.	Stacking sequence
1	0.9485	$[(\pm45)_{18}/(90_2)_2/0_2/(90_2)_2/0_2/(90_2)_2/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2]$
2	0.9486	$[(\pm45)_{17}/(90_2)_2/0_2/(90_2)_2/0_2/90_2/0_2/90_2/(0_2)_2/90_2/(90_2)_2/90_2/0_2]$
3	0.9101	$[(\pm45)_{15}/(90_2)_2/0_2/(90_2)_2/0_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2]$
4	0.8710	$[(\pm45)_{12}/(90_2)_2/0_2/(90_2)_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2]$
5	0.7763	$[(\pm45)_8/(90_2)_2/0_2/90_2/(0_2)_2/90_2/0_2]$
6	0.7756	$[(\pm45)_{15}/(90_2)_2/\pm45/(90_2)_2/0_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2]$
7	0.7820	$[(\pm45)_8/90_2/(0_2)_2/(90_2)_2/0_2/(90_2)_2/0_2/(90_2)_2/0_2/90_2/0_2/90_2/0_2/(90_2)_2/0_2/90_2/0_2/90_2]$
8	1.1042	$[(\pm45)_7/(90_2)_2/0_2/(90_2)_2/0_2/(90_2)_2/0_2/(90_2)_2/0_2/(90_2)_2/0_2/90_2/0_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2]$
9	2.7967	$[90_2/\pm45/(90_2)_2/\pm45/(90_2)_2/\pm45/90_2/\pm45/90_2/\pm45/(90_2)_2/\pm45/(90_2)_2/\pm45/0_2/90_2/(0_2)_2/\pm45/(0_2)_2/\pm45/(0_2)_2/\pm45/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2]$
10	2.7951	$[90_2/(\pm45)_2/(90_2)_2/\pm45/(90_2)_2/\pm45/(90_2)_2/\pm45/(90_2)_2/\pm45/(90_2)_2/\pm45/0_2/90_2/(0_2)_2/\pm45/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/(0_2)_2/90_2/0_2/]$
11	0.9584	$[(90_2)_2/\pm45/90_2/\pm45/\pm45/(90_2)_2/(\pm45)_2/(90_2)_2/\pm45/(90_2)_2/0_2/90_2/(0_2)_2/\pm45/(0_2)_2/90_2/(0_2)_2/\pm45/(0_2)_2/90_2/0_2]$

B.2. Very thick composite panel examples

Table B-2: Optimal stacking sequence for the very thick composite panel

Load case	Obj.	Stacking sequence
1	4.0401	$[(\pm45)_{72}/(\pm60)_{24}/(\pm30)_{24}/(90_2)_5/(0_2)_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/$

Load case	Obj.	Stacking sequence
2	4.0370	$(90_2)_5/0_2/(90_2)_5/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_3/90_2/(0_2)_2]$ $[(\pm45)_{68}/(\pm60)_{22}/(\pm30)_{22}/(90_2)_5/0_2/(90_2)_2/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_2]$
3	3.8301	$[(\pm45)_{60}/(\pm60)_{18}/(\pm30)_{20}/(90_2)_5/0_2/(90_2)_4/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_3/0_2/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/0_2/90_2/(0_2)_2]$
4	3.7497	$[(\pm45)_{48}/(\pm60)_{16}/(\pm30)_{16}/(90_2)_5/(0_2)_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_3/90_2/(0_2)_2]$
5	3.8307	$[(\pm45)_{32}/(\pm60)_{10}/(\pm30)_{12}/(90_2)_5/0_2/(90_2)_4/0_2/(90_2)_3/0_2/90_2/(0_2)_5/90_2/(0_2)_5/90_2/0_2/90_2/(0_2)_2]$
6	3.6937	$[(\pm45)_{64}/(\pm60)_{20}/(\pm30)_{22}/(90_2)_5/0_2/(90_2)_2/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_2]$
7	4.8251	$[(90_2)_5/\pm30/(90_2)_5/0_2/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/(\pm30)_{20}/(\pm45)_{32}/(\pm60)_{16}/(90_2)_3/60_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/60_2/(0_2)_5/60_2/0_2/60_2/(0_2)_2]$
8	5.0305	$[(\pm45)_{28}/(\pm60)_{22}/(\pm30)_{23}/90_2/30_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/0_2/90_2/(0_2)_2]$
9	14.171	$[0_2/(90_2)_2/30_2/90_2/60_2/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/90_2/60_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/(\pm60)_{24}/(\pm45)_{40}/(\pm30)_{26}/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/0_2/90_2]$
10	14.034	$[(0_2)_3/(90_2)_2/60_2/(90_2)_2/60_2/(90_2)_4/(0_2)_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/0_2/(90_2)_5/60_2/(90_2)_2/(\pm60)_{23}/(\pm45)_{32}/(\pm30)_{26}/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_4/(90_2)_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_3/(90_2)_2/0_2/(90_2)]$
11	5.1018	$[(\pm45)_5/(\pm60)_2/(90_2)_5/\pm45/(90_2)_5/\pm45/(90_2)_5/\pm45/(90_2)_5/\pm45/(90_2)_5/\pm45/(90_2)_4/\pm60/\pm45/(90_2)_4/\pm60/90_2/\pm60/(90_2)_2/(\pm60)_{14}/(\pm45)_2/90_2/\pm60/(\pm45)_{19}/\pm30/90_2/(\pm30)_5/(90_2)_2/(\pm30)_2/90_2/(\pm30)_{11}/(0_2)_5/\pm30/(0_2)_5/90_2/(0_2)_4/90_2/(0_2)_4/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_5/90_2/(0_2)_2]$

B.3. Composite panel with stiffeners

Table B-3: Optimal stacking sequence considering only buckling

Panel	$[(0_{10}/\pm30)_2/(0_6/\pm30)_2/(\pm30)_5/\pm45/\pm30/(\pm45)_9/(\pm60)_7/90_8/\pm60/(90_{10}/\pm60)_2/90_4]_s$
Stiffener	$[\pm45/90_2/\pm60/0_2/90_2/\pm45/0_2/\pm60/\pm45/0_4/\pm30/0_4/\pm30/90_2/\pm30/0_2/\pm45/90_2/(\pm60)_2/90_4/0_2/\pm30/(\pm45)_2/(\pm30)_2/90_2/0_2/\pm45/0_2/90_2/\pm45/\pm60/90_2/0_2/(\pm60)_2/\pm30/\pm60/0_2/\pm45/0_2/90_2/0_4/90_6/\pm60/(\pm30)_2/90_2/\pm30/\pm60/0_2/90_4/\pm45]_s$

Table B-4: Optimal stacking sequence considering buckling, strain and stress

Panel	$[(\pm45)_{10}/(\pm30)_{10}/0_{10}/\pm60/0_6/\pm60/0_8/(\pm60)_7/0_2/90_8/0_2/90_4/0_2/90_{10}/0_2/90_6/\pm60/90_4]_s$
Stiffener	$[\pm45/\pm60/0_4/\pm60/90_2/0_2/\pm30/0_2/(\pm60)_2/0_2/90_2/(\pm30/0_2)_2/90_4/\pm60/0_2/\pm45/0_6/(\pm60)_2/\pm45/0_2/\pm30/90_2/\pm60/0_2/\pm45/90_2/(\pm45)_2/\pm30/90_2/\pm30/\pm45/\pm30/\pm60/90_8/0_2/\pm45/90_4/\pm45/\pm30/90_4/\pm30/\pm45/90_2/0_2/\pm30/\pm60/0_2]_s$