Hard Optimization of Structural Design Subjected to Buckling Using the Evolutionary Computation Approach

by

Ali Elmbrok Salem AHMID

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE IN PARTIAL FULFILLMENT FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, FEBRURAY 10, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

ACKNOWLEDGMENT

I would like to express my feelings and gratitude to my great supervisors Prof.Van Ngan Le and Prof. Thien-my Dao for their support and guidance and thank them for offering me the opportunity to work under their supervision. I truly appreciate the time they gave whenever I needed their assistance. Being your student has been the most remarkable experience that I obtained in my academic life.

I am truly grateful to Advanced Center of Technology (ACT-Tripoli) and Libyan government for giving me the opportunity to study abroad. I thank my colleagues at ACT Alla Alzoubi, Hatem Gnaba and all my supporters. I am indeed thankful to Dr. Ahmed Arkoub, for his unceasing and significant help from the very beginning of my study here in Canada.

I would also like to thank my colleagues in mechanical engineering department at Ecole Technologies Superrior for their kind friendship and continues support especially Thi Hong Dang Nguyen and Azedden Abdo.

Additionally, a special thanks go to my examiner committee: our guest examiner Prof. Nabil Nahas, Prof. Henri Champliaud, and the president Prof. Tony Wong. Thanks for giving me the honor by being my Ph.D. committee, and for your valuable time and efforts.

Lastly, I would like to thank my mom, brothers and sisters for their emotional care and unconditioned support even though what they are going through there. Furthermore, I desire to thank my wife for her continuous support and encouragement over the years that we spent here. Your kindness, patient and quietness during hard times we went through together is priceless support that I will appreciate for the rest of my life. Also, my beloved ones Abdo, Ayeob, and Soheb; and my sweethearts Fatma and Safia; I could never express how grateful I am for the sacrifices you made during this long journey which have allowed me being fully concentrated on my work. Believe me I love you all more than meta-heuristics!

Optimisation dure de la conception structurelle soumise au flambage à l'aide de l'approche de calcul évolutif

Ali Elmbrok Salem AHMID

RÉSUMÉ

Les techniques d’optimisation ont pour objectif de réduire la quantité des matériaux utilisés lors des opérations de conception structurelles sans pour autant altérer les exigences imposées par les processus de fabrication, menant ainsi à une réduction de coûts. En plus du coût réduit ainsi que l’impact positif de durabilité résultant de l’optimisation de conception structurelle (Structural Design Optimization, SDO), elle sert aussi d’outil prometteur pour les ingénieurs à développer des conceptions innovantes dans diverses applications structurelles réelles.

Les études antérieures dans le domaine SDO ont été d’un apport considérable dans l’approfondissement de nos connaissances dans ce champ d’intérêt spécifique. Cependant, cette thématique de recherche fait encore l’objet de débats dans ces différents aspects, non seulement sur la façon de réaliser les caractéristiques physiques requises d’une structure, mais aussi, et surtout, sur la possibilité d’atteindre un tel objectif efficacement, et avec un temps de calcul le moins couteux possible. De ce point de vue, certains axes de recherche, potentiellement prometteurs, qui ont été repris de la littérature, ainsi que de nouvelles contributions, ont été traités dans le cadre de la présente thèse.

La revue de littérature portant sur l’optimisation des problèmes de conception structurelle révéle que les algorithmes d’optimisation globale ou ce qui est communément appelé Méta-heuristiques (MHs) sont considérés comme étant les meilleures techniques disponibles en mesure de résoudre de tels problèmes complexes d’optimisation. Toutefois, le principal défi de l’ingénieur est de trouver l’algorithme MH qui répond au mieux au problème de conception structurelle qu’il doit solutionner. Malheureusement, les travaux rapportés dans la littérature révèlent le manque d’un modèle d’évaluation systématique qui pourrait aider les ingénieurs à surmonter cette insuffisance. Les mesures les plus communément utilisées actuellement pour la performance des MHs sont les opérateurs statistiques des solutions obtenues tels que : min, max, moyenne et écart-type. Cependant, de tels opérateurs ne sont pas assez suffisant pour refléter la performance actuelle des algorithmes MH lorsque ces mesures sont utilisées séparément. Ainsi, un critère d’évaluation adéquat a été développé dans le cadre de ce travail afin d’inclure plus de paramètres efficaces comme la fiabilité pratique, le prix (le coût de calcul), le prix normalisé, le taux de performance, la qualité de la solution et l’analyse Fitness-landscape. De plus, deux différents taux de convergence ont été imposés pour examiner les algorithmes MH pour les taux lent et rapide, ainsi que la reproductibilité des résultats d’expériences numériques considérés lors de procédure d’évaluation des MHs. Récemment, le critère proposé a été utilisé afin de comparer cinq différentes variantes d’optimisation par colonies de fourmis (Ant Colony , ACO). Les démarches proposées ont montré une évaluation comparative efficiente de la performance de l’optimisation par colonies de fourmis.
Plusieurs études ont été menées afin d’améliorer la performance de recherche des techniques MHs, et les résultats ont été prometteurs dans cette direction. Malheureusement, la littérature

portant sur l’optimisation de conception structurelle a montré une tendance forte à développer de nouveaux “metaphor” MHs au lieu de procéder à une amélioration de la performance des algorithmes MHs déjà existants, et qui ont une remarquable réputation dans la résolution des problèmes d’optimisation Non-Polynôme ($\mathcal{NP}$) qui sont connus par leur complexité. Cependant, la présente thèse examine d’éventuelles améliorations des caractéristiques de recherche des deux approches MHs sélectionnés moyennant une intégration d’une recherche locale de mouvements à la structure principale de l’algorithme MH afin d’améliorer l’effort d’intensification. Le premier algorithme MH sélectionné est Cuckoo Search (CS), qui est fortement utilisé pour résoudre une variété de problèmes d’optimisation tels que la minimisation pondérée des structures en treillis, Travelling Salesman Problem (TSP),…etc. L’algorithme de recherche Cuckoo (CS) est conçu pour résoudre des problèmes d'optimisation continue sans contrainte comme la plupart des techniques Méta-Heuristiques. Par conséquent, le CS original a été adapté et modifié, dans le cadre de cette thèse, pour résoudre des problèmes discrets d'optimisation de conception structurelles, et il est nommé Algorithme de recherche Cuckoo CS adapté (Adapted CS Algorithm, ADCSA). L’effort d’intensification de l’algorithme CS adapté est amélioré à travers quatre différentes recherches locales de mouvements de permutation, d’échange, bit flip et insertion. L’algorithme CS adapté a été appliqué pour résoudre deux problèmes d’optimisation structurelles différents, et les résultats obtenus ont montré que cet algorithme présente de remarquables performances pour l’optimisation de ce type de problèmes. L’autre algorithme MH qui a l’objet d’une amélioration est la variante d’optimisation par colonies de fourmis du cadre Hyper Cube (Hyper Cube Framework, HCFACO), qui a été sélectionnée en se basant sur les résultats de l’étude comparative de cinq différentes variantes mentionnées précédemment. L’augmentation de l’effort d’intensification du ‘HCFACO’ a été réalisée en intégrant deux recherches locales de mouvement d’insertion et de bit flip. La performance de la version améliorée du ‘HCFACO’ appelée ‘EHCFACO’ a été examinée moyennant la résolution de problèmes de référence d’optimisation de conception structurelle, et les résultats ont montré une performance significative de ce dernier comparativement à la version originale du ‘HCFACO’, ainsi qu’aux cinq autres variantes de l’optimisation par colonies de fourmis.

Concernant les différents cadres d’optimisation de conception structurelle, la revue de littérature a révélé que les deux approches déterministe et probabiliste sont communément utilisées. Néanmoins, le manque d’études portant sur les incertitudes de conception en utilisant l’anti-optimisation est compréhensible en raison du cout élevé associé à l’analyse de la fonction objective. Cette approche se distingue par deux niveaux, un niveau haut consacré à la phase d’optimisation, et un niveau bas qui a pour objectif l’anti-optimisation de la solution optimale obtenue. Durant ces processus d’optimisation et d’anti-optimisation, un nombre important d’appels de la fonction objective est effectué, ce qui peut présenter un défi majeur pour les problèmes d’optimisation structurelles présentant des fonctions objectives couteuses en termes d’effort de calcul. Cependant, un cadre d’incertitude à cout réel a été développé dans la présente étude. Le cout exorbitant associé à l’évaluation de la fonction objective a été contourné en remplaçant la fonction boite-noir (logiciel d’analyse par éléments finis) par un réseau de neurones artificiel (ANN). La procédure proposée a été appliquée pour optimiser un nouveau cas d’étude d’une plaque composite laminé perforée soumise à des incertitudes sur les conditions de charge et la localisation du centre de découpe. Les résultats obtenus ont montré que l'utilisation de la technique des réseaux de neurones artificiels présente une solution fiable à cout réel pour les problèmes d'optimisation de conception structurelle ayant des fonctions objectives avec un cout de calcul considérable.

En plus des axes de recherche spécifiques mentionnés ci-dessus, d'autres pistes sont apparues durant le déroulement des travaux de cette thèse, tels que l'effet de sélection de la population initiale, la représentation de la solution et la génération adaptative d'une nouvelle solution sur la performance des algorithmes Méta- Heuristiques (MHs). Deux nouveaux exemples d'optimisation de conception structurelle ont aussi été développés pour le cas d'une grue avec une poutre en forme en I et une plaque composite laminée perforée. En général, cette thèse a permis d'améliorer notre compréhension de la façon d'aborder la complexité des problèmes d'optimisation de conception structurelle.

Mots clefs: Méta-Heuristiques, $\mathcal{NP}$-problèmes d'optimisation complexes, optimisation de conception structurelle, facteur de charge critique de flambement, grue, plate composite laminée, Hyper-Cube Amélioré, algorithme de recherche adaptative discret Cuckoo, incertitude, Réseaux de Neurones Artificiels.

Hard Optimization of Structural Design Subjected to Buckling Using the Evolutionary Computation Approach

Ali Elmbrok Salem AHMID

ABSTRACT

The optimization techniques aim to reduce the used material in the structure design without violating the imposed design and manufacturing constraints; thus, the materials cost is decreased, and less material is consumed. In addition to the low-cost and positive sustainability impact of Structural Design Optimization (SDO), it promotes the engineers to develop innovative designs for several real-life structural applications.

The previous studies in the domain of SDO gainfully contributed in the way that expanded our knowledge in this specific field. However, this research theme still watches debates on various issues, not only how we could achieve the desired physical features of a structure, but also how we could make this is happening efficiently at a lowest possible computational cost. With this regard, a couple of potential research opportunities have been extracted from the literature, and in-context novel contributions were presented in the current thesis.

The literature of structural design optimization problems reveals that global optimization algorithms or Meta-heuristics (MHs) are the best available techniques that could be used to solve such hard optimization problems. Though, the main challenge confronted the engineer is which available MH fits much better to the structure design problem of his attention. Unfortunately, the literature of SDO experiences a lack of systematic assessment pattern that could help the engineers to overcome this issue. Currently, the commonly used measures of MHs performance by MHs developers are the statistical operators such as min, max, mean and standard deviation of the obtained solutions. However, such measures are not enough to reflect the actual MH performance when these measures are used alone. So, a comprehensive assessment criterion has been developed here to include more efficient measures like the practical reliability, price (computational cost), normalized price, performance rate, solution quality and Fitness-landscape analysis. Additionally, two different convergence rates were imposed to examine the MHs at slow and fast rates. As well as the reproducibility of the numerical experiments results considered within the procedure of MHs assessment. Lastly, the proposed criterion has been employed to compare five different Ant Colony Optimization (ACO) variants. The proposed measures demonstrated a comprehensive assessment of the compared ACOs performance.

Several studies were conducted to improve the MHs searching performance, and the results were promising in this direction. Unfortunately, the literature of SDO demonstrated an extreme tendency to develop new “metaphor” MHs instead of improving the performance of well-established MHs that have a remarkable history of solving Non-Polynomial ($\mathcal{NP}$) -hard optimization problems. However, this thesis examines the possible improvements of two selected MHs searching features via integrating local search movements to MH's main structure to improve the intensification effort. The first selected MH is Cuckoo Search (CS) algorithm, which is intensively used to solve a variety of optimization problems such as weight minimization of the truss structure, Travelling Salesman Problem (TSP),…etc. CS is designed to solve unconstrained continuous optimization problems as most of MHs. Consequently, the original CS has been adapted and modified here to solve discrete SDO problems, and it is named Adapted CS Algorithm (ADCSA). The intensification effort of ADCSA improved through four different local search movements of permutation, swap, bit flip and insertion. ADCSA has been applied to solve two different SDO problems, and the obtained results of both case studies reveal that the proposed ADCSA has a considerable performance in solving SDO problems. The other improved MH was the ACO variant of the Hyper Cube Framework (HCFACO), which was selected based on the results of the comparison study of five different ACO variants that previously mentioned. The enhancement of the HCFACO intensification effort was carried out by integrating two local search movements of insertion and bit flip. The performance of improved version HCFACO, Enhanced HCFACO (EHCFACO), has been examined through solving a well-known SDO benchmarking problem. EHCFACO exhibited a significant performance compared to the original HCFACO and the other five ACO variants.

Regarding the structural design optimization frameworks, the literature review has shown that both deterministic and probabilistic SDO approaches are mostly used. Nevertheless, the scarcity of studies of uncertainty design using anti-optimization is understandable because of the associated expensive design analysis cost of the objective function. This approach has two levels, the top level devoted to the optimization phase, while the bottom level works to anti-optimize the obtained optimal solution. During this process of optimization and anti-optimization, a large number of objective function calls is taking place, and for those SDO problems with expensive functions, this approach becomes unfeasible. However, a cost-effective uncertainty framework has been developed in the current study. The accompanied expensive cost of the objective function evaluation has been tackled by replacing the black-box function (FEA software) by an Artificial Neural Network (ANN). The proposed procedure was applied to optimize a novel case study of a perforated composite laminated plate subjected to the uncertainty of loading conditions and the location of the cut-out center. The attained results reveal that using ANN techniques offers a cost-effective solution for SDO problems with expensive objective functions.

In addition to those specific research opportunities mentioned above, some others appeared during the research process, such as the effect of selection of the initial population, solution representation and adaptive generation of a new solution on the performance of MHs. Moreover, two novel SDO examples have been developed for customized I-beam overhead gantry crane and perforated composite laminated plate. In general, this thesis has gone some way towards enhancing our understanding of how to tackle the complexity of SDO problems.

Keywords: Meta-Heuristics, $\mathcal{NP}$-hard optimization problems, structural design optimization, critical buckling load factor, crane, composite laminate plate, Enhanced Hyper Cube ACO, Adapted Discrete Cuckoo Search Algorithm, uncertainty framework, anti-optimization, Artificial Neural Network.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF ALGORITHMS

Page

LIST OF ABREVIATIONS

ACO	Ant Colony Optimization
ADCSA	Adaptive Discrete Cuckoo Search Algorithm
AI	Artificial intelligence
ANN	Artificial Neural Network
BWACO	Best-Worst ACO
CC	Continues optimized Custom crane
CMAA	Crane Manufacturers Association of America
CO	Constraints
CS	Cuckoo Search
DC	Discrete optimized Custom crane
DF	Design Framework
DOE	Design Of Experiments
DUD	Discrete Uniform Distribution
EACO	Elites Ant Colony Optimization
EHCFACO	Enhanced Hyper Cube Framework ACO
EOT	Electrical Overhead Travelling
ES	Equivalent Standard I-beam
FEA/FEM	Finite Element Analysis/Model
FDC	Fitness-Distance Correlation
GA	Genetic Algorithm
GUI	Graphical User Interface

HCFACO	Hyper Cube Framework ACO
MH	Meta-Heuristic
ML	Machine Learning
MMACO	Max-Min-ACO
OB	Objective
OM	Optimization Method
PSO	Particle Swarm Optimization
PyI	Python Interface
RBACO	Rank Based Ant Colony Optimization
RG	Research Gab
RO	Research Objective
SA	Simulated Annealing
SDA	Structural Design Analysis
SDO	Structural Design Optimization
SSD	Stack Sequence Design
TSP	Travelling Salesman Problem

INTRODUCTION

Nowadays, structural design optimization (SDO) topic continues to attract the attention of research community and the industry due to the obtained benefits in terms of cost-effective structures. The large-scale structures such as airplanes, space shuttles, ships, bridges, …etc., are consuming a significant portion of the global natural resources, and structural design optimization could help in this by decreasing the required consumed materials. Besides of this positive sustainability impact, structural design optimization could save the structurer cost by reducing the designing cost via parametrizing the iterative structure design calculations. Furthermore, structural design optimization derived innovative designs for several structural applications in real-life.

The strategies of the structural design optimization are varying based on the design objective, for example, it could be size optimization, where dimensions of the structure are the design variables that needed to be optimized against certain design constraints. Also, the objective could be topology optimization, where the shape and material distribution need to be optimized. Other factor is the design and manufacturing constraints which have a great influence on the nature of the optimization type of the designated structure problem. For instance, when the design variables of a specific structure belong to a continuous design domain (continuous optimization) it results an easy implementation and better performance solutions. In contrast, when the design variables are belonging to a discrete design domain of individual values (discrete optimization) the complexity of the optimization problem will grow exponentially as much as the number of design variables is increased. Consequently, the computational time to find the optimal solution is growing exponentially until it becomes Non-Polynomial (NP) time, and this turns the structure optimization problem into an NP-hard problem according to the theory of complexity (Marco Dorigo & Stützle, 2019). This class of the optimization problems is common in structural design practice; for example, optimization of composite laminated plate where the fiber orientation is limited to discrete available angles $(0°, \pm 45°, 90°)$. Also, the number of plies is representing the number of design variables to be optimized, and when the number of plies increases the computational problem complexity will increase too.

From the optimization methodology prospective, the classical (mostly gradient) methods are favourable because of their implementation simplicity and good solution quality they offer at a low computational cost. Unfortunately, these techniques performed poorly in solving NP-hard optimization problems where they tend to be stuck in local optimum rather than not finding any optimum. The significant performance of Meta-Heuristics (MHs) compared to gradient optimization algorithms addressed many times in literature due to their efficiency and stability (Ghiasi, Fayazbakhsh, Pasini, & Lessard, 2010). However, MHs are an ongoing optimization research domain to solve medium as well as large-scale problems that appear in different disciplines (Almufti, 2019). Even though MHs, in general, could solve the discrete optimization problems efficiently, we still need to determine which MH well-matched to solve a specific structural design optimization problem according to the No Free Lunch theorem (NFL) by Wolpert and Macready (1997).

Motivation

The literature of structural design optimization is full of valuable contributions that advanced our knowledge in this specific area. However, this subject of research still watches debates on various issues, not only how we could achieve the desired physical features of a structure, but also how we could make this is happening efficiently at a lowest possible computational cost. With this regard, a couple of potential research opportunities that we extracted from the literature are briefly introduced here:

- The literature of structural optimization experiences a scarcity of systematic assessment paradigm that could support the designer to decide what MHs fit better to the structure design problem of his attention. Therefore, the selection of an efficient optimization algorithm needs to develop substantial compromise criteria to determine which MH algorithm offers a cost-effective solution for a designated optimization problem, and it deserves to be selected.

- The literature of NP-hard optimization problems reveals that global optimization algorithms or MHs are the best available techniques that could be used to solve such hard optimization problems. Several studies conducted to improve the MHs' searching performance, and the results were promising in this direction. So, improving the well-known MHs searching capabilities is another opportunity of contribution.

- The widely used option for structure analysis is deploying a multi-purpose commercial FE software such as ANSYS, NASTRAN, and ABAQUS, and this option gives the designers more flexibility to enlarge the complexity of the design of their structures as they need. Unfortunately, when the design problem involves more complexity, the evaluation of the structure analysis will become more expensive too. Therefore, examining other possible alternatives, such as using a surrogate model or Artificial Neural Networks, deserve attention.

- Some commercial FEA software has built-in optimization tools, e.g. Multi-Objective Genetic Algorithm (MOGA) in ANSYS WB. But when it used to solve real-life optimization problems, it becomes useless in terms of solution cost and quality; as we mentioned previously that no one optimization algorithm could solve all optimization problems. Developing an in-house optimization package that fits the design problem needs is a favourable choice. Though, this option involves technical challenges of how to integrate it into the commercial FEA software with no more extra cost. Hence, developing a sort of an open-source interface between the commercial FEA software and in-house optimization package (e.g. MH has written in a Matlab program) representing a practical opportunity.

- Lastly, it has been observed from the literature review of structural design optimization that the uncertainty design optimization procedures were infrequently used, although they produce robust optimal designs. Consequently, scheming an uncertainty optimization framework that could handle expensive structural optimization problems (e.g. thin perforated plate) is another vital opportunity.

Problem Statement

The discrete nature of a structural design optimization problem makes the objective function a multimodal (non-convex) function. As a consequence, the design space will have more than one optimal solution, and this turns the structure design into an NP-hard design optimization problem. The complexity of the problem is continuing to grow when the designated problem has a larger number of design variables that need to be optimized. Furthermore, for those structures without an analytical solution of the design analysis, using the approximated solutions such as a multi-purpose commercial Finite Elements Analysis (FEA) software could result in an expensive objective function evaluation.

However, this thesis considered two different structures as case studies, thin composite laminated plate and a customized I-beam crane to answer the following research questions:

- Which optimization technique is suited best to solve a specific structural design optimization problem?

- Does the candidate optimization technique performance could be improved in the way that makes the obtained optimal solution, by the improved version, is much cheaper with the same quality (or better) of the original one?

- How could we reduce the computational cost of the optimization procedure if the obtained objective function of a particular loading analysis is computationally expensive?

- How could we develop a robust optimization framework that able to handle a structural optimization problem which has an expensive objective function, and subjected to uncertainty influences of loading conditions?

Thesis Objectives

Realizing the prominence of developing efficient designs of engineering structures drove this thesis to attempt bridging the research gaps that addressed in the literature review chapter. Consequently, a list of specific objectives has been established to answer the research questions previously mentioned.

- Extending our knowledge about MHs as optimization techniques for structural design through developing an intensive literature review and determining the possible research gaps or opportunities and propose new solutions to potential existing problems.

- Build and examine a comprehensive comparison criterion based on practical measures to help the designer selecting the right MH that most fit his design problem of interest.

- Investigate possible improvements that possibly improve the candidate MH exploration and exploitation features in the desire to find a new cost-effective solution for a structural design that defeats the solution obtained by the original MH.

- Investigate the using of possible techniques that deal with the expensive computational cost of the design analysis for some structures.

- Finally, developing a robust optimization procedure for an engineering structure that considers the uncertainty of loading conditions. Explicitly, this procedure should be formed in a manner that makes it capable of handling expensive objective functions.

Thesis outline

The thesis is a manuscript-based dissertation written based on four individual articles; however, they are all tied together to answer the research questions. Also, the thesis structure consists of a literature review, thesis framework, conclusion, and appendixes for additional journal paper and supplementary data.

Chapter 1 is devoted to present a synopsis of intensive previous work review in fields of structural optimization, Meta-Heuristics (MH), and recent trends in solving structural optimization problems. Moreover, a discussion of potential research gaps that orient the thesis direction is also presented.

Chapter 2 introduces the research methodology proposed to attain the predefined objectives of the thesis. Then it explains the main components of the developed approach, which corresponds to the individual purposes of the thesis.

Chapter 3 is a journal paper titled "An Optimization Procedure for Overhead Gantry Crane Exposed to Buckling and Yield Criteria," published in the IRA-International Journal of Technology & Engineering (IRAJTE) (2017).

Chapter 4 is a journal paper, titled "An adaptive Discrete Cuckoo Search Algorithm to Solve Structural Engineering Problems," published in the Journal of Multidisciplinary Engineering Science and Technology (JMEST) (2020).

Chapter 5 is a journal paper, entitled "Enhanced Hyper-Cube Framework ACO for structural combinatorial optimization problems. " submitted to Elsevier Computer and Structures (2020).

Chapter 6 is a journal paper titled " Optimization of Perforated Composite Laminated Plate Subjected to Uncertain Geometrical and Loading Conditions " submitted to Elsevier Composite Structures journal (2020).

The conclusion is the last section; it dedicated to introducing the summary of the work achievements and highlight the obtained main contributions of this thesis. Moreover, some alternatives for future work are mentioned. Finally, different publications produced during the current research time were listed at the end of the section for the interested reader.

CHAPTER 1

LITERATURE REVIEW ON STRUCTURAL DESIGN OPTIMIZATION

Structural Design Optimization (SDO) is a recursive process that aims to determine the best possible design (solution) among several feasible designs. The SDO differs from the classical design approach in the formulation of the design problem, where SDO forms the design problem as an optimization problem that has objectives and design constraints. The classical design approach depends on a conceptual design developed based on data collection of the design problem (see Figure 1.1). Furthermore, SDO examines the feasible design violation of specific design constraints, whereas the classical design approach inspects the design satisfaction of the performance criteria. The design updating is another difference between both approaches, in classical design occurs based on the designer experience and heuristics information while SDO updates the design based on optimization concept,(Singh, 2017).

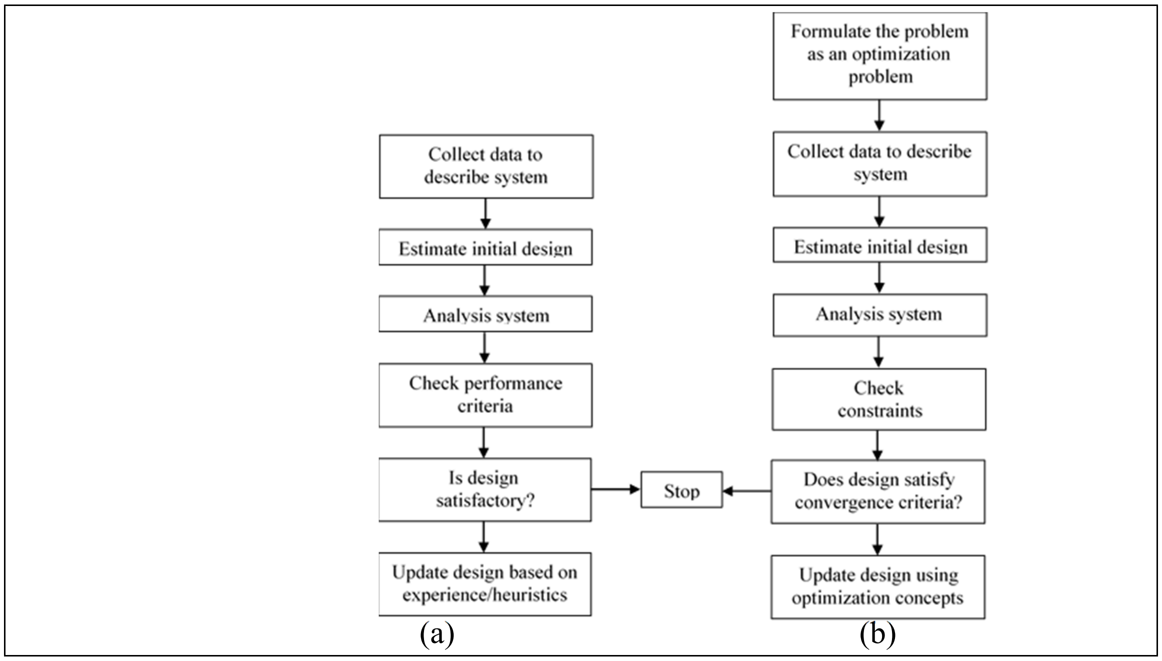

Collect data to describe system
Estimate initial design
Analysis system
Check performance criteria
Is design satisfactory?
Update design based on experience/heuristics
Stop
Formulate the problem as an optimization problem
Collect data to describe system
Estimate initial design
Analysis system
Check constraints
Does design satisfy convergence criteria?
Update design using optimization concepts
(a)
(b)

Figure 1.1 (a) Classical structural design approach, (b) Structural Design Optimization (SDO) Taken from Singh (2017)

The literature of the SDO is rich of benchmarking problems; for example, Cohn and Dinovitzer (1994) accounted about 500 published benchmarking SDO examples regarding the structure type and optimization method. More recently, Clune (2013) reported that 55% of benchmarking examples of SDO considered in the literature were truss and frames structures. Still, there is a growing trend to optimize other structures such as beams, plates, columns and composites. However, the current chapter is devoted to reviewing the recent trends in SDO benchmarking problems, optimization methods and structural design frameworks. Accordingly, the most relevant studies published in the last decades have been discussed here and summarized based on the literature indicators illustrated in Figure 1.2.

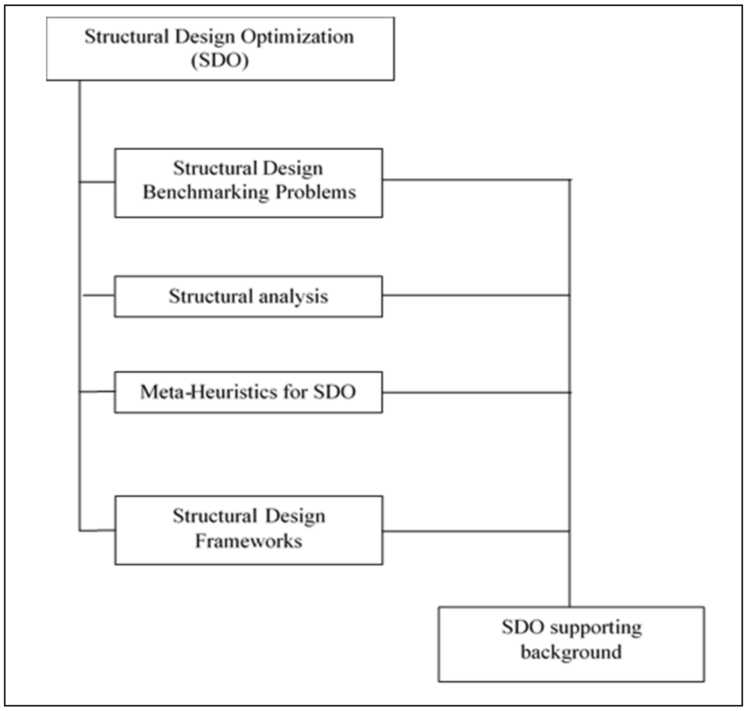

Structural Design Optimization (SDO)
Structural Design Benchmarking Problems
Structural analysis
Meta-Heuristics for SDO
Structural Design Frameworks
SDO supporting background

Figure 1.2 SDO literature review indicators

1.1 Structural Design Benchmarking Problems

SDO started the early 1900s years when Michell, in 1904, presented a formal structural optimization procedure through his well-known benchmark problem of weight minimization of the truss structure. However, SDO research watched an increased interest since the 1960s, and by the mid-1990s were around 2500 published articles and more than 150 books in the SDO domain, and since then, an increasing volume of published studies has been noticed, (Clune, 2013). The main purpose of SDO benchmarking problems is verifying the performance of new or modified optimization methods. The majority of SDO problems have a multimodal design space as a result of design variables number and complexity of some nonlinear constraints such as maximum stresses, deflection and other design configuration. Consequently, using traditional optimization techniques, e.g. steepest descent algorithm, are not efficient in solving such problems, and only Meta-Heuristics (MHs) could find the global optimal solution for such design optimization problems (Gandomi, Yang, & Alavi, 2013).

The isotropic materials structures such as steel truss, frames, beams, …etc., have been intensively studied for more than a century. Also, anisotropic materials structures such as composite laminated plates grabbed the attention on account of their significant characteristics. A subset of the benchmarking structure problems for both materials types is considered here to be reviewed, including a space truss structures, I-beam, and composite plate. These benchmarks are discussed in the following sections to explain the SDO problem formulation and some obtained results in the literature.

1.1.1 Truss Design Optimization Problems

Usually, truss structures optimized for minimum weight by reducing the elements' cross-section area. These benchmarking problems used intensively to evaluate the performance of optimization techniques. The truss structure always optimized with different design constraints such as stress, deflection and buckling constraints (Gandomi & Yang, 2011). They are known for their complexity due to the high number of design variables, and some of them have no global optimal solution yet (Clune, 2013). Thus, many studies have been conducted to solve these problems using new or modified MHs. Several types of truss structures were employed in the literature; for instance, Sadollah, Eskandar, Bahreininejad, and Kim (2015) examined the performance of three different MHs, Mine Blast Algorithm (MBA), Water Cycle Algorithm (WCA) and Improved MBA (IMBA). The proposed MHs applied to four different structures of 52,72, 200 and 582-bar space trusses, which yield a range of design variables from 12 to 96 that need to be optimized for a discrete design domain. Ho-Huu, Nguyen-Thoi, Vo-Duy, and Nguyen-Trang (2016) proposed a modified Differential Evolution MH called (aeDE) that uses an adaptive technique that balances the exploration and exploitation features of the original DE; also, it uses the elitist strategy to select the best individual for the next generation. Then, aeDE applied for six different discrete truss examples and the rounding to the nearest value used to solve the discrete design domain problem. Mirjalili and Lewis (2016) examined the new MH of Whale Optimization Algorithm (WOA) using six different structural benchmarks of spring, welded beam, pressure vessel and three truss structures of 15, 25 and 52 bar truss. Other MH called Fireworks Algorithm (FWA) was introduced by Gholizadeh and Milany (2018) to solve discrete structural design optimization problems. Four different benchmarks of truss and frame structures were optimized using FWA and the results compared to other results in the literature.

However, the truss structures continue to challenge the MH developers as a consequence of their objective function complexity and a large number of design variables. Hence, the optimization problem of truss structures could be generally formulated as follow:

a.) Objective Function

$$w(A) = \sum_{i=1}^{n} \rho_i A_i L_i \tag{1.1}$$

where $w(A)$ is the structure weight; n is the number of structure members; ρ_i is the density of the structure member; A_i is the member cross-section area and L_i is the structure member length.

b.) Constraints

The truss elements are subjected to compression or tension loading conditions that generate the equivalent stresses. The elements' strength of tension and compression should remain under the allowable stress values. Accordingly, the stress constraints could be written as follow:

$$\sigma_{t_i} \leq \sigma_{t_{allwable}}, i = 1,2,\dots,n_t$$
$$\sigma_{c_i} \geq \sigma_{c_{allwable}}, i = 1,2,\dots,n_c$$

where n_c, n_t denote the number of truss elements subjected to compression and tension loading, respectively. Each node in the truss structure (element jointing point) is exposed to local displacement known as node deflection, and the optimal solution should not violate the maximum value of the node deflection, $\delta_{j,max}$, and the deflection constraint can be expressed as follow:

$$\delta_j \leq \delta_{j,max} \,, j = 1,2, \ldots., n_{nodes}$$

The excessive compression can buckle the truss element and lead to a buckling failure mode where the maximum applied stresses at the failure point are higher than the bearing capacity of the element. To avoid such failure mode, the SDO procedure should determine the buckling element stress, σ_{b_k}, to be less than permitted buckling stress and this yields:

$$\sigma_{b_k} \leq \sigma_b \,, \;\; k = 1,2, \ldots..n_c$$

Besides, the cross-sectional area A of the member should be selected from a specific range, and this range could be continuous (upper and lower values) or discrete from specific individual values. So,

$$A_i \;\in\; [A_{min}, A_{max}\,]$$

or

$$A_i \;\in\; [A_1, A_2, \ldots \ldots.., A_k], \text{ for discrete SDO.}$$

This optimization formulation assumed that the material is homogenous isotropic material of all structure elements, and thus the mechanical properties of the elements such as density, ρ, and young's module, E, are constant all over the optimization procedure.

c.) Objective Function Transformation

The optimization techniques designed to solve unconstrained continuous optimization problems. The real-life optimization problems, including SDO, have their constraints that need to be unviolated, and for this, different methods of handling the constraints were proposed. The commonly used approach in SDO is the penalty functions approach, and it has different formulation forms, here the exterior point penalty function is used to explain how the objective function transformed to include the design constraints, (Saka, Hasançebi, & Geem, 2016). The general form of the transformed objective function using an exterior point penalty function is:

$$F(X, r_h, r_g) = f(X) + r_h \left[\sum_{k=1}^{i} h_k(X)^2 \right] + r_g \left[\sum_{j=1}^{m} (max\{0, g_j(X)\})^2 \right] \tag{1.2}$$

where X is the vector representing the design variables, h_k is the k^{th} equality constraint if any, g_j is the j^{th} inequality constraint, r_h and r_g are two additional variables called penalty multipliers.

Eventually, to review and compare the results of SDO truss benchmark problem solved by different MHs, an example of 25-bar space truss has been selected from the literature and its different obtained results were listed in Table1.1 and also plotted in Figure 1.4.

Example: 25-bar space truss

This benchmark problem is commonly considered in the literature, and it has two variants, a continuous and discrete design domain (Mirjalili & Lewis, 2016).

Figure 1.3 shows a 25-bar space truss with four fixed nodes (foundations) and the truss members are classified to eight design groups, D_g, based on their cross-sectional areas as follow:$D_{g_1} = \{A_1\}$; $D_{g_2} = \{\mathrm{A}_2, \mathrm{A}_3, \mathrm{A}_4, \mathrm{A}_5\}$; $D_{g_3} = \{\mathrm{A}_6, \mathrm{A}_7, \mathrm{A}_8, \mathrm{A}_9\}$; $D_{g_4} = \{\mathrm{A}_{10}, \mathrm{A}_{11}\}$; $D_{g_5} = \{\mathrm{A}_{12}, \mathrm{A}_{13}\}$;$D_{g_6} = \{\mathrm{A}_{14}, \mathrm{A}_{15}, \mathrm{A}_{16}, \mathrm{A}_{17}\}$;$D_{g_7} = \{\mathrm{A}_{18}, \mathrm{A}_{19}, \mathrm{A}_{20}, A_{21}\}$;$D_{g_8} = \{\mathrm{A}_{22}, \mathrm{A}_{23}, \mathrm{A}_{24}, \mathrm{A}_{25}\}$. Accordingly, the problem has eight design variables that need to be determined to find the minimum truss weight. The truss material density, ρ, is $0.1\ (lb/in^3)$ and elasticity's module modulus $E = 10{,}000\ (ksi)$ while the member cross-sectional area$A_i \in [0.1{:}\,0.1{:}\,3.4]\ (in^2)$. The maximum node displacement $\delta_{max} = 2\ (in)$ and the maximum allowable compression stresses for each truss member group, D_{g_i} , is:

$$\sigma_{Comp_a} = \{35.092, 11.59, 17.305, 35.092, 35.092, 6.759, 6.959, 11.082\}\ (ksi)$$

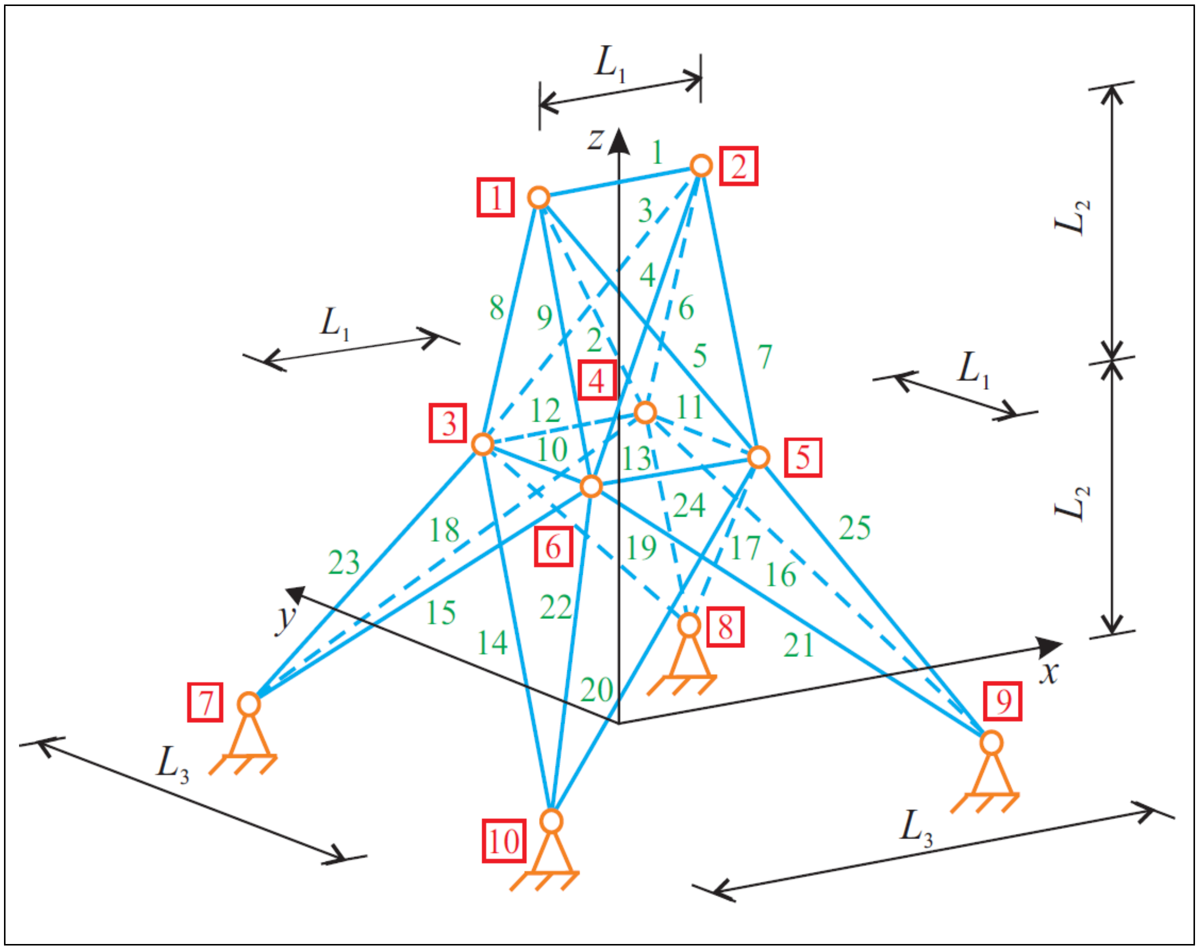


Figure 1.3 Twenty-five bar spatial truss
Taken from Seripk (2020, p.7)

and the allowable member tension stress is $\sigma_{Ten_a} = 40\ (ksi)$. The problem solved using different optimization methods by several authors such as Degertekin (2012), Degertekin and Hayalioglu (2013), Talatahari, Kheirollahi, Farahmandpour, and Gandomi (2013), A Kaveh, Bakhshpoori, and Afshari (2014), A Kaveh and Bakhshpoori (2016), Krempser, Bernardino, Barbosa, and Lemonge (2017), de Castro Lemonge, Duarte, and da Fonseca (2019) and summary of their results are listed in Table1.1. The result of reviewed studies of 25-bar truss for average weight, the standard deviation of the solution and number of function evaluations are plotted in Figure 1.4 . Lastly, a summary of different truss structures optimization solutions is listed in Table 1.2

Table1.1 Comparison of different published results of 25-bar space truss, weight minimization

Author/s	**MH**	**Optimal weight (lb)**	**No. of function evaluation**	**Average**	**Worst**	**SD**	**Termination criterion**
Degertekin (2012)	SAHS[1]	545.12	9488	545.38	546.60	0.91	max number of iterations # of exp=20
Degertekin & Hayalioglu (2013)	TLBO[2]	545.09	15,318	545.38	545.41	0.42	
Talatahari et al. (2013)	MSPSO[3]	545.16	12500	546.03	548.78	0.8	Max number of iteration
Kaveh et al. (2014)	HPSSO[4]	545.164	13,326	545.556	546.990	0.432	NA
Kaveh and Bakhshpoori (2016)	WEO[5]	545.166	19,750	545.226	545.592	0.083	Max number of iteration
Lemonge et al. (2019)	ABC	545.242188	52200	545.416	545.477	0.062	NA /# of exp=25

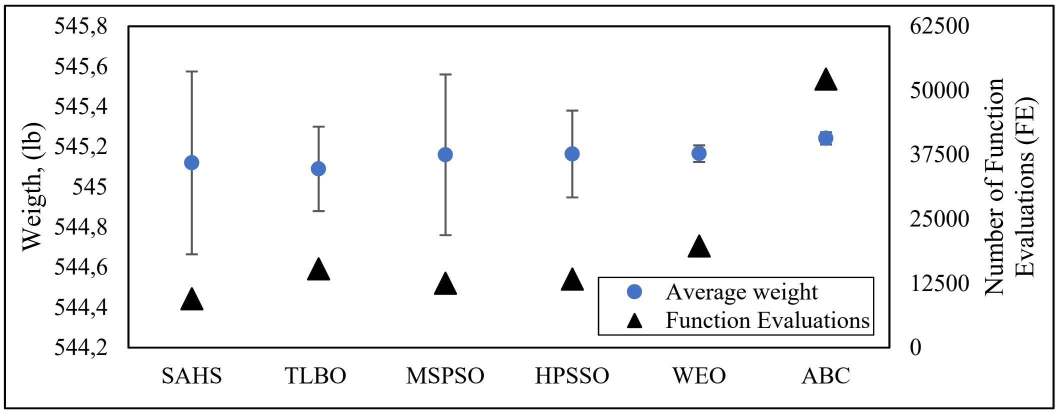


Figure 1.4 Results comparison of average weight and number of Function evaluations for the 25-bar truss

[1] SAHS : Self-Adaptive Harmony Search Algorithm
[2] TLBO : Teaching-Learning-Based Optimization
[3] MSPSO : Multi-Stage Particle Swarm Optimization
[4] HPSSO : Hybrid Particle Swarm and Swallow Swarm Optimization
[5] WEO : Water Evaporation Optimization

Table 1.2 Truss structure optimization problems summary

SDO Problem	Modelling	Domain	MH	Comments
10,25,37,52 and 72-bar space truss by Kaveh & Zolghadr (2010)	OB[6]: Weight minimization. CO[7]: Multi frequency constraints.	Continues	CSS ECSS[8]	ECSS produced better results
304- and 132-members frame structures by Hasançebi et al. (2010)	OB: Weight minimization. CO: stress constraints.	Discrete	Improved SA	
welded beam and pressure vessel by Kaveh & Talatahri (2010)	OB: cost minimization. CO: stress constraints.	Continues	CSS	Performed better compared to previous MHs results e.g. GA and SA
10,18,25,72 and 200-bar space structure by Sonmez (2011)	OB: weight minimization. CO: stress constraints	Continues	ABC-AP	Used adaptive penalty function
10,25, 72 and 200-bar space truss by Degertekin (2012)	OB: Weight minimization. CO: stress constraints.	Continues	EHS[9] SAHS[10]	EHS has improved local search feature SAHS used new probabilistic method to determine the new feasible solution.
10,25, 72 and 200-bar space truss by Degertekin & Hayalioglu (2013)	OB: Weight minimization. CO: stress constraints.	Continues	TLBO	It needs higher number of objective function evaluations.

[6] OB : Optimization oBjective
[7] CO : COnstarints
[8] ECSS : Enhanced Charged System Search
[9] EHS : Effective Harmony Search
[10] SAHS : Self Adaptive Harmony Search

Table 1.2 Continued

SDO Problem	Modelling	Domain	MH	Comments
25,22,72 and 120-bar space truss by Talatahari et al. (2013)	OB: Weight minimization. CO: stress constraints.	Continues	MSPSO	MSPSO performed better than standard PSO
25,22,72,120 and 200-bar space truss by Kaveh & Bakhshpoori (2014)	OB: Weight minimization. CO: stress constraints.	Continues	HPSSO	HPSSO exhibited better performance compared to PSO
10,22,25,72,120 and 200-bar by Kaveh & Bakhshpoori (2016)	OB: Weight minimization. CO: stress constraints.	Continues	WEO	WEO has expensive solution
10,25,52,72,160 and 200-bar space truss de Castro Lemonge et al. (2019)	OB: Weight minimization. CO: stress constraints.	Discrete & Continues	ABC-APF	Handled the discrete domain using penalty function
25,72,200 and 582-bar space truss by Mortazavi, Toğan, and Moloodpoor (2019),	OB: Weight minimization. CO: stress constraints.	Continues	ISA[11]	Using FEM to determine the truss weight.
25 and 72-bar space truss by Yuan, Lv et al. (2020)	OB: Weight minimization. CO: stress constraints.	Discrete	CFSSDA[12]	Handling the problem discreteness is not clear
25,72 and 200-bar space truss by Jahangiri, Hadianfard et al. (2020)	OB: Weight minimization. CO: stress constraints.	Discrete	IAS[13]	Using rounding to nearest discrete value

[11] ISA : Interactive search Algorithm
[12] CFSSDA : Coulomb Force Search Strategy Dragonfly Algorithm
[13] IAS : Interactive Autodidactic School

1.1.2 I-Beam Design Optimization Problem

I-beam profile sections are popular in the structural design applications for trusses, frames, cranes or even more sophisticated applications such as front car axles (Yuan, Lv, Wang, & Song, 2020). The original I-beam benchmark SDO problem developed by Gold and Krishnamurty (1997) and an updated version was introduced by Gandomi et al. (2013), which becomes more used in recent years (see Figure 1.5). The objective of this problem is minimizing the vertical deflection of the beam subjected to axial loading and cross-sectional constraints. Yadav and Arora (2019) applied a new MH known as Artificial Electric Field Algorithm (AEFA) for the I-beam benchmark, and it compared the results reported by Pan (Ye & Pan, 2017), and (Wang, 2003). The proposed AEFA exhibits a better solution with a fast convergence rate.

The problem formulation of the updated version of I-beam benchmark proposed by Gandomi et al. (2013) is shown here:

a.) Objective Function

$$min.\ f(x) = \frac{PL^3}{48EI} \tag{1.3}$$

The beam length $L = 5.2\ m$, and elasticity module $E = 523{,}104 \frac{kN}{cm^2}$; this yields that objective function of maximum deflection could be formed as:

$$f(b, h, t_w, t_f) = \frac{5000}{\frac{t_w(h-2t_f)^3}{12} + \frac{bt_f^3}{6} + 2bt_f\left(\frac{h-t_f}{2}\right)^2} \tag{1.4}$$

b.) Constraints

The beam is subjected to area constraint $A\left(b, h, t_w, t_f\right)$ and to allowable bending stress constraint $\sigma_a = 56\, KN/cm^2$, thus:

$$A\left(b, h, t_w, t_f\right) \leq 300\ cm^2$$
$$\sigma_b \leq \sigma_a$$

The I-beam cross-sectional dimensions are limited to upper and lower values as follow:

$$h \in [10,80]cm$$
$$b \in [10,50]cm$$
$$t_w \in [0.9,5]cm$$
$$t_f \in [0.9,5]cm$$

The objective function transformation of this problem not mentioned, but the exterior penalty function in Eq.(1.1) is working too.

Table 1.3 Comparison of different published results of I-beam optimization problem

Author/s	**MH**	b (cm)	h (cm)	t_w (cm)	t_f (cm)	δ_{max} (cm)
Wang (2003)	GWO	80	50	0.9	2.32	0.0131
	ARSM	80	37.05	1.71	2.31	0.0157
	IARSM	79.99	48.42	0.9	2.4	0.0131
Gandomi et al. (2013)	CS	80	50	0.9	2.32	0.0131
Ye & Pan (2017)	EMGO-FCR	80	50	0.9	2.32	0.0131
Anita et al. (2020)	AEFA-C	79.97	50	0.9	2.32	0.0131

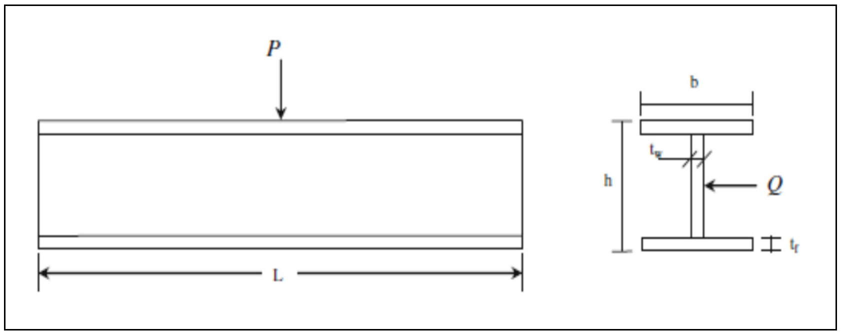


Figure 1.5 I-Beam design problem
Taken form Gandomi & Yang (2013, p.22)

1.1.3 Composite Laminated Plate Design Optimization Problem

The composite materials (anisotropic) continue to attract attention due to their shared property of high specific strength (or strength/weight ratio), thermal stability, corrosion resistance, and their high impact strength. Composites combined two or more materials that have different properties to form a new material with new outperforming properties compared to those of each material alone. These properties can be tailored for specific product requirements, whether it is for extreme temperatures (hot or cold), stiffens, corrosion, fatigue, or other working conditions. Thus, many industrial disciplines, such as aerospace, automotive, marine, construction, and others, demonstrated a great interest in deploying the composite materials in their products. For instance, in the aerospace industry, the aluminum alloys were the dominating material choice for structures of airplanes for decades because of their lightweight, mechanical properties and low cost of production compared to other metal alternatives. But this dominance did not last long when composite materials became a favoured material choice for structural airplane design (Kaw, 2006). However, the design of the composite structure involves a significant complexity because of the diversity of available matrix-fibre materials, manufacturing constraints or imperfections, ply thickness, the number of plies, plies contiguity, and variation of possible stacking sequence configurations. Besides, other design variables such as structure surface topology, edge boundary conditions, and loading type have a considerable impact on the composites design. The demand for a low-cost product with high performance while respecting predefined design criteria (e.g. strength, strain limits, critical buckling load, and materials) is common practice in the structural design.

A review study conducted by Nikbakt, Kamarian, and Shakeri (2018), which covered the published composite optimization articles since 2000, concluded that the maximization buckling load, fundamental frequency and weight minimization are the most addressed objective functions. The reviewed papers classified based on their objective function and simple counting of published papers number for each classification revealed that weight and buckling optimization studies occupied nearly 50%. In comparison, the fundamental frequency has 19.5% of the total number of 347 reviewed papers. The observed interest of optimizing for buckling and weight objectives could be interpreted by the demand on a lighter structure that could bear high buckling load in seek of structure stability, e.g. marine and aerospace structures. In the aerospace industry, thin composite plates are desirable for their lightweight and tailored strength capacity under different applied loads. The failure due to excessive compression loading is a common design mode in awareness of aero-structure designers, and as a result of this, maximizing the critical buckling load of thin plates becomes a vital objective of the design process (de Faria, 2002; Wu, 2020). To date, various studies have been intended to maximize the critical buckling load for different design variables; however, the fiber orientation of the layer is mostly considered a design variable (R. Le Riche & Haftka, 1993; Nikbakt et al., 2018).

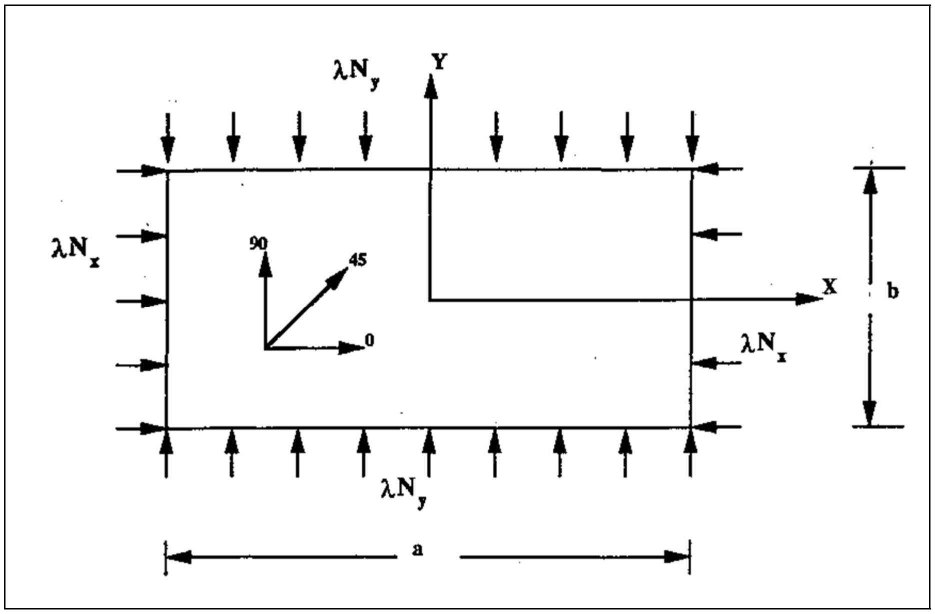


Figure 1.6 Simply supported plate subjected to biaxial loading
Taken from Le Riche and Haftak (1993, p.951)

Usually, the fiber orientation of the ply is limited to a set of available fiber angles, which turns the optimization problem design space into a discrete one; and when the number of layers increases, the number of design variables is increasing too. Consequently, the computational time to find the optimal solution is exponentially growing, until it becomes Non-Polynomial time, and this turns the optimization of the composite laminated plate to an NP-hard optimization problem according to the theory of complexity (Marco Dorigo & Stützle, 2019).

However, the formulation of a composite laminated plate optimization problem, Figure 1.6, for maximum buckling load could be explained as follow:

a.) Objective Function

$$\max(\lambda_{cr}(p,q)) \tag{1.5}$$

where λ_{cb} is the critical buckling load factor that buckles a simply supported plate subjected to in-plane loads of λN_X and λN_y into p and q half-waves in x, y directions. Buckling load factor, $\lambda_{b,}$ could be defined according to Classical Laminated Plate Theory (CLPT) as follow:

$$\lambda_b(\mathrm{p},\mathrm{q}) = \pi^2 \frac{\left[D_{11}\left({}^{p}/_{a}\right)^4 + 2(D_{12} + 2D_{66})\left({}^{p}/_{a}\right)^2 + D_{22}\left({}^{q}/_{b}\right)^4\right]}{\left({}^{p}/_{a}\right)^2 N_x + \left({}^{q}/_{b}\right)^2 N_y} \tag{1.6}$$

The smallest value of $\lambda_b(p,q)$ is considered the critical buckling load factor. The critical values of p and q are linked to different factors such as laminate material, a number of plies, loading conditions, and the plate aspect ratio. In uniaxial loading of a simply supported plate, the critical buckling load occurs when $p = 1$ whereas in biaxial critical buckling loads, it needs to be determined as the minimum value of $\lambda_{cb}(p,q)$ (Reddy, 2004).

b.) Constraints

The design optimization scheme of the composite plates should respect certain limitations of manufacturing and specific design considerations. In literature, some rules have been proposed to improve the effectiveness of a laminate design for different applications (Peeters & Abdalla, 2017; Rama Mohan Rao, 2009; Zein, Madhavan, Dumas, Ravier, & Yague, 2016)

The design and manufacturing rules could be concise as follow:

- *Manufacturing limitations*: the thickness of the plies and fiber orientations are limited to the available manufactured values, which are usually integer, for ply thickness or certain angles such as ±45°, 0°, and 90°and ply orientations. Additionally, the symmetrical laminate makes the manufacturing process more straightforward.

- *Strength and stiffness considerations*: the symmetry of laminate is necessary to prevent extension-bending coupling $\left(B_{(i,j)} = 0\right)$. Furthermore, the balanced laminate (which has pairs of plies with the same thickness and different signs of same orientation angle θ) condition is needed to avoid shear-extension coupling $(A_{16} = A_{26} = 0)$. All the plies with $\pm\theta$ will be grouped to minimize the effect of bending and twisting coupling. Moreover, the congestion of the same orientation plies should be limited to 4 plies for each group to develop a homogeneous laminate and reduce inter-laminate stresses and matrix crack failure.

c.) Objective Function Transformation

Generally, the constraints in stacking sequence optimization with constant laminate thickness t could be integrated easily by considering half of the laminate in optimization for symmetry constraint and group each to plies with the same orientation angle to make balanced laminate. In contrast, the contiguity constraint could be handled by imposing a penalty factor, p, on the value of the objective function violated solutions. Thus, the transformed objective function of this design problem could be written as follow:

$$\lambda_{Opt} = (1 - p) * \max \lambda_{cb}(p, q) \tag{1.7}$$

Lastly, this formulation of the composite SDO problem could be applied for other structures that have a sort of analytical solution; for instance, the same composite laminated plate with different edges conditions (Reddy, 2004). For more complicated structures such as perforated composite plate, finite elements methods could be used to determine λ_{cb} , then evaluate the transformed objective function λ_{Opt} (D. Kumar & Singh, 2012).

Example: Composite laminated plate subjected to bi-directional buckling load

The widespread benchmarking problem used in the literature of stacking sequence optimization is accredited to R. Le Riche and Haftka (1993), and it is widely used by several authors (Deveci, Aydin, & Seçil Artem, 2016; Erdal & Sonmez, 2005; A. Rama Mohan Rao & Shyju, 2008). The original problem describes a simply supported plate subjected to in-plane biaxial loading, as shown in Figure 1.6.

This SDO benchmark examined for different number of plies and loads (see Table 1.4), and the material is graphite-epoxy and the thickness of each ply, $t_p = .127mm$, while the elasticity modulus is $E_1 = 127.59$ GPa , $E_2 = 13.03$ GPa, $G_{12} = 6.41GPa$ and $\upsilon_{12} = 0.3$.

Table 1.4 Different loading conditions of the composite optimization problem

Case	**Plies number** $(\boldsymbol{n_p})$	**Width** $(\boldsymbol{a})$ $(\boldsymbol{mm})$	**Height** $(\boldsymbol{b})$ $(\boldsymbol{mm})$	$\boldsymbol{N_x}$ $(\boldsymbol{N/mm})$	$\boldsymbol{N_y}$ $(\boldsymbol{N/mm})$
1	48	508	127	175	22
2	48	508	127	175	44
3	48	508	127	175	88
4	64	508	254	175	175

1.2 Analytical Versus Finite Element Structural Design Optimization

Some SDO problems have an analytical solution as those we explained in previous sections, for example, the analytical solution of buckling analysis of a thin rectangular plate with different boundary conditions is available for unpunched composite plates (Reddy, 2004). These analytical solutions widely used to demonstrate the various purposes of composites plates SDO, such as showing the performance of MHs, examining the effect of changing materials properties, or used to investigate the impact of hybrid laminate on the buckling load capacity (Awad, 2012; de Almeida, 2016). In a more recent study, A Kaveh, Dadras, and Malek (2019), used the analytical solution to conduct uncertainty optimization of a composite laminated plate subjected to uncertain buckling loads. These solutions are easy to be implemented as an objective function in any optimization procedure, but they are limited for simple cases usually (Abolghasemi, Eipakchi, & Shariati, 2019). The solution of buckling analysis for perforated composite laminated plates is far more complicated than that. Accordingly, many studies considered the optimization problem of perforated composite plate used FEM analysis solution to determine the objective function (D. Kumar & Singh, 2012). The multi-purpose FE software, such as Ansys Workbench or Abaqus, has been intensively employed in the literature. Lakshmi Narayana, Rao, and Kumar (2013) used ANSYS to investigate the effects of cut-out shape on the buckling behaviour of a quasi-isotropic laminated plate. In other individual paper, Narayana Narayana, Rao, and Kumar (2014) extended the previous study to consider the effects of the cut-out size and orientation using ANSYS software. In a study that set out to determine the possible influences of stacking sequence design of hybrid composite plate on the buckling capacity, Dhuban, Karuppanan, Mengal, and Patil (2017) found that generated results by using nonlinear FE analysis of ANSYS have a good agreement with those experimentally obtained. Nevertheless, the availability of the simple solution will reduce the time needed to evaluate the objective function while using the approximated solutions, such as FEM, which could be a very time-consuming process in complex cases (Abolghasemi et al., 2019).

However, the current thesis focuses on solving optimization problems of design structures exposed to buckling criterion. There is no doubt that real-life SDO applications involve different degrees of buckling analysis complexity that intend to use different FEM solutions. The next section will present a concise summary of related FEM concepts and recent trends in determining the critical buckling load factor.

1.2.1 Buckling Analysis using FEM

The main objective of FEM buckling analysis is determining the critical buckling load of the structure. Accordingly, the commercial FEM software, such as ANSYS, offers the users two approaches of buckling analysis; the first is linear buckling analysis (also known as eigenvalue buckling analysis). The second is the nonlinear buckling analysis.

The linear buckling approach uses linear relationships between the stresses and the 2nd order stiffnesses to establish the Eigen value-and- vector equations and solve them for determining the Eigen value which stands for the buckling load factor. On the other hand, nonlinear buckling analysis (also known as post-buckling analysis) uses nonlinear static analysis that considers large deformation to predict the buckling load. Unlike the eigenvalue buckling analysis, the obtained buckling load is given in the applied load value that suddenly produces a large deformation. No doubt that nonlinear buckling analysis gives more accurate results than eigenvalue buckling analysis, but it is computationally expensive (Ansys, 2015). The eigenvalue analysis yields overestimated buckling strength solutions, but it is still widely used in designing real-life structures due to its simplicity and low computational cost (Lê & Champliaud, 2014).

Nevertheless, the predicted values of buckling strength using eigenvalue analysis need to be corrected to include nonlinear behavior of materials, geometric imperfection, and load perturbations to ensure a safe structure design. Before going further, an overview of different ANSYS eigenvalue buckling analysis steps is presented in the following subsections.

a.) Linear Static Analysis

After creating the FE model, the linear static analysis is devoted to calculate the stresses in the structure in three following stages :

The first stage is to calculate the 1st order stiffness matrices, $[K_{1\,e}]$ and the load vectors of body and surface forces, $\{F_{b\,e} + F_{s\,e}\}$ for all elements which are related together by the equilibrium equation (1.8) for each element.

$$[K_{1\,e}]\{D_e\}^? = \{F_{b\,e} + F_{s\,e}\} + \{F_{int\,e}\}^? \qquad (1.8)$$

Where the subscripts $e, b,$ and s indicate element, body, and surface, respectively, the elements degree of freedom $\{D_e\}$ and internal force $\{F_{int\ e}\}$ are unknown at this stage, the question mark "?" has been added for both terms here.

The individual element stiffness matrices and load vectors are assembled into the total equilibrium equation that yields the following form:

$$[K_{1\ t}]\begin{bmatrix}\{D_{unknown}\}^{?}\\ \{D_{restrained}\}\end{bmatrix} = \{F_{b\ t} + F_{s\ t}\} + \begin{bmatrix}\{F_{applied}\}\\ \{F_{reaction}\}^{?}\end{bmatrix} \tag{1.9}$$

where the subscript t stands for total and the applied forces $F_{applied}$ and reactions $F_{reaction}$ represent the external forces.

The next stage is to specify the boundary conditions which consist of all restrained degrees of freedom and all applied forces so that (1.9) can be solved for the remaining parameters ($D_{unknown}$ and $F_{reaction}$).

The final stage of linear static analysis calculate stresses $\{\sigma\}$ using the suitable constitutive formulas such as the displacement-strain transformation matrix $[B]$ and stress-strain material matrix $[C]$, see Eq. (1.10).

$$\{\sigma\} = [C][B]\{D_e\} \tag{1.10}$$

b.) Eigen Buckling Analysis

The stresses given by linear static analysis are used to calculate the 2nd order stiffness matrices $[K_{2\ e}]$ which are function of membrane stress $\{\sigma_m\}$ and structure geometry. The total matrix equation which considers the 2nd order stiffness is:

$$[K_{1\,t} - f * K_{2\,t}]\{D\} = \{F_{a\ t} + F_{ext}\} \tag{1.11}$$

where f is load multiplication factor.

The buckling occurs when the applied load lead to an indefinitely increase in the displacement without increasing the load, which means that $\{\Delta D\} \neq 0$ while $\{\Delta F\} = 0$. Thus, the incremental form of Eq.(1.11) at buckling load level gives the following eigenvalue and eigenvector equations:

$$[K_{1\,t} - f * K_{2\,t}]\{\Delta D\} = \{zeros\} \tag{1.12}$$

Since ΔD is not equal to zero when the buckling occurs, the determinant of the stiffness matrix must be zero :

$$|K_{1\,t} - f * K_{2\,t}| = 0 \tag{1.13}$$

Solving Eq. (1.13) gives eigenvalues which stand for the buckling load factors f, and by substituting the obtained values of f in Eq.(1.12) we could find the buckling modes. The minimum positive value of f is named elastic buckling load factor f_E (Lê & Champliaud, 2014).

1.2.2 Safety Factor of Buckling

Many real-life structures are exposed to failure due to different buckling criteria. For instance, the customized beam structures built by welding plates are subjected to local and lateral buckling. The multiplication elastic buckling load factor f_E that obtained using eigenvalue buckling analysis could be considered as a safety factor against the buckling as long as the membrane compression stresses at critical buckling zones remain less than the yield buckling stress of the structure material. Unfortunately, this is not always the case where the structure may expose to a combination of compression and bending stresses that could exceed the yield stress and steer the structure to fail.

The eigenvalue buckling analysis quickly turns out the results, which keeps it a preferred choice by structure designers. To overcome the overestimating of f_E values, Lê and Champliaud (2014) proposed a correction procedure, and they examined it for the case of welded plates I-beam crane, see Figure 1.7. The procedure steps could be summarized as follow:

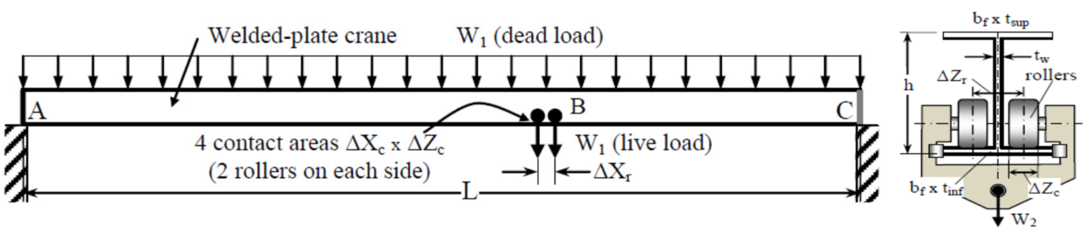


Figure 1.7 Customized beam structure of welded plates
Taken from (Lê & Champliaud, 2014)

Step1: determine the minimum buckling load factor f_E using FEM linear buckling analysis, as explained in section 1.2.1.

Step 2: detect the width of the first buckling mode zone, b.

Step 3: find the value of equivalent linear membrane stress σ_{meqvL} of the buckled width from the results of FEM static analysis.

Step 4: compute the elastic critical buckling stress S_{crE} using the following formula:

$$S_{crE} = f_E * \sigma_{meqvL} \tag{1.14}$$

Step 5: check the obtained buckling stress value, by Eq. (1.14), against the yield buckling stress of the structure.

$$S_{cr} = \begin{cases} S_{crE} & if\ S_{crE} \leq 0.5\, S_y \\ S_y\left(1 - 0.25\dfrac{S_y}{S_{crE}}\right) & otherwise \end{cases} \tag{1.15}$$

When $S_{crE} > 0.5\, S_y$ the S_{cr} should be corrected using the reduction formula shown in Eq.(1.15). This formula was developed based on the original Johnson's parabola expression for short columns, see Figure 1.8.

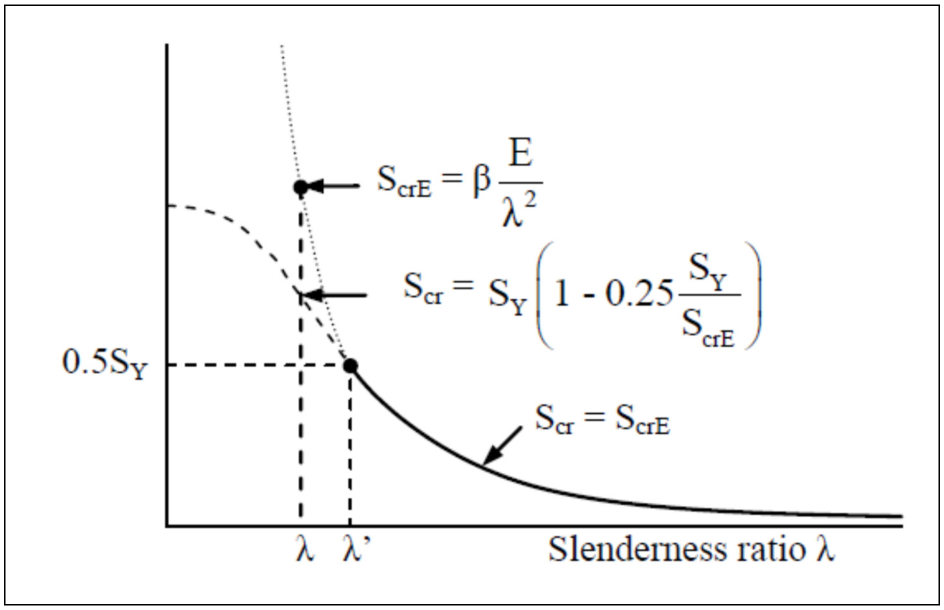


Figure 1.8 Critical stress based on Jonhson's correction.
Taken from (Lê & Champliaud, 2014).

Step 6: determine the factor of safety against buckling load f_{safety} using the following formula:

$$f_{safety} = \frac{S_{cr}}{\sigma_{meqvL}} \tag{1.16}$$

1.3 Meta-Heuristics for SDO

From the optimization methodology perspective, the classical (mostly gradient) methods are favourable because of their simplicity in the implementation and good solution quality at a low computational cost they deliver. Unfortunately, these techniques performed poorly in solving non-convex objective functions where they tend to be stuck in local optimum or even not finding any optimum. These drawbacks of gradient-based methods refer to the assumption of continuity of the problem design variables. In addition, classical techniques need gradient computation of the objective function and constraints, which not possible for most cases of SDO problems where the constraints or objective functions are non-convex functions, see Figure 1.9. Therefore, the majority of recent SDO studies carried out using MH techniques. MH is a stochastic iterative process that follows a particular approach to find feasible solutions in the design space of the problem. MH techniques do not require a gradient computation to find the optimal solution, and for this, they are efficient in solving the combinatorial optimization problems, and this includes SDO problems.

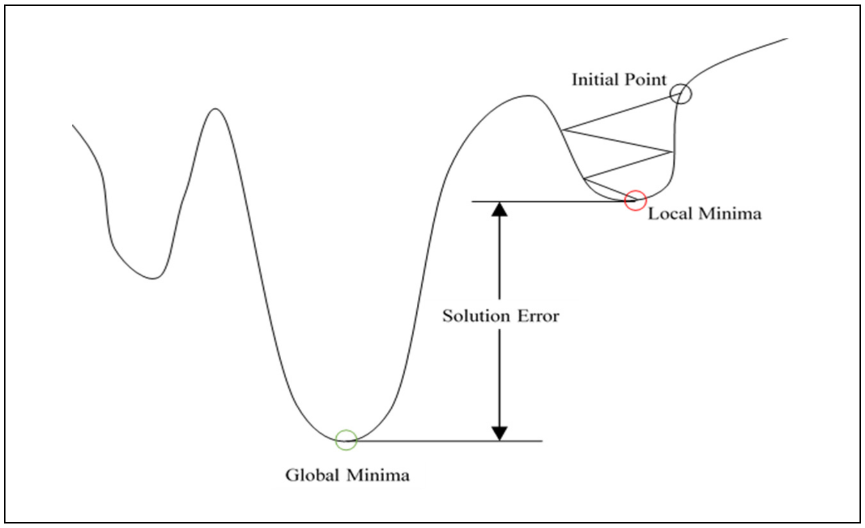


Figure 1.9 Non-convex function example

Many new MHs were developed specially to handle SDO problems such as steel structures or frames in an attempt to present a cost-effective optimal solution (Saka et al., 2016). Ghiasi et al. (2010) reviewed different techniques used in recent decades to optimize composite laminated designs, and he concluded that MHs are superior to gradient-based methods. Furthermore, Nikbakt et al. (2018) reported the outperformance of MHs compared to gradient optimization algorithms due to their efficiency and stability. Also, trajectory based MHs demonstrated a substantial local search capacity on its track to find the optimal solution. In contrast, population based MHs exhibited a significant ability to explore the design space. Even though MHs, in general, could solve the discrete optimization problems efficiently, we still need to determine which algorithm outperforms the others for a specific problem according to the No Free Lunch theorem (NFL) by Wolpert and Macready (1997). In this manner, Mark W. Bloomfield, J. Enrique Herencia, and Paul M. Weaver (2010) conducted a comparison study of three MHs of GA, ACO, and Particle Swarm Optimization (PSO) to determine the optimal stacking sequence composite laminate. Based on the results of this comparison study, ACO found to outperform GA and PSO algorithms in the field of stacking sequence design. This remarkable performance of ACO in solving such NP-hard combinatorial optimization is expected where it designed to solve discrete optimization problems (Marco Dorigo & Stützle, 2019). Here three selected MHs to be reviewed in the following sections for their originality and long history of successful applications. In addition, a brief description of state-of the art MHs, known as metaphor-MHs, is introduced too.

1.3.1 Genetic Algorithm (GA)

Holland suggested the original genetic algorithm in the 1960s, which was later detailed in its generally known form by Goldberg and Holland (1988). It is based on Darwin's theory of natural evolution, and it is implemented using elements of the natural genetics of reproduction, crossover, and mutation. Since then, GA frequently used to tackle a small and large scale SDO problems. The discrete nature of real-life SDO problems challenged the GA that initially developed to solve continuous optimization problems. Thereby, several researchers proposed different approaches; Cheng (2010) proposed a real/integer coded GA to solve large real-life SDO problem of an arch bridge. The optimized bridge design became lighter by 45.5% compared to the weight obtained by the traditional design method. Akshay Kumar and Rangavittal (2019) used GA to minimize the weight of the 25-bar space truss, and they used integer solution representation. Their study results exhibited the efficiency of using integer coding with GA to solve SDO problems. More recently, Xingyu, Jiayi, and Hai (2020) used an improved version of GA to solve different DSO benchmarks, and their results lead to finding a cheaper optimal solution.

Moreover, GA shows its worthiness over classical optimization methods in solving composite SDO problems (Nikbakt et al., 2018). The significant adaptation of GA to optimize composite laminate design is credited to R. Le Riche and Haftka (1993), as he proposed a modified GA that replaces binary coding of solution strings by integer coding. This formulation turned the binary GA algorithm into Permutation Genetic Algorithm (PGA). The results show a 2% reduction in the solution cost compared to binary GA (R. Le Riche & Haftka, 1993). The gene-rank GA introduced by B. Liu, Haftka, Akgün, and Todoroki (2000) is a permutation GA with a gene-rank crossover operator. He compared his proposed GA with standard GA and older permutation GAs, and the gene-rank GA demonstrated better computational performance.

Furthermore, Ehsani and Rezaeepazhand (2016) used binary GA to determine the optimal stacking sequence of grid laminate by considering the different boundary conditions of the laminate edges. Moreover, GA algorithms are known for their expensive solution due to the slow convergence to the optimal solution. To overcome such drawbacks, Vosoughi, Darabi, and Forkhorji (2017) made hybrid GA with PSO algorithms as an operator to increase the convergence rate of standard GA. However, binary GA is still used as a stacking sequence design optimizer. It offers a costly solution, while PGA demonstrates excellent performance for cheaper solutions.

1.3.2 Ant Colony Optimization (ACO)

M. Dorigo (1991) developed the Ant Colony Optimization system that is inspired by the natural phenomena of the food searching strategy of the ant colony. He proposed a mathematical model that simulates this strategy of the cooperative attitude of an actual ant colony to find the optimal solution. He implemented his model to solve well-known optimization problems, such as the travel salesman problem (TSP). The main advantage of ACO is that it designed to solve discrete (combinatorial) optimization problems, basically for TSP, which is common in practice. Thus, ACO has been extended in different engineering areas to solve problems such as the discrete SDO of composite laminated structures (SDO). Aymerich and Serra (2008) investigated the computational efficiency of ACO as an optimizer that maximizes the buckling load of a simply supported plate exposed to uniaxial loading. He compared the solution quality and robustness of ACO with GA and TS algorithms for the same reference case study, and the results show that the ACO algorithm has better performance. Furthermore, Rubem Matimoto Koide, França, and Luersen (2013) used the ACOA combined with finite element analysis to maximize the buckling load factor. They compared the obtained results of their proposed optimization solution with those previously obtained for GA by R. Le Riche and Haftka (1993). The procedure starts with random initial laminate stacking being selected from the feasible solution set (available fiber orientations). This step is followed by an evaluation of the objective function, which will be stored in the ant routing table and used to generate a new feasible stacking sequence. Finally, the global pheromone table is updated where only the ants with the best solution deposited more pheromone trail on their path to the solution. This procedure continues until the termination criterion is satisfied.

1.3.3 Cuckoo Search Algorithm (CS)

Cuckoo Search (CS) algorithm is population-based meta-heuristic inspired by the aggressive reproduction strategy of some cuckoo bird species enhanced by Lèvy flights. It presented by X.-S. Yang and Deb (2009) to solve a variety of continuous multimodal optimization problems. Since then, it attracted attention due to the simplicity of implementation and the fast convergency rate and accuracy of the delivered solutions. Also, CS has a view number of parameters (almost one) to be tuned, compared to other meta-heuristics such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO).... etc. The size of CS applications is fascinatingly growing, where it could be observed through the number of solved optimization problems using CS in the last decade (Shehab, 2020; X.-S. Yang, 2014). Shehab (2020) tracked the progress of published papers that uses CS in the literature. Based on different publishers' metrics, for the CS published articles between 2009 and 2016, he summarized that there are three classes of research interest. The dominant class went to the application, and it represented 67% of publications, whereas CS hybridization and CS modifications had respectively 15% and 18% of the research interest. The applications of CS involve several main optimization problems such as Travelling Salesman Problem (TSP) (Jati & Manurung, 2012; Ouaarab, Ahiod, & Yang, 2014; X.-S. Yang & Deb, 2013; Zhou, Ouyang, & Xie, 2014), and binary optimization problems, for instance, Knapsack optimization problem (Gherboudj, Layeb, & Chikhi, 2012; Layeb, 2011; Xin, Zhang, & Chen, 2019), Computer vision and image detection (Agrawal, Panda, Bhuyan, & Panigrahi, 2013; Loubna, Mohamed, Abdelaziz, & Fatimaezzahra, 2017), Energy sector (de Moura Meneses, da Silva, Nast, Araujo, & Schirru, 2020; Piechocki, Ambroziak, Palkowski, & Redlarski, 2014), supply chain (Q. Li, Liu, & Yang, 2020; Z. Li, Dey, Ashour, & Tang, 2018) and SDO problems (Gandomi et al., 2013; A Kaveh & Bakhshpoori, 2013).

1.3.4 Metaphor Based MHs

The recent two decades have witnessed a massive number of new MHs (see Figure 1.10) that proposed to solve different scale SDO problems. Most of these MHs were mimic a natural or physical phenome, for instance, the water evaporation process. There was some debate about whether the metaphors are novel MHs or just a clone of others. Dennis (2010) argued that the Harmony Search (HS) metaphor is nothing more than a special case of evolution strategies, and it has no transparent mathematical background.

Sörensen (2015) also criticizes the metaphor-based MH research, and he said there are enough "novel" MHs and developing new ones may lead the research area of MHs away from the scientific intent. In contrast, the metaphor-based MHs developers are saying that new metaphor MHs demonstrated remarkable performance in solving hard SDO problems, and they implemented on a different basis of older MHs (Saka et al., 2016).

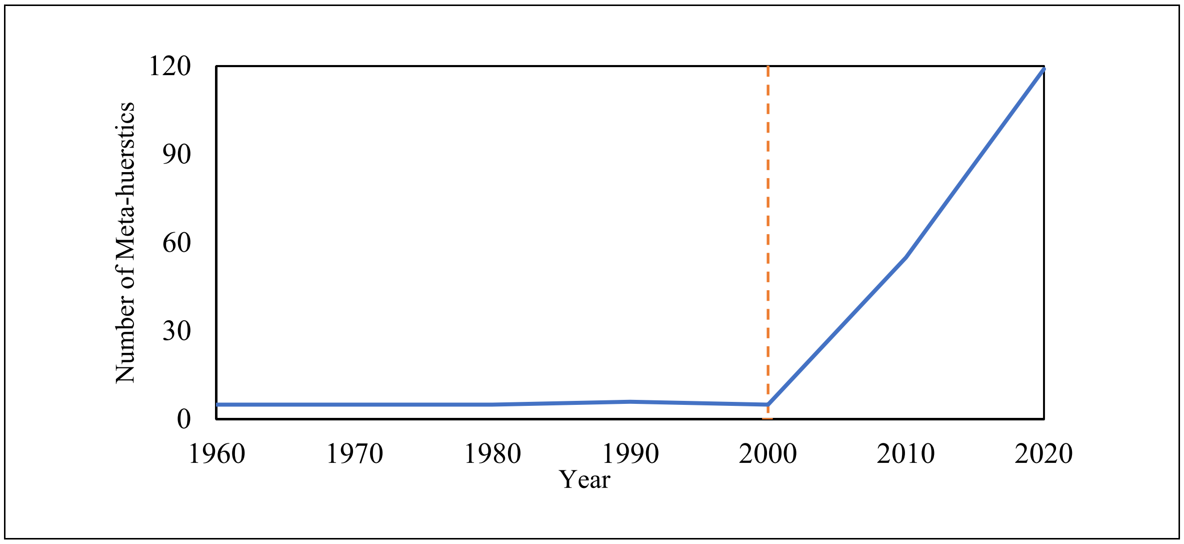


Figure 1.10 Chronologically ordered plot for the number of developed Meta-Heuristics

Table 1.5 Summary of selected metaphor MHs in SDO

MH name	Concept	Applications	Comments
Charged System Search (CSS) by Kaveh & Zolghadr (2010)	Electrostatics and Newtonian mechanics laws	- Truss weight minimization	It appears in 12 published paper for SDO between 2010 to 2020[14]
Artifcial Bee Colony with Adaptive Penalty function (ABC-AP) by Snomez (2011)	Based on intelligent behaviour of honeybee swarm	- Truss weight minimization. - Composite structures.	It appears in 400 (ABC) related paper for SDO between 2010 to 2020
Teaching-Learning-Based Optimization (TLBO) by Degertekin & Hayalioglu (2013)	Based on learning process	- Frame and truss structures. - pin jointed structures. - Composite laminated plate.	It appears in 57 published paper for SDO between 2010 to 2020
Water Evaporation Optimization algorithm (WEO) by Kaveh & Bakhshpoori (2016)	Based on evaporation of water molecules on solid surface	- Truss structures	It appears in 6 published paper for SDO between 2010 to 2020

[14] The number of published papers determined based on Google Scholar search for related article.

In the current study results of some of these metaphors MHs, in solving SDO problems, were shown in Table1.1, and depicted in Figure 1.4. It can be noticed from the figure that some recently MHs developed to solve the same SDO problems. But they delivered a more computationally expensive solution with no significant improvement in the optimal truss weight rather than make it worst, e.g. ABC MH. The description and applications of selected metaphor MHs in SDO, from 2010 up to date, is briefly summarized in the Table 1.5.

1.4 Structural Design Frameworks

The common shared attribute between the previously stated SDO studies is using a deterministic design approach, where the applied load is remaining fixed during the optimization process. Thus, the attained optimal designs are not guaranteed to endure any loading fluctuations, or in other words, the current design will no longer be an optimal solution when the applied load is changed (de Faria, 2002). The deterministic design uses the factors of safety to deal with uncertainty influences (e.g. buckling loads). Still, it can result in an inefficient design that fails to spot one or more failure modes when different failure modes are optimized against the design limits (Clune, 2013). The first alternative to deterministic design optimization is the probabilistic design optimization approach. Unfortunately, probabilistic design optimization is very sensitive to the accuracy and amount of statistics design data. For instance, the scarcity or inaccurate data leads to misfit the probabilistic distribution of different design variables within the uncertainty domains (Lombardi & Haftka, 1998; Qiu & Wang, 2010).

The robust design optimization became the preferred approach by engineers, where it eliminates the uncertainties influences via considering bounded uncertainty domains of the design variables (Isaac Elishakoff & Ohsaki, 2010; A Kaveh et al., 2019). The Anti-optimization method is the common form of robust design optimization used for different SDO problems, also known as a two-level optimization, and it introduced originally by I Elishakoff, Haftka, and Fang (1994). The anti-optimization levels create a nested optimization/anti-optimization loop where the top level is devoted to determining the optimal solution of a given design, while the bottom level is anti-optimize the uncertainty to find the worst scenario case (A Kaveh et al., 2019; Lombardi & Haftka, 1998). I Elishakoff et al. (1994) introduced the anti-optimization approach to include the uncertainty influences in structures design. The proposed approach was formulated to optimize ten-bar structure weight subjected to the uncertainty of loading, stress and displacements. The uncertainty of loading variations was limited to a multi-dimensional box uncertainty domain. Based on their work findings, the authors pointed out that anti-optimization approach overcomes the numerical complexity that accompanied the probabilistic optimization. Venter and Haftka (1996) introduced a two species Genetic Algorithm to reduce the computational effort of GA as an optimizer of two-level problems. They demonstrated the effectiveness of the improved algorithm by solving the anti-optimization problem of a composite laminate plate subjected to in-plane bi-directional compression loading in addition to the uncertain out-of-plane uniform load. The proposed algorithm exhibited a significant saving in computational effort. Adali, Lene, Duvaut, and Chiaruttini (2003) studied the maximization of the critical buckling load of a composite laminated plate subjected to uncertain loading conditions and lamina material type. Both deterministic and robust optimization approaches were examined. The authors concluded that deterministic critical buckling load factor values were less than these obtained by robust optimization approach as a result of different stacking sequence design obtained by both approaches. Jiang, Han, and Liu (2008) developed a method that uses the interval analysis with a hybrid numerical method to compute the transient response bounds of composite laminated plate undergo to load and material properties uncertainties. The influence of different design variables uncertainty was investigated. The transient response bounds acquired by using first-order Taylor expansion together with interval extension. The results imply that the proposed method was confined to a small level of uncertainty applications. On the other side, the method could be extended to solve hybrid composite laminated structures.

1.5 Research Gaps

Even though a considerable amount of literature has been published on the SDO topic, more than 5000 articles by 2013 Clune (2013), the SDO still restricted to limited structures and, to some extent, impractical. Moreover, the developed SDO solutions are mostly focusing on new optimization tools rather than the practical difficulties of SDO problems. In this scene, a couple of potential research gaps that we extracted from the literature are explained here:

- **SDO problem statements, in general, do not reflect the practical difficulties of SDO problems.** The statement of SDO benchmarking problems introduced in the literature does not reflect the real-life complexity of SDO problems. Over decades of using truss and frames structures as benchmarking problems to examine the MHs, they mostly assume that the problem has a continuous design domain while the real-life applications face a discrete one, (Saka et al., 2016). In discrete SDO problems, the design variables' values should be selected from a design vector with individual values. Thus, the Design Space Size (DSS) is growing exponentially and when the number of design variables increases, DSS is increasing too until it becomes hard to be solved within polynomial computational time or NP-hard problem. The majority of MHs developed to solve unconstrained continuous optimization problems. When they used to solve discrete optimization problems, the developers were usually going to use the rounding to the nearest discrete value, which has its drawbacks on the solution cost, as we will demonstrate in this thesis. In this thesis, we introduce two different SDO problems that have applicability in real-life, and we used them to demonstrate various aspects of optimization barriers such as solution representation in discrete design domains and uncertainty influences.

- **Performance assessment of MHs for SDO problems is full of ambiguity.** The literature of SDO experiences a scarcity of systematic assessment paradigm that could support the designer to decide what MHs fit better to the SDO problem of his attention. A significant number of published studies did not mention why they used this specific MH algorithm in their optimization procedures. In the SDO domain, the developers of MHs solutions used statistical measures such as mean, best(max), worst(min) and standard deviation (SD) to demonstrate the performance of their proposed MHs as a solver of truss or frame structures (see Figure 1.4). The question is why the developers of MHs for SDO do not use the other performance measures that successfully used to assess MHs performance in other optimization disciplines, e.g. those used in operations research? Also, it was noticed that in the literature of MHs comparison, there is no common basis in terms of convergence rate or even the maximum number of iterations. Another critical point is the lack of reproducibility of the experimental results of MHs introduced to solve SDO problems. This thesis attempts to develop a sort of comprehensive assessment criteria of MHs performance in solving SDO problems and makes MHs comparison more fairly.

- **The benefits of improving the existing efficient MHs to solve SDO problems does not addressed fairly.** Several studies conducted to improve the MHs searching performance and the results were promising in this direction. Unfortunately, the literature of SDO demonstrated an extreme tendency to develop new metaphor MHs instead of improving the performance of well-established MHs that have a remarkable history of solving NP-hard optimization problems. However, this thesis examines the possible improvements of selected MHs searching features via integrating local search movements to MH's main structure to improve the intensification effort. Moreover, the initial population generation effect on the MH performance has been investigated too.

- **Underestimate the structural analysis difficulties during developing optimization solutions.** The associated high cost of the structural analysis of complex structures addressed in the literature, especially for anisotropic materials. The widely used option is deploying a multi-purpose commercial FE software such as ANSYS, NASTRAN, and ABAQUS. Unfortunately, when the design problem involves more complexity, the evaluation of the design analysis by using this commercial software will become more expensive too. The current thesis uses the cutting-edge methods of complex function modelling to predict the evaluation of expensive SDO objective functions. Further, a kind of an open-source interface between the commercial FEA software of ANSYS WB and in-house optimization package (e.g. MH has written in Matlab program) is introduced to make the using of the improved MHs more practical.

- **Bias to deterministic or probabilistic design approaches over the uncertainty one.** The literature revealed that both deterministic and probabilistic SDO approaches are mostly used in structural design. Nevertheless, the scarcity of studies of uncertainty design, using anti-optimization, is understandable because of the associated expensive design analysis cost of the objective function (G. A. da Silva, Cardoso, & Beck, 2020; Gurav & Goosen, 2005). Consequently, this thesis intends to develop a scheme of an uncertainty optimization framework that could handle expensive SDO problems (e.g. thin perforated plate

CHAPTER 2

METHODOLOGY FRAMEWORK

2.1 The Proposed Research Methodology Framework

There are five specified Research Objectives (RO), mentioned in the introduction section, need to be attained to develop this thesis. The five objectives determined based on the literature review and the observed Research Gaps (RG) in the SDO domain, as explained in the previous chapter. Figure 2.1 outlines the research methodology framework that followed here to bridge the RGs and achieve the thesis ROs. The first RO fulfilled by conducting a comprehensive literature review of solving SDO problems using MHs for the last three decades. Thus, the RGs were determined, and the research problem has been formulated. Then after, the RG1 has been bridged through introducing a novel SDO benchmark problem of customized I-beam profile overhead gantry crane. Besides, a well-known variant of GA MH has been applied to solve this novel SDO benchmark. This work followed by solving the same problem using a new discrete variant of CS optimization called Adaptive Discrete CS (ADCS). This time the design domain of the problem turned to be discrete to make it a more real-world SDO case study and to meet RO3. Next, developing a comprehensive MHs assessment criterion, which represents RO2, has been introduced and applied to five variants of ACO MH. Accordingly, an HCFACO MH has been selected to examine the possible improvement of its local search capabilities to reach the thesis RO3. Finally, another novel SDO benchmark of a perforated composite laminated plate subjected to uncertainty conditions has been introduced to fulfill RO4 and RO5.

For all SDO problems mentioned above, there were different Structural Design Analysis (SDA) solutions developed to evaluate their objective functions as a part of the SDO problem formulation. The type of SDA solution depends on the problem complexity; thus, it could be an analytical solution, FEA or ANN model.

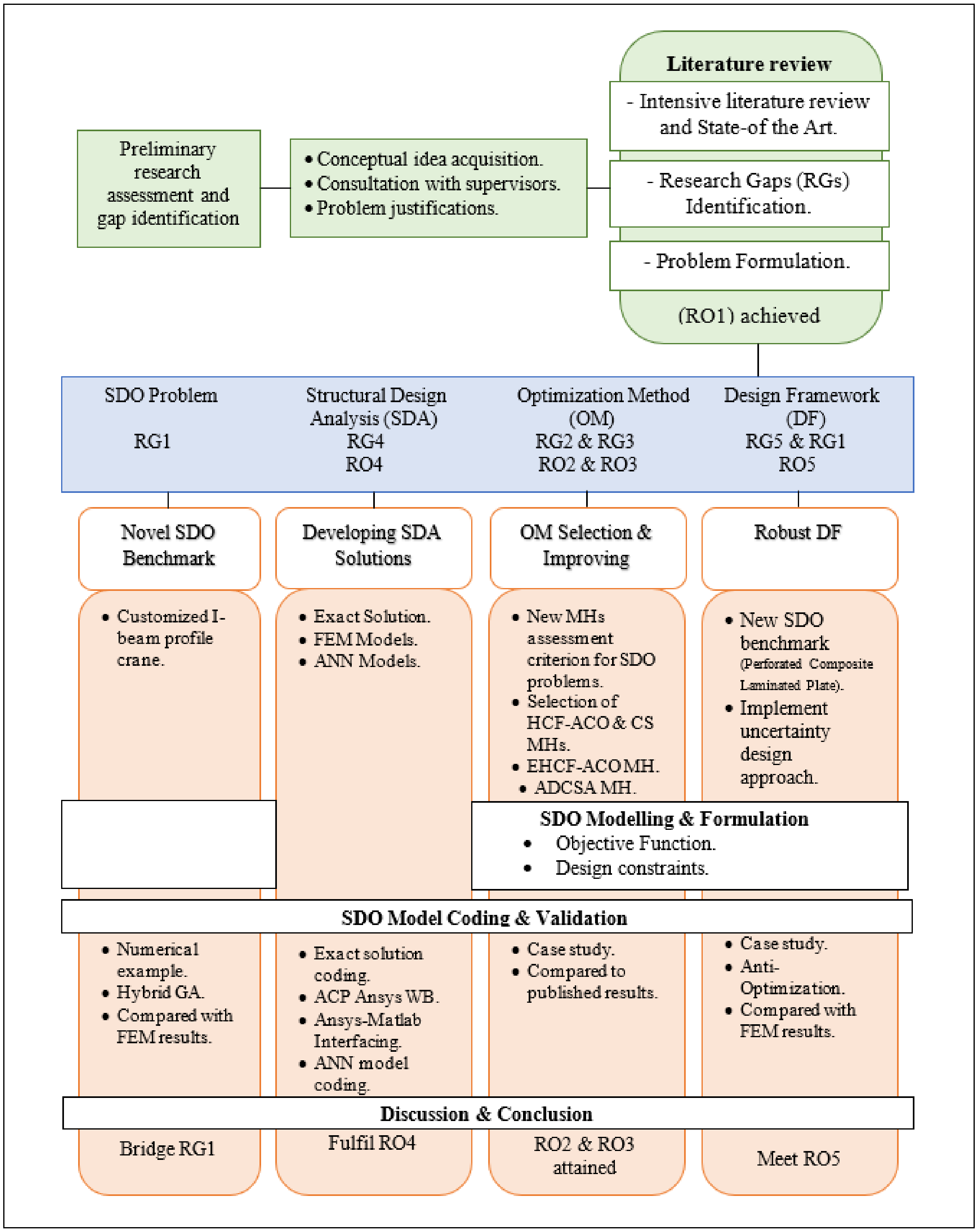

Literature review
- Intensive literature review and State-of the Art.
- Research Gaps (RGs) Identification.
- Problem Formulation.
(RO1) achieved
Preliminary research assessment and gap identification
• Conceptual idea acquisition.
• Consultation with supervisors.
• Problem justifications.
SDO Problem RG1
Structural Design Analysis (SDA) RG4 RO4
Optimization Method (OM) RG2 & RG3 RO2 & RO3
Design Framework (DF) RG5 & RG1 RO5
Novel SDO Benchmark
Developing SDA Solutions
OM Selection & Improving
Robust DF
• Customized I-beam profile crane.
• Exact Solution.
• FEM Models.
• ANN Models.
• New MHs assessment criterion for SDO problems.
• Selection of HCF-ACO & CS MHs.
• EHCF-ACO MH.
• ADCSA MH.
• New SDO benchmark (Perforated Composite Laminated Plate).
• Implement uncertainty design approach.
SDO Modelling & Formulation
• Objective Function.
• Design constraints.
SDO Model Coding & Validation
• Numerical example.
• Hybrid GA.
• Compared with FEM results.
• Exact solution coding.
• ACP Ansys WB.
• Ansys-Matlab Interfacing.
• ANN model coding.
• Case study.
• Compared to published results.
• Case study.
• Anti-Optimization.
• Compared with FEM results.
Discussion & Conclusion
Bridge RG1
Fulfil RO4
RO2 & RO3 attained
Meet RO5

Figure 2.1 Methodological Framework layout

2.2 Novel SDO Benchmark Problem

2.2.1 Purpose and Context

Purpose: Introduce a novel SDO benchmark problem for a real-life isotropic material structure and to apply an MH optimizer to solve this new SDO problem.

Context: The benchmark problems in the SDO domain are generally pure theoretical problems that have no practical application in real life, and they are usually used to evaluate the performance of new MHs. On the other hand, there are some SDO problems for real applications, but they are specific or very complicated to be reused. Here, we proposed a practical case study of customized I-beam profile overhead gantry crane. The previously published articles in crane optimization dealt with the optimization of cranes have standard I-beam profiles, which limits the design to specific profiles configurations. In the customized I-beam profile crane, the designer has a broader range of profile configurations where the crane formed by welding three different plates that have the same span length with different widths and thicknesses. The I-beam profile crane has many applications in building and auto maintenance workshops.

2.3 SDO Problem Modelling and Optimization

a.) Assumptions

The crane built up using a full penetration welding to form an I-beam profile crane using three plates. The material of the plates is 350W structural steel, and it has homogeneous mechanical properties.

b.) Objective Function

Where the span length is fixed with a constant cross-section area of the crane beam, the weight is just proportional to the cross-section area so that the objective function is defined by the cross-section area, as follow:

$$f = b_1 . t_1 + b_2 . t_2 + h . t_3$$

where the parameters b1, t1 …etc. are the width and thicknesses of the bottom, top and web flanges.

c.) Constraints

The essential criteria of the Crane Manufacturers Association of America specification, known as CMAA74-2010, are considered and summarized as follow:

- Tension stress Constraints (due to gravity and live load).
- Lateral Buckling Constraint.
- Local Buckling Constraints.
- Deflection Constraint.
- Fatigue Constraint (due to repeated load fluctuation only).

d.) Input

The crane service class is heavy, and it has three different rated capacities with three different span lengths. The weight of the trolley and other equipment are given too; also, the crane designed to meet the requirements of cycles class N2 for fatigue loading.

e.) Output

The optimal I-beam profile crane configurations for nine different cases have been found using hybrid GA MH. The new configurations have the same characteristics of thick and narrow bottom flange, thin and wider upper flange, and the web is very thin and significantly very wide. The design constraints have been respected in all obtained optimal configurations, but some constraints were critical for certain cases.

2.4 MHs as SDO Problem Optimizer

2.4.1 An Improved CS MH for SDO Problems

2.4.1.1 Purpose and Context

Purpose: Present an improved discrete variant of CS MH to solve the discrete version of the novel SDO benchmark problem. Furthermore, examining the effect of the initial population on the performance of CS MH when it used to solve the SDO problem.

Context: CS is a relatively new developed MH designed to solve unconstrained continuous optimization problems. One of the main features of CS is it has only one parameter to tune known as discovery rate. CS used to solve different NP-hard optimization problems such as TSP, knapsack and SDO problems too. However, several discrete variants were introduced to solve a specific discrete optimization problem, such as Binary CS, to solve the knapsack problem or CS to solve the TSP optimization problem. Other CS variants were more general, such as discrete CS that using rounding of generated solutions into the nearest discrete value in the design domain. This variant of CS used to solve some SDO benchmark problems, and it is implemented here to compare it to the new variant proposed here. In this thesis, a new variant of discrete CS has been proposed and examined as a new MH optimizer of SDO problems. ADCSA is a novel variant of discrete CS that uses rank/value concept to generate step/jumps with integer values to obey Lèvy flights random walk. Furthermore, ADCSA has been used to solve a discrete version of the “ customized I-beam profile overhead gantry crane” SDO problem.

2.4.1.2 SDO Problem Modelling and Optimization

a.) Assumptions

Two SDO problems examined here have homogenous material properties. For the composite laminated plate SDO problem, we assumed that the laminate is symmetrical and balanced with simply supported edges. In the second SDO problem, we assumed that all six design variables have the same length of design domain vector, so it will be easy to permute different ranks of each design variable with the ranks of the other.

b.) Objective Function

The objective of the composite laminated plate subjected to buckling loads is maximining the critical buckling load. The objective function formulation and modelling are mentioned in section1.1.3 in the previous chapter; for the discrete crane SDO problem, the objective function still as described in section 2.3(b).

c.) Constraints

The validation SDO problem of the composite laminated plate has specific fiber orientations to select from, and it assumed to be balanced, which means that we need to consider the two plies with the same orientation as one plies group. Also, the symmetry constraint needs to be enforced by split the laminate into two similar parts. For the crane SDO problem, the same constraints listed in section 2.3 (c) still considered here.

d.) Input

The loading conditions and geometrical information of the validation SDO problem are given. For the discrete version of the crane SDO problem, all the input described in section 2.3 (d) has been used here too.

e.) Output

The optimal stacking sequence design of the composite laminated that maximize the critical buckling load factor has been obtained. Furthermore, the optimal cross-section dimensions of customized I-beam crane have been determined.

f.) Model Validation

The obtained results of the maximum critical buckling load factor of the composite laminated plate, that examined here, shown typical agreement with those published in the literature.

2.4.2 A comprehensive MHs assessment criterion

2.4.2.1 Purpose and Context

Purpose: Implement a comprehensive assessment criterion of MHs performance in solving SDO problems. Furthermore, select the best MH of the compared ACO MHs to examine a further improvement of the local search effort.

Context: The literature review addressed the lack of fair performance assessment measures. The previous comparison studies were interesting, and they still have some drawbacks, such as the diversity of convergence criteria for the compared algorithms or the comparison limitation to one category of meta-heuristics. Moreover, the common performance measures used in the previous studies are statistical measures, such as average, min/max or standard deviation of the objective function. In this thesis, we proposed a conceptual assessment criterion that could be applied to assess the performance of MHs, which may be used to optimize SDO problems. To demonstrate the efficiency of the MHs proposed assessment criterion, an unprecedented comparison of five different ACO variants was carried out. The comparison study revealed that the HCFACO variant demonstrated a promising performance compared to the other ACO family members in the filed of maximizing the critical buckling load of the composite plate. This remarkable performance motivated us to investigate the possible options of improving the local search capabilities of the HCFACO MH.

2.4.2.2 SDO Problem Modelling and Optimization

a.) Assumptions

The assumptions considered here are the same ones mentioned in the previous section for the validation SDO benchmark problem of the composite laminated plate. Furthermore, each ACO variant examined here should be run ten times for ten different seeds numbers.

b.) Objective Function

The modelling of the validation SDO problem remains the same as described in section 1.1.3 in the previous chapter.

c.) Constraints

The design and manufacturing constraints of the SDO benchmark problem enforced in the same manner, followed in section 2.4.1.2 (c).

d.) Input

The buckling load and the geometrical information of the composite laminated plate were given. Ten fixed seed numbers that selected randomly were provided to ensure the reproducing of the obtained results of the assessment criterion.

e.) Output

A comprehensive comparison results of five different ACO MHs have been obtained. The proposed assessment criterion exhibited a distinguished performance in measuring each ACO MHs performance. This work reveals that HCFACO outperforms other ACO family members, and with further improvement, it became a promising MH to solve SDO problems.

2.5 Robust Design Framework of SDO Problems

2.5.1 Purpose and Context

Purpose: Introduce a robust design optimization procedure that could consider the influences of different uncertainty conditions.

Context: The commonly used structural design approach is deterministic design. This approach used successfully for several decades due to its simplicity. The main drawback of the deterministic approach is the excessive consumption of the available resources, e.g. used material, to ensure the safety of the designed structure. Thus, the probabilistic structural design approach appeared as another alternative to the deterministic approach. It proved a significant performance in solving different SDO problems where the probabilistic information of the design variables is available and accurate. The availability and accuracy of the probabilistic data are the main barriers that face this approach. The last alternative in this, it is the robust design optimization procedure where the structure is designed to consider a bounded or predefined loading or design uncertainties. The challenging issue of this approach is the accompanying cost of the structural analysis of the designated SDO problem. Consequently, this thesis intends to develop a scheming of an uncertainty optimization framework that could handle expensive SDO problems (e.g. thin perforated plate).

2.5.2 SDO Problem Modelling and Optimization

a.) Assumptions

The selected SDO problem of the perforated composite laminated plate is exposed to uncertain bounded buckling loading conditions and uncertain position of the circular cut out. The available fiber orientations are limited to standard fiber angles of $0°, \pm 45°$ and $90°$.

b.) Objective Function

The objective function is the critical buckling load factor determined by an ANN model built based on the composite plate analysis results obtained by using Ansys workbench FEA software.

c.) Constraints

The examined SDO problem of the perforated composite laminated plate has to be symmetrical, balanced and simply supported with a certain number of plies.

d.) Input

The bounded buckling loads interval is given, and the range of the circular cut out is specified too. Furthermore, the materials properties and plate dimensions are defined.

e.) Output

A new SDO benchmark problem has been introduced, and its robust stacking sequence design for the given uncertainties conditions has been determined using two different MHs that developed in this thesis.

CHAPTER 3

AN OPTIMIZATION PROCEDURE FOR OVERHEAD GANTRY CRANE EXPOSED TO BUCKLING AND YIELD CRITERIA

A.Ahmid [a] ,V. N. Lê [b] and T. M. Dao [c]

[a,b,c] Department of Mechanical Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper published in *International Journal of Technology & Engineering*, August 2017

3.1 Abstract

The current study presents a general optimization procedure that could be used in designing of various structural applications. To validate the performance of the proposed procedure, a real-life application of a custom welded I-Beam gantry crane is selected. The crane is composed of three rectangular plates with the same length and different thicknesses and widths welded together by full penetration welds over the span length to form an I-Beam profile. The thicknesses and widths of plates are to be optimized to have the minimum cross section area while respecting yield, buckling, deflection and fatigue criteria. A mathematical procedure based on Timoshenko beam theory and Crane Manufacturers Association of America (CMAA) in combination with the Genetic Algorithm (GA) is presented, and a Mathcad code is implemented to find the optimal I-Beam cross section dimensions. Nine examples are introduced for 8, 12 and 20 m crane span subjected to 10, 20 and 40-ton capacities. It is noticed that the optimized I-section configurations always show narrow and thick lower flange, wider and thinner upper flange and tall and very thin web. The upper flange local buckling and the lateral buckling limits are achieved for all nine cases, 75% of cases for the web buckling limit, about 33% of cases for the fatigue and yield limits whereas the maximum deflection constraint is never critical. The obtained results were verified using ANSYS Workbench software with a

3D Solid Finite Element model and shown good agreement, which confirms that the proposed procedure is efficient.

Keywords: optimization; I-beam; yield; buckling; design criteria; finite element.

3.2 Introduction

The gantry cranes are frequently used for different industrial applications. According to CMAA (2010), the cranes in real life engineering are classified into five main classes based on their service capacity: standby, light, moderate, heavy and severe service cranes. The overhead gantry crane type is widely used to serve small or medium duty jobs, like a repair shop, buildings service or in a machine shop. The lightweight crane with high capacity design depicts an essential requirement of the industry. To reach such requirement, a customized I-beam crane is a motivating optimization research. Even though standard I-Beam profiles are available, they are just limited to some standard dimensions, which are usually far from the optimum design. The crane weight, concerning initial standard profiles design, could be reduced up to 10% using an optimized beam (P. Liu, Xing, Liu, & Zheng, 2014).

Several researches have been conducted on optimization of customized and standard crane beams with different profiles. Gąska, Haniszewski, and Margielewicz (2017), developed a numerical model of flange local stresses under the wheels acting points to determine the final dimension of the I-beam girder. The numerical example of 12.5-ton capacity and 25m span with three different wheel thickness demonstrated. The mathematical and FE analysis results compared to show an acceptable error range of 6 to 15 %.Also, they mentioned that the lower flange deflection has a great influence on the final girder dimensions.

Other researchers have worked on optimization of the box profile girders (Ashutosh Kumar & Arakerimath, 2016; Qu, Xu, Fan, & Bi, 2015; Zuberi, Kai, & Zhengxing, 2008); they had, in general, the similar procedure of optimization but they used different optimization tools. Their objective was investigating the same concept of weight-strength ratio using theoretical optimization routines backed up by Finite Element (FE) simulation.

Qu et al. (2015) proposed a modified Ant Colony Optimization (ACO) algorithm with new local search technique using mutation and applied it to solve nonlinear optimization problems having discrete variables. The developed algorithm of Ant Colony Algorithm with Mutation-based (ACAM) used to determine optimal crane design variables and found to be faster by about 20% compared to the genetic algorithm (GA) and by 11% compared to particle swarm algorithm (PSO). Furthermore, it always gives a globally optimized solution, while the original ACO algorithm may stick at some local solution and fail to go further.

Zuberi et al. (2008), examined the effect of rolling load on welded box cross section-crane girder regarding buckling and compression stresses in the flange. The volume of the girder considered as an objective function subjected to the stress and deflection criteria constraints. The built-in MS-Excel nonlinear optimization solver, called Generalized Reduced Gradient (GRG), employed to give preliminary optimized design variables. The obtained values are then used as initial inputs to ANSYS code that can handle more accurate stress and deflection calculations for verification purpose and do further optimization if needed.

Kumar et al. (2016) conducted research that aims to optimize the weight of Electrical Overhead Travelling (EOT) Crane Bridge girder by adding sufficient stiffeners along the girder plate instead of increasing plate thickness. He used mathematical modeling and Finite Element Analysis to investigate the effect of adding stiffeners and then verify the optimal design experimentally. His work concluded that the plate stability could be increased four times using stiffeners without the need to increase the plate thickness.

Liu et al. (2014), carried out a parametric FE study of a doubly trolley box-girder using APDL tool in conjunction with a Matlab code that handles the crane parameters. A three-dimensional girder model subjected to various loading conditions established to predict the limit of load-bearing capacity. Two different optimization algorithms, Arc Length Algorithm (ALA) and

Nonlinear Stabilization Algorithm (NLA), used in sequence to overcome the optimization failures. The obtained results of their work shown a significant weight reduction of the girder by 16% compared to the original design.

Few publications about the customized I-beam crane girder subjected to yield and buckling criteria are reported. Therefore, the current paper extends the similar techniques mentioned above to optimize custom I-Beam crane designs. Three rectangular plates having the same length (L) and different thicknesses and widths welded by continues full penetration welds to form a custom I-Beam crane design, see Figure 3.3. The live load and the beam span are imposed while each plate thickness and width are considered as design variables that need to be determined to have the minimum weight that respecting the yield, buckling, deflection and fatigue criteria. However, the mathematical calculations based on Cranes Manufacturer Association of America (CMAA) design procedure and the Hybrid Genetic algorithm (GA) are used to find the optimal dimensions of the cross section that satisfy the design constraints. A Mathcad platform is written to handle these calculations. Also, a 3D-solid FE model created; stress analyzed and optimized using ANSYS Workbench software.

3.3 Design Optimization Procedure

Highly sophisticated optimization techniques are needed to achieve an optimal crane design that considers yield, buckling, deflection and fatigue criteria. Such techniques must deal with iterative schemes that require a programming language or a mathematical application such as Mathcad. The general trends of solving such problems in the recent years were emphasizing on carrying out a mathematical solution, an FE solution or a math-FE combined solution. The combined solution conducted in two different ways (Zuberi et al., 2008); the first way is carrying out both types of analysis techniques with the same initial values and takes the most optimal results between them. The second one uses the output results of the mathematical solution as input values of an FE solution. The present study follows the second method. The flowchart in Figure 3.1 illustrates the proposed procedure. It starts with problem formulation, i.e., defines design variables, objective function, etc. Follows that entering the data of crane, which are in our case the span length, the rated load, and the material; then performing the optimization Hybrid Genetic Algorithm (GA) code, the details of which are shown in Figure 3.2, to give the so called Math-Optimal design variables. The Math-optimal design variables are input as initial variables to the FE Optimization phase using ANSYS Workbench 15 software in which the Response Surface Optimization method is used (Ansys, 2015; Lee, 2014).

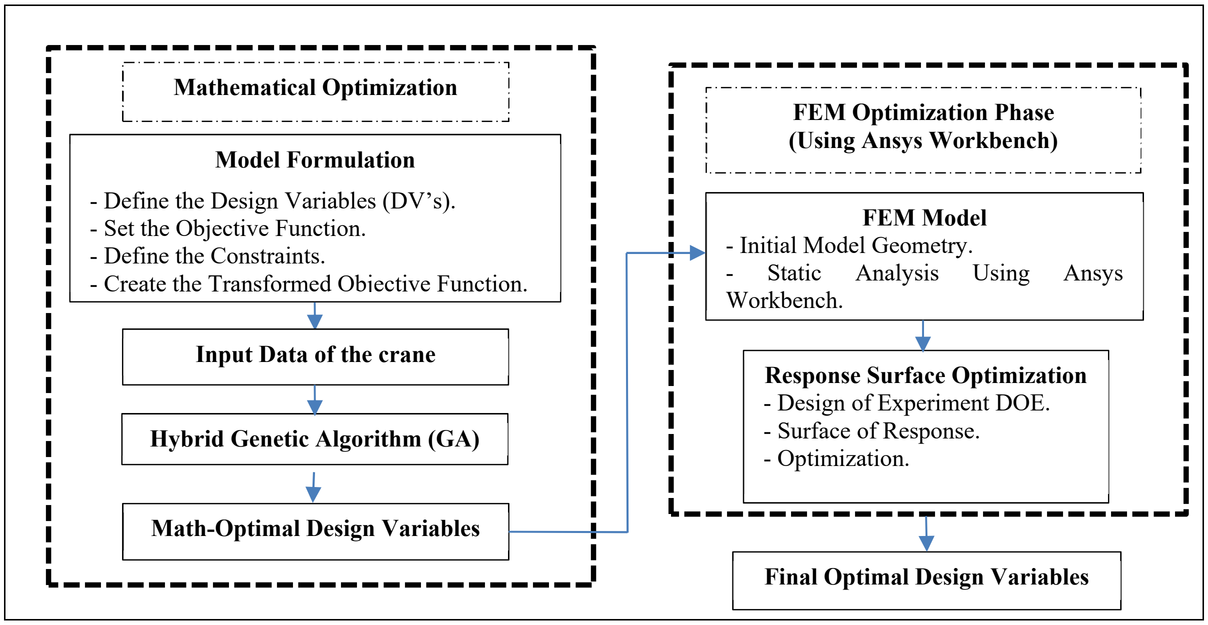

Mathematical Optimization
Model Formulation
- Define the Design Variables (DV's).
- Set the Objective Function.
- Define the Constraints.
- Create the Transformed Objective Function.
Input Data of the crane
Hybrid Genetic Algorithm (GA)
Math-Optimal Design Variables
FEM Optimization Phase (Using Ansys Workbench)
FEM Model
- Initial Model Geometry.
- Static Analysis Using Ansys Workbench.
Response Surface Optimization
- Design of Experiment DOE.
- Surface of Response.
- Optimization.
Final Optimal Design Variables

Figure 3.1 Proposed design optimization procedure

3.4 Problem Description

The welded I-Beam crane and the loading conditions are shown in Figure 3.3. The beam formed by three plates joined by continuous welds over the beam length. They have the same length but different thicknesses and widths; the dimensions and loading conditions defined as follow:

b_1 :	lower flange width,	t_1 :	lower flange thickness,	b_2 :	upper flange width,
t_2 :	upper flange thickness,	h :	web height,	t_3 :	web thickness,
L :	beam span,	W_1:	crane weight,	W_2 :	live load (Lifting load),
x :	distance of live load from the left end				

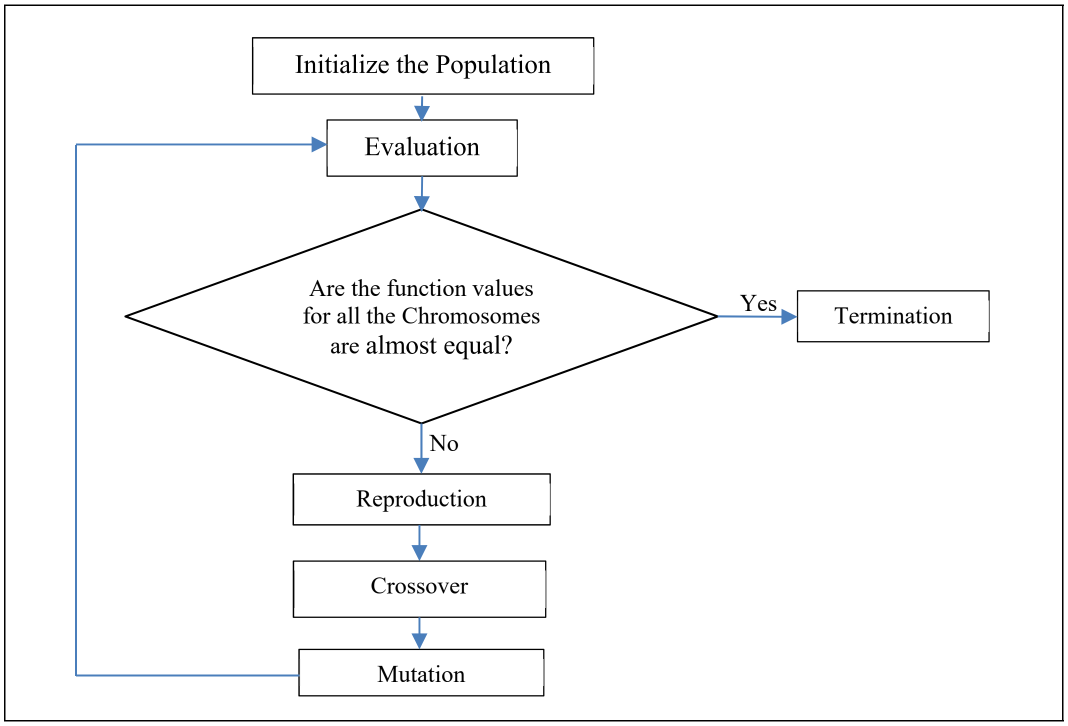

Initialize the Population
Evaluation
Are the function values for all the Chromosomes are almost equal?
Yes
Termination
No
Reproduction
Crossover
Mutation

Figure 3.2 Hybrid Genetic Algorithm
Taken from W. Y. Yang et al. (2005)

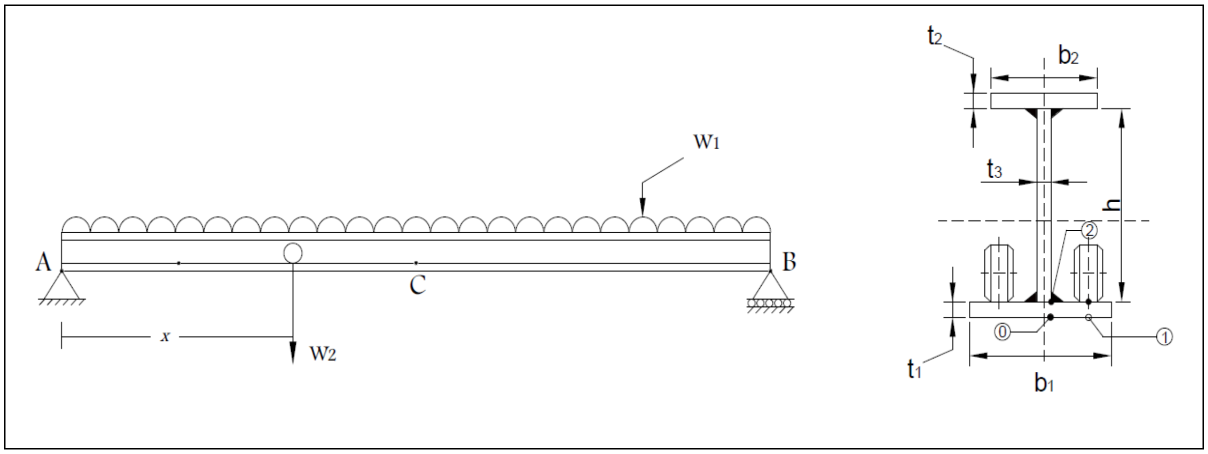


Figure 3.3 The crane beam dimensions and loading conditions

3.4.1 Objective Function

Where the span length is fixed with a constant cross section area of the crane beam, the weight is just proportional to the cross-section area so that the objective function is defined by the cross-section area, as follows :

$$f = b_1 . t_1 + b_2 . t_2 + h . t_3 \tag{3.1}$$

where the parameters b_1, t_1 …etc. are shown in Figure 3.3

3.4.2 Constraints

The most important criteria of the Crane Manufacturers Association of America specification, known as CMAA74-2010, are considered and summarized as follows :

- Tension stress Constraints (due to gravity and live load) :

$$\sigma_{\mathrm{comb}_{\mathrm{max}}} - \sigma_{\mathrm{Tallowed}} \leq 0 \tag{3.2}$$

- Lateral Buckling Constraint:

$$1.9 - \mathrm{f}_{\mathrm{Buckling}} \leq 0 \tag{3.3}$$

- Local Buckling Constraints:

$$\mathrm{h}/\mathrm{t}_3 - 260 \leq 0 \tag{3.4}$$

$$\mathrm{b}_2/2\mathrm{t}_2 - 260/\sqrt{\sigma_{\mathrm{y}}} \leq 0 \tag{3.5}$$

- Deflection Constraint :

$$\delta_{\mathrm{v}} - \mathrm{L}/600 \leq 0 \tag{3.6}$$

- Fatigue Constraint (due to repeated load fluctuation ΔW2 only) :

$$(\Delta\sigma)_{\text{comb-max}} - \Delta\sigma_{\mathrm{allowed}} \leq 0 \tag{3.7}$$

where $\sigma_{comb} = \sqrt{\sigma_x^2 + \sigma_y^2 - \sigma_x\sigma_x + 3\tau_{xy}^2}$ is the Von-Mises equivalent stress;

$\sigma_{Tallowed} =$	Allowable tension stress, according to CMAA 74;
$\delta_v =$	Maximum vertical deflection
$\Delta\sigma =$	stands for stress range
$\sigma_Y =$	Yield strength

$f_{Buckling} =$ Buckling load factor, which means a factor to be multiplied to all applied loads to produce linear buckling of the structure. This factor is given initially by the linear buckling theory, e.g., Timoshenko formulas or by an FE model. It is valid only if the linear buckling stress, which is $\sigma_{Cr0} = f_{Buckling}.\left|\sigma_{upper\ flange}\right|$, is less than ${}^{1}/_{2}.\sigma_Y$; otherwise, it must be modified to take into account the plastic deformation during buckling. The corrected critical stress calculated using Johnson's empirical formula, $\sigma_{cr} = \sigma_Y.\left[1 - \frac{\sigma_Y}{\sigma_{cr0}}\right]$, (Popov, 1976), and the corrected buckling load factor is given by $\frac{\sigma_{cr}}{\left|\sigma_{upperflange}\right|}$.

3.4.3 Objective Function Transformation

The exterior point penalty function is used to transform the constrained optimization problem into an unconstrained problem. The general form of the transformed objective function is:

$$F(X, r_h, r_g) = f(X) + r_h\left[\sum_{k=1}^{i} h_k\,(X)^2\right] + r_g\left[\sum_{j=1}^{m}(max\{0, g_j(X)\})^2\right] \qquad (3.8)$$

where X is the vector representing the design variables, h_k is the kth equality constraint if any, g_j is the j^{th} inequality constraint, r_h and r_g are two additional variables called penalty multipliers (Ragsdell & Phillips, 1976).

3.5 Numerical Examples

Nine cases defined by three span lengths (8, 12 and 20 m) and three rated loads (10, 20 and 40 tons) are selected as numerical examples. The crane specifications are listed in Table 3.1. The material used for the crane is 350W structure steel with yielding strength $S_y = 350\ MPa$, density $\rho = 7850 \frac{kg}{m^3}$, Young's modulus $E = 200\ GPa$, shear modulus $G = 77\ GPa$ and Poisson's ratio $\nu = 0.3$.

The mathematical optimization procedure, described in section 3.3, programmed using Mathcad Code (PTC, 2011). Table 3.2 summarizes the values of GA parameters.

Table 3.1 Crane Specifications according to CMAA 74-2010

Variable	Value/Units
Rated Capacity:	10, 20 or 40 tons
Service Class D:	Heavy Service
Load Class L3:	Normal load = 2/3 of
Cycles Class N2:	rated load
Span:	Up to 500000 cycles
Trolley Weight:	8, 12 or 20 m
Other equipment	1 tons
Load:	1 tons
Bridge Wheel per rail:	One on each side

Table 3.2 Genetic Algorithm Parameters

Parameter	Used value
Number of Variables:	NV = 6
Population size:	NP = 120
Probability of crossover:	PC = 0.85
Probability of mutation:	PM = 0.05
Mutation Parameter:	BM = 5
Maximum generation number:	GMAX = 300

3.6 Finite Element Model

The Figure 3.4 shows a 3D drawing of an I-Beam crane, the Figure 3.5(a) shows the overall view of a 3D-solid FE model of the crane created in ANSYS Workbench © 15, and the Figure 3.5 (b) shows a local zoom around the contact region between the lower flange and the wheels. The lower edges at ends are vertically supported, and the loads to be considered are composed of the distributed gravity load W1 (weight of the beam), and the concentrated load W2 applied on the wheels, W2 being the combination of the lifted load, the weights of trolley and hoist.

All loads are adjusted by factors according to CMAA 74 Specifications. The rated load plus gravity are applied when considering the yield and buckling constraints, inequalities Eq. (3.2) to Eq. (3.5), while the normal load fluctuation, which is just 2/3 of rated load without gravity, is applied when considering the deflection and fatigue constraints, Eq. (3.6) and Eq. (3.7). For the FE model, the Surface Response Optimization method (Ansys, 2015; Lee, 2014), already integrated into ANSYS Workbench, is used. This model contains about 28300 nodes.

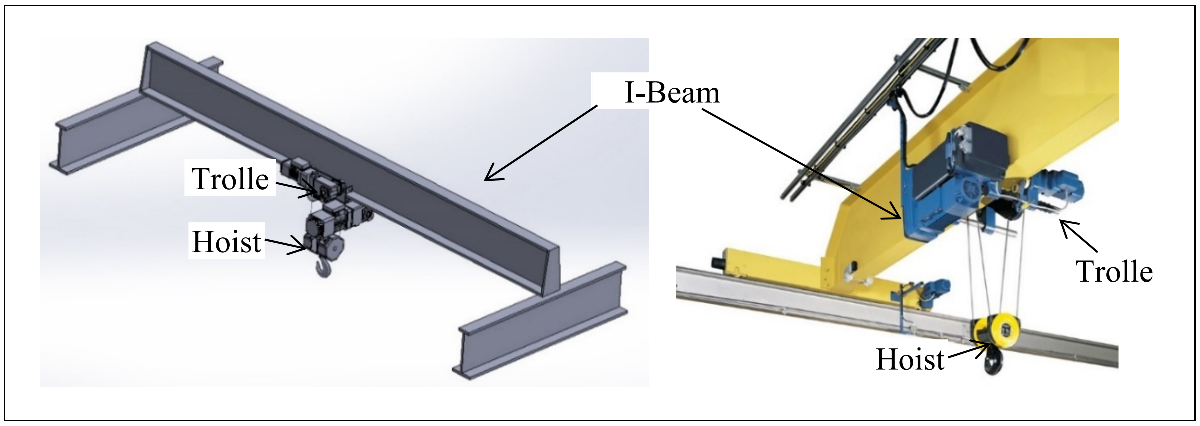
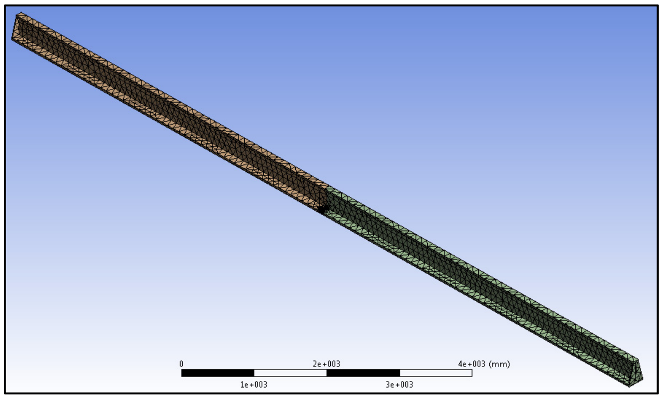
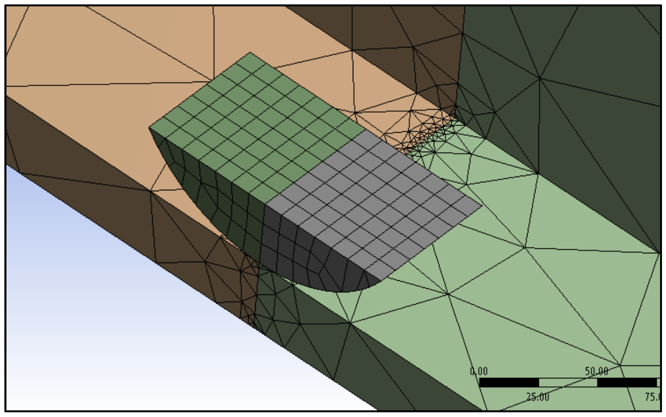


Figure 3.4 Three-dimensional images of the crane



(a)



(b)

Figure 3.5 Finite Element Model of the crane

3.7 Numerical Results

For reducing calculation time, it needs to input the reasonable lower and upper bound values of each design variable. The bounds used in all 9 cases are shown in Table 3.3.

Table 3.3 Lower bound and upper bound of design variables in mm

Variables	t_1	b_1	t_2	b_2	t_3	h
Lower bound	2	150	2	150	3	250
Upper bound	100	600	100	600	100	1675

The optimized design variables for nine cases are presented in the Table 3.4, Table 3.5 and Table 3.6. The results listed in Table 3.4 are for short span cranes with three different rated loads, the Table 3.5 shows the results for intermediate span cranes, and the Table 3.6 for long span cranes results.

Table 3.4 Optimal Design variables and constraint parameters for 8 m cranes

L = 8 m	**10 tons**		**20 tons**	**40 tons**	**Bounds**
	MATH	**FEM**			
$t_1\ (mm)$	27.82	27.98	37.88	52.62	[2, 100]
$b_1\ (mm)$	150.01	150.0	150.16	150.04	[150, 600]
$t_2\ (mm)$	6.99	7.43	8.38	9.08	[2, 100]
$b_2\ (mm)$	194.20	186.1	220.18	252.14	[150, 600]
$t_3\ (mm)$	3.00	3.00	3.19	4.38	[3, 100]
$h\ (mm)$	608.64	650.88	826.46	1137.03	[250, 1675]
$Area\ (m^2)$	0.00736	0.00753	0.0102	0.0152	
$\sigma com_{\max(MPa)}$	221.9	201.9	224.9	225	$\leq\ 225\, MPa$
$f_{Buckling}$	1.9	1.96	1.9	1.91	$\geq\ 1.9$
h/t^3	202.876	216.96	259.37	259.88	$\leq\ 260$
$b^2/2t^2$	13.895	12.524	13.14	13.887	$\leq\ 13.898$
$\delta_v\ (m)$	0.0074	0.0063	0.0057	0.0043	$\leq\ 0.013\, m$
$(\Delta\sigma)_{com\_max}\ (MPa)$	166	160.1	165.9	165.2	$\leq\ 166\, MPa$

Table 3.5 Optimal Design variables and constraint parameters for 12 m cranes

L =12 m	**10 tons**	**20 tons**	**40 tons**	**Bounds**
$t_1\ (mm)$	27.88	53.52	54.61	[2, 100]
$b1\ (mm)$	179.54	150.01	206.13	[150, 600]
$t_2\ (mm)$	10.34	12.04	13.41	[2, 100]
$b_2\ (mm)$	287.15	334.55	372.14	[150, 600]
$t_3\ (mm)$	3.02	3.12	4.21	[3, 100]
$h\ (mm)$	785.98	811.78	1094.81	[250, 1675]
$Area\ (m^2)$	0.0104	0.0146	0.0209	
$\sigma_{com\_max}\ (MPa)$	203.9	188.9	225	$\leq\ 225\, MPa$
$f_{Buckling}$	1.9	1.9	1.9	$\geq\ 1.9$
h/t^3	259.93	259.98	259.98	$\leq\ 260$
$b^2/2t^2$	13.884	13.896	13.877	$\leq\ 13.898$
$\delta_v\ (m)$	0.0095	0.012	0.0097	$\leq\ 0.013\, m$
$(\Delta\sigma)_{com\_max}\ (MPa)$	163	121.3	164.1	$\leq\ 166\, MPa$

Table 3.6 Optimal Design variables and constraint parameters for 20 m cranes

L = 20 m	**10 tons**	**20 tons**	**40 tons**	**Bounds**
$t_1\ (mm)$	30.14	92.15	54.41	[2, 100]
$b_1\ (mm)$	275.88	150.29	347.13	[150, 600]
$t_2\ (mm)$	15.06	14.98	18.1	[2, 100]
$b_2\ (mm)$	418.41	416.3	501.7	[150, 600]
$t_3\ (mm)$	3.29	3.36	4.72	[3, 100]
$h\ (mm)$	821.75	872.94	1226.51	[250, 1675]
$Area\ (m^2)$	0.0173	.0230	0.034	
$\sigma_{com\_max}\ (MPa)$	202.5	199.5	218.2	≤ 225 MPa
$f_{Buckling}$	1.9	1.9	1.9	≥ 1.9
h/t^3	249.96	259.47	259.95	≤ 260
$b^2/2t^2$	13.89	13.891	13.896	≤ 13.898
$\delta_v\ (m)$	0.025	0.032	0.023	≤ 0.033 m
$(\Delta\sigma)_{com\_max}\ (MPa)$	157.7	94.61	166	≤ 166 MPa

It is noticed that the lateral buckling and the upper flange local buckling limits are reached for nine over 9 cases, the web buckling limit for 6/9 cases, the yield and fatigue limits for 3/9 cases and the deflection constraint is never critical. In addition, the optimized I-section configurations always show narrow and thick lower flange, wider and thinner upper flange and tall and very thin web. The Figure 3.6 approximately illustrates the optimum I-Beam cross sectional configuration for a 20 m crane subjected to 20 tons lifted load.

The comparison between the custom I-beam configuration as shown in Figure 3.6, which has $A = 0.023\,m^2$, and a doubly symmetrical I-beam ($t_1 = t_2 = 39.53\,mm$, $b_1 = b_2 = 307\,mm$, $t_3 = 3.85\,mm$, $h = 996\,mm$ and $A = 0.028\,m^2$) shows that the customized I-beam could save almost 18% of the weight. The design parameters given by the Math optimization are then inputted to an FE procedure using ANSYS Workbench 15 with a 3D nonlinear solid model due to the contact between the wheels and the lower flange. The Surface Response Optimization method in ANSYS Workbench used with considering the same constraints, except the linear buckling constraint, because linear buckling does not work with nonlinear contact models.

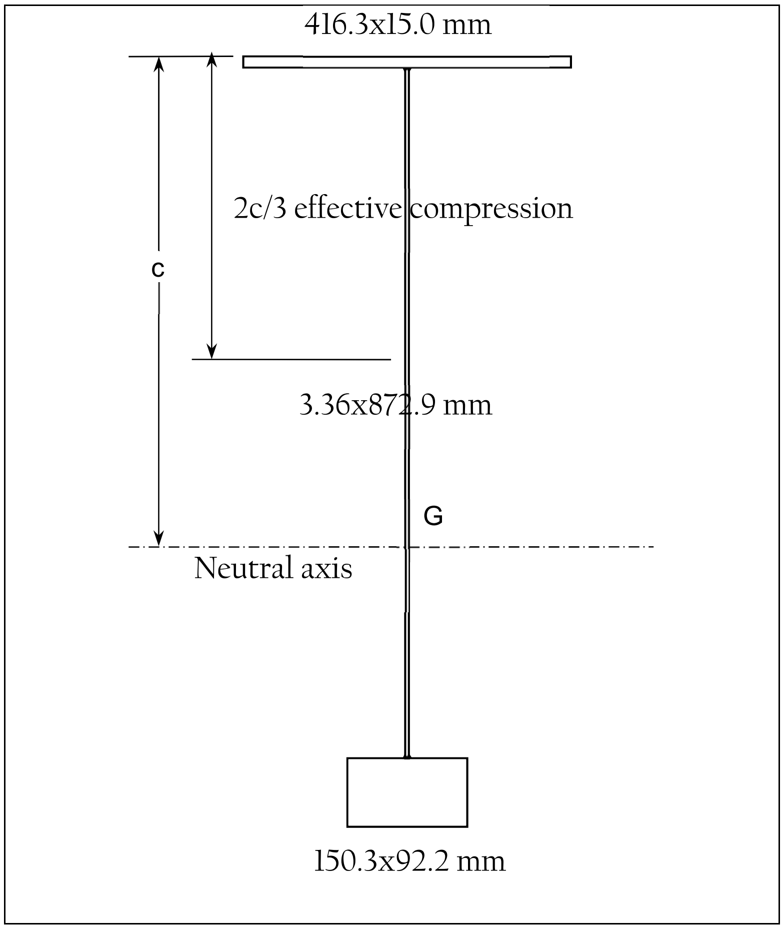


Figure 3.6 Optimum configuration of an I-section

However, the buckling constraint ($f_{Buckling} \geq 1.9$) replaced by an approximate constraint on the slenderness ratio against lateral buckling to give a comparable buckling load factor. This slenderness ratio is given by $\lambda = \frac{L}{r_{cy}}$ where r_{cy} is the lateral radius of gyration of the effective compression area which is empirically the $\frac{2}{3}$ outermost of the compression side of the cross section (see Figure 3.6). FE stress calculation with nonlinear contact and optimization procedure is very time consuming; so only one case selected to show FE results, which is the 8 m and 10-ton case. The slenderness ratio constraint for this case is $\lambda \leq 190$.

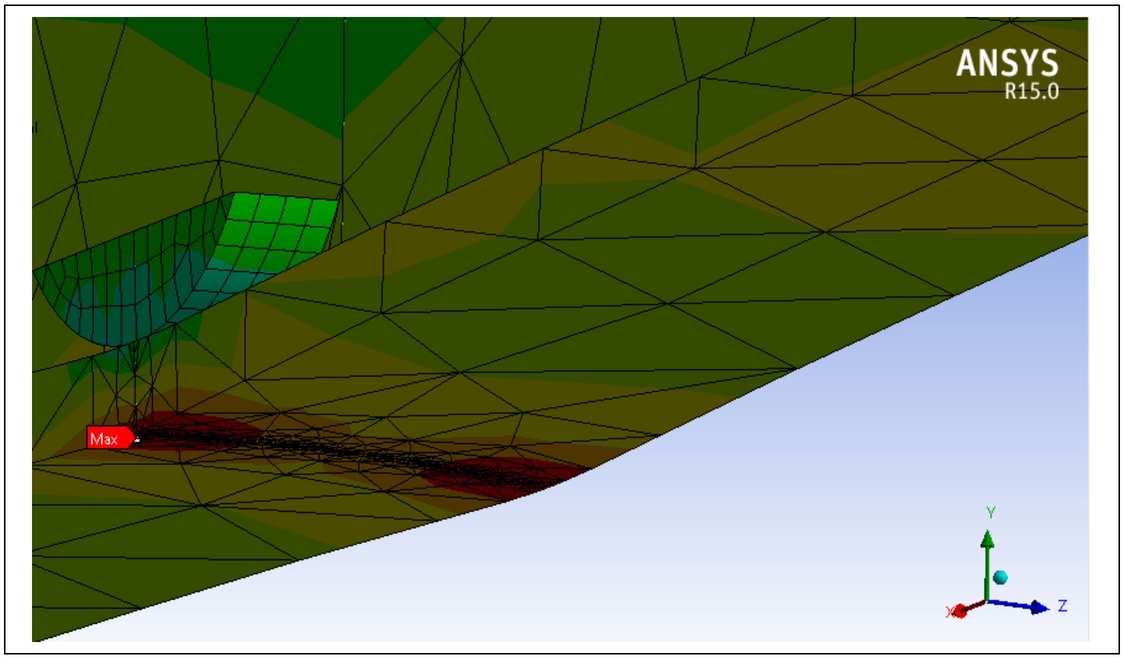


Figure 3.7 Location of maximum Von-Mises stress

The new optimized design parameters given by FE procedure are shown in the FEM column of Table 3.4; they are slightly different but quite close to the Math results. The Figure 3.7 reveals that the maximum Von-Mises stress is in the lower flange right under the wheels.

3.8 Conclusion

A Hybrid Genetic Optimization Algorithm (GA) and a Mathematical optimization procedure are programmed in Mathcad and successfully applied to custom welded I-Beam cranes with different spans and rated loads subjected to yield, buckling, deflection and fatigue criteria. It is found that the constraints of general lateral buckling and local buckling of the upper flange are always reached for all cases. The web local buckling constraint is critical for about 66% of cases, the yield and fatigue constraints found critical for 33% of cases and the deflection constraint is not a problem at all. The optimized custom I-section has a configuration of narrow and thick lower flange, thinner and wider upper flange and the web is tall and very thin, which could save about 18% of weight compared to commercial standard I-Beam. FEM optimization using Surface Response method gives comparable results and confirms that the proposed procedure is efficient.

For future works, the FE optimization taking into account nonlinear buckling due to contact or plasticity constitutes a significant challenge. Furthermore, the optimization procedure with multi objective functions such as weight and cost will also be an interesting future work.

CHAPTER 4

AN ADAPTIVE DISCRETE CUCKOO SEARCH ALGORITHM TO SOLVE STRUCTURAL OPTIMIZATION PROBLEMS

A.Ahmid [a] , T. M. Dao [b] and V. N. Lê [c]

[a,b,c] Department of Mechanical Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper published in *Journal of Multidisciplinary Engineering Science and Technology*,
June 2020

4.1 Abstract

The Cuckoo Search optimization algorithm (CS) continues to grab the attention of the scientific community due to its simplicity and robustness. CS applied successfully to solve a wide range of hard optimization problems, and it exhibited outstanding performance. The current study presents an Adapted variant of Discrete CS Algorithm (ADCSA) that uses the rank-value approach to turn real values of random Lèvy walks (steps/jumps) into the equivalent discrete values. Besides, the proposed ADCSA intensification effort was enhanced by adding four different local search movements of permutation, swap, insertion and bit flip. The solution accuracy of ADCSA was validated across a benchmarking case study of a composite laminated plate. Moreover, a further structural optimization problem of customized I-beam gantry crane was solved using ADCSA. Eventually, the results of both case studies reveal that the proposed ADCSA has a considerable performance in solving discrete structural optimization problems.

Keywords: Cuckoo Search; Discrete optimization; Composite laminate; Gantry crane; critical buckling load.

4.2 Introduction

Cuckoo Search (CS) algorithm is population-based meta-heuristic inspired by the aggressive reproduction strategy of some cuckoo bird species enhanced by Lèvy flights. It presented by X.-S. Yang and Deb (2009) to solve a variety of continuous multimodal optimization problems. Since then, it attracted the attention due to the simplicity of implementation and the fast convergency rate and accuracy of the delivered solutions. Also, CS has a view number of parameters (almost one) to be tuned, compared to other meta-heuristics such as Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO).... etc. The size of CS applications is fascinatingly growing, where it could be observed through the number of solved optimization problems using CS in the last decade (X.-S. Yang, 2014),. Shehab et al. (2017) tracked the progress of published papers that uses CS in the literature. Based on different publisher's metrices, for the CS published articles between 2009 and 2016, he summarized that there are three classes of research interest. The dominant class went to the application, and it represented 67% of publications, whereas CS hybridization and CS modifications had respectively 15 % and 18% of the research interest. The applications of CS involve several main optimization problems such as Travelling Salesman Problem (TSP) (Jati & Manurung, 2012; Ouaarab et al., 2014; X.-S. Yang & Deb, 2013; Zhou et al., 2014), and binary optimization problems, for instance, Knapsack optimization problem (Gherboudj et al., 2012; Layeb, 2011; Xin et al., 2019), Computer vision and image detection (Agrawal et al., 2013; Loubna et al., 2017), Energy sector (de Moura Meneses et al., 2020; Piechocki et al., 2014), supply chain (Q. Li et al., 2020; Z. Li et al., 2018) and structure optimization problem (Gandomi et al., 2013; A Kaveh & Bakhshpoori, 2013). However, the size of CS applications is likely to escalate in the prospective researches where the fast-growing research areas of Artificial intelligence (AI) and data mining are seeking more robust optimization algorithms to build faster response models (Cobos et al., 2014).

Some proposed CS modifications were presented to improve the basic CS through imposing some enhancements of step size, such as using generative scaling factor instead of using constant value (Loubna et al., 2017). Other proposed modifications, in the literature, were mainly focused on adjusting CS to solve discrete optimization problems where the design space is limited to certain values or options (Shehab, 2020; Xin et al., 2019). Some discrete CS variants were explicitly developed to solve particular problems, X.-S. Yang and Deb (2013) used CS to solve TSP or Xin et al. (2019) who developed a discrete binary CS to address allocation of cognitive radio network spectrum optimization problem. The further general approach of handling discreetness constraint was worked out through rounding the generated continuous values, by Lèvy flights, into the nearest integer, and it seems to be work for specific structural optimization problems (A Kaveh & Bakhshpoori, 2013). Loubna et al. (2017) proposed a fascinating approach where he used a rank-value approach to handle a sizeable discrete domain of image detection problem, and the results were impressive, and it deserves attention. Even though these modifications went so far to benefit from CS special features, but it still hard to say that there is one common variant of CS that could solve different discrete nature problems.

The current work presents an Adapted version of Discrete Cuckoo Search Algorithm (ADCSA) that uses a modified rank-value approach to interpret Lèvy flights random steps into equivalent discrete steps. In addition, the intensification capability of ADCSA is enhanced through introducing four different local search movements of permutation, swap, insertion and bit flip. The performance of ADCSA was firstly investigated through a well-known benchmark problem of a composite laminated plate. The obtained results of ADCSA were compared across the previously published results for other meta-heuristics. Moreover, ADCSA results also compared with other two different discrete CS variants, which implemented based on the rounding and original rank-value approaches (A Kaveh & Bakhshpoori, 2013; Loubna et al., 2017). The performance of the proposed ADCSA was remarkably superior to other metaheuristics, and the obtained results demonstrated promising performance of ADCSA in solving discrete structural optimization problems.

Consequently, ADCSA applied to solve the problem of optimization of customized I-beam gantry crane, which started to grab more attention in recent years (Ali Ahmid, Le, & Dao, 2017; Pavlovic , Savkovic, Zdravkovic, Bulatovic, & Markovic, 2018). The different dimensions of the I- beam section need to be taken from a discrete range of steel plates, whereas the span length is fixed. The objective of the optimization is minimizing the cross-section area to reduce the crane weight where it is subjected to different strength constraints. The benefits of using customized I-beam cranes, rather than using standard I and H beams, were explained too.

Finally, the rest of this paper is arranged to explain the original CS in the second section, whereas the proposed ADCSA conceptual implementation is presented in the third section. The fourth section devoted to the validation case study while the customized crane case study demonstrated in the fifth section, and the summary of the current work outcomes and findings, with possible prospective research studies, were stated in the conclusion section.

4.3 Cuckoo Search Via Lévy Flights (CS)

The original Cuckoo Search is a population-based metaheuristic inspired by the reproduction strategy of Cuckoo Search bird. The bird starts searching for the surrounding to find a host nest of other birds. In each candidate nest, Cuckoo bird lay just one egg, and it flies to find another one to lay the next egg. This strategy has precisely coincided with the wisdom says, “Don’t put all eggs in one basket”, which in this occasion, means that the chance of Cuckoo eggs to survive is becoming better. The host bird could discover some of the eggs, and they may discard or abounded (X.-S. Yang, 2014). X.-S. Yang and Deb (2009) introduced the CS algorithm to simulate this natural phenomenon where the total number of candidate nests represents the population size (n), and each nest is a possible solution (S_i). A fraction (Pa) of the whole population with worse fitness is going to be discarded, and this mimic the discovery of the eggs by the hosting birds. Next, new randomly generated solutions are going to substitute the discarded solutions. The top-ranked nests will remain within the next generations. The CS searching of the design space goes via a random walk that taken out from Lèvy probability distribution. The original CS pseudo-code is listed in Algorithm 4.1.

Lèvy flight is the strengthening component of CS where it offers the random walk, though steps/jumps length is selected from Lèvy probability distribution. The jumps (long steps) in the design space are possible because of the heavily tailed nature of Lèvy probability distribution (X.-S. Yang, 2014).

Algorithm 4.1 Cuckoo Search Algorithm (CS)

```
  $\boldsymbol{Initialization}$:
  Initial and evaluate a random population of $n$ host nests $(xi, fi)$.
  $\boldsymbol{While\ (convergence\ not\ met) Do}$:
- Generate a new Cuckoo (population) randomly by Lèvy flights (Eq.(4.1)).
- Evaluate the new Cuckoo fitness $(f_i)$.
- Choose a nest among $n$ $(say, j)$ randomly.
      $If\ fi > fj$
- Replace the j with the new solution.
      $end$
- Rank the solutions and find the current best.
- Discard Pa fraction of worst solutions.
- Substitute the discarded solutions by new ones generated by Lèvy flights.
  $\boldsymbol{End}$
```

In general, the random walk depends on the previous location, x_i^t , and the length of the step/jump, which is the second term of Eq. (4.1).

$$x_i^{t+1} = x_i^t + \alpha \otimes L\grave{e}vy(\lambda, s) \tag{4.1}$$

where α is the step size scale factor and $\alpha > 0$. For most optimization problems, the unity step scale factor could work well (X.-S. Yang & Deb, 2009). The term $L\grave{e}vy(\lambda)$ represents Lèvy probability distribution, λ is Lèvy exponent, and s is the step size.

$$L\grave{e}vy(\lambda, step) \sim u = t^{-\lambda}, (1 < \lambda \leq 3) \tag{4.2}$$

The random direction of the step/jump and step size that follows Lèvy probability are two essential elements to generate random numbers via Lèvy flights. The direction of the step could be randomly drawn from the normal distribution, $N(\mu, \sigma^2)$, whereas the step length needs to be determined through the Magenta algorithm. According, the step size (s) can be determined through the following formula:

$$s = \frac{u}{|v|^{1/\lambda}} \tag{4.3}$$

where

$$u \sim N(0, \sigma^2),\ v \sim N(0,1) \tag{4.4}$$

u, v are Gaussian normal distributions. The definition of u means that the random samples are drawn from a normal distribution that has 0 mean and variance of σ^2 . The variance value could be obtained from:

$$\sigma^2 = \left[\frac{\Gamma(1+\lambda)}{\lambda\Gamma((1+\lambda)/2)} \cdot \frac{\sin(\pi\lambda/2)}{2^{(\lambda-1)/2}}\right]^{1/\lambda} \tag{4.5}$$

where $\Gamma(\mathrm{n})$ is nothing more than factorial of n or n!

4.4 Adaptive Discrete Cuckoo Search Algorithm (ADCSA)

CS Originally presented as population-based metaheuristic to solve unconstrained continuous optimization problems (X.-S. Yang & Deb, 2009). However, several discrete variants were introduced to solve a particular discrete optimization problem (Ouaarab et al., 2014). Others were more general, such as using rounding of the Lèvy flight step into the nearest integer (A Kaveh & Bakhshpoori, 2013). Loubna et al. (2017) presented a universal approach that could generate steps with integer values to obey Lèvy flights random walk. A similar approach has used here, and it is explained in subsection 4.4.2.

The proposed ADCSA bears three main modifications to the original CS. First is using Latin Hypercube (LHC) sampling method to generate the initial population; the second is presenting discrete Lèvy flights representation and finally improve the neighbourhood search of the best solution through four different permutation movements. The proposed ADCSA pseudo-code is listed in Algorithm 4.2.

4.4.1 Initial Population

Q. Li et al. (2020) investigated the impact of initialization methods on the meta-heuristics searching performance. They examined eight different ways of random initialization sampling for five meta-heuristics. Their work results revealed that CS is sensitive to the initialization method, where it performed differently in 73.68 % of the tested functions based on the used method of initialization. Moreover, they suggested that the hybridization of different sampling methods could boost the algorithm performance of searching the design space. However, in the current study, three different sampling methods, Discrete Uniform Distribution (DUD), Latin Hypercube (LHC) and hybrid DUD-LHC, were examined. The numerical experiment results exhibited a slight improvement in the overall performance of ADCSA when LHC was used compared to the other two methods; see Figure 4.5.

Algorithm 4.2 Adapted Discrete Cuckoo Search Algorithm (ADCSA)

```
Initialization:
-   Initial and evaluate a random population of $n$ host nests $(xi, fi)$ using Latin
    Hypercube (LHC) random generator.
While (convergence not met) Do:
Generate a new Cuckoo (population) randomly by Lèvy flights (Eq.8).
    Evaluate the new Cuckoo fitness $(f_i)$.
    Choose a nest among $n$ $(say, j)$ randomly.
    If $fi > fj$
-   Replace the j with the new solution.
    end
    Rank the solutions and find the current best.
    Discard Pa fraction of worst solutions.
    Substitute the discarded solutions by new ones generated by Lèvy flights.
            If (maximum successful runs number exceeded)
    Do permutation, swap, insertion and bit flip for the current best solution.
              end
    Update the best solution.
    End
```

4.4.2 Discrete Lèvy Flights Representation

The proposed approach by Loubna et al. (2017) defines the design space domain by rank and value. The "rank" refers to the location of the variable within the design domain vector, D, while "value" represents the corresponding assigned variable value/option, d_i. So, the new Cuckoo,$x_{(i+1)}$ is generated based on the current solution element rank, integer number, and an integer step size that obeys Lèvy flights, see Eq.(4.1). Next, the new cuckoo with rank form is transformed into the equivalent values form. Therefore, $X_i^t = [x_1, x_2, \ldots . x_n]$ represents the design variables vector, and N is the problem size. Whereas, $D = [d_1\ d_2\ \ldots .\ d_M]$ indicates the discrete domain of the optimization problem. So,

$$Rank\ (D) = \{1,2, \ldots ., M\},$$
$$Value\big(Rank(D)\big) = \{\ d_1, d_2, \ldots, d_M\}$$

An improved version of the value-rank approach was implemented here. The improvements went to the update strategy of the size scale factor α, and to the step size determination. Selectin the step size factor α could influence the performance of the algorithm significantly, and it linked to the problem nature (X.-S. Yang & Deb, 2009). Using a constant value for α, e.g. 0.01 or 1, might work, but it doesn't consider any problem characteristics such as the solution fitness/quality. So, the proposed scale factor here is examining the quality of the fitness of the individual solution, f_i, to the fitness of the best solution, f_{best} .

$$\alpha = \frac{f_i}{f_{best}} \tag{4.6}$$

Consequently, the new solution, x_{i+1}, will be updated according to:

$$x_{i+1} = x_i + \frac{f_i}{f_{best}} . step \otimes (x_{best} - x_i) \tag{4.7}$$

Now, the updated step size, second term in Eq.(4.7), produces real values, while the current solution x_i has an integer representation (rank) of discrete values and thus the new solution,

x_{i+1} is going to have real values which we couldn't use it directly as ranking values. Therefore, a transformation function was used to turn the real values of the step size into their equivalent integer values. The sigmoid function, Eq.(4.8), is widely used in solving classification problems by machine learning (ML) algorithms (Shalev-Shwartz & Ben-David, 2014); also, it used in the binary variant of CS to solve the knapsack problem, (Ouaarab et al., 2014).

$$Sigmoid(z) = \frac{1}{1+e^{-z}} \tag{4.8}$$

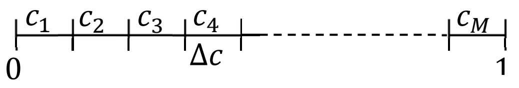
Sigmoid function, Figure 4.1, gives a selection probability between 0 and 1 for any input values, which is the step size in our case.

$$p_i(\alpha.s.\Delta xx_{bset}) = \frac{1}{1+e^{-\alpha.s.\Delta xx_{bset}}} \tag{4.9}$$

The next step is dividing the interval [0,1] to the desired number of classes (ranks), C, and this gives each class (rank) a range of selection, $\triangle c$.

$$C = |D| \implies C = M$$

Then, the obtained selection probability,p_i, is compared across all ranges to determine the class to which this p_i belongs.



$$\Delta c = \frac{1}{M}$$

$$(k-1).\Delta c \leq c_i < k.\Delta c\,, \qquad (i,k = 1{:}M)$$

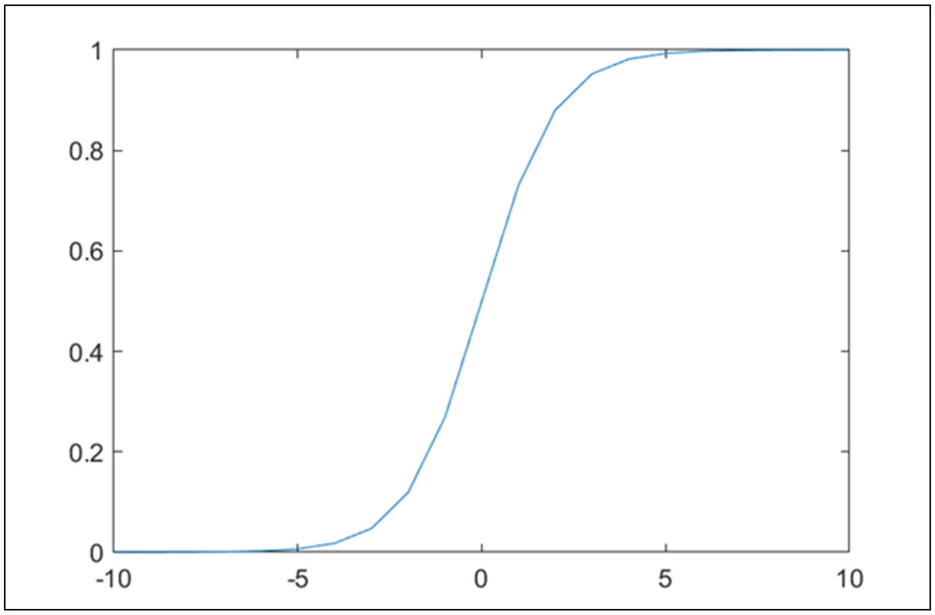


Figure 4.1 Sigmoid transformation function

Despite the successful transformation of the original step size into its equivalent integer size, the new solution x_{i+1} might exceed the domain bounds when Eq. (4.8) applied. However, we experience a similar situation every day when we use the clock arithmetic to keep the time; e.g. the clock now is (8:00 am), and we want to know where will the hour hand be in 5 hours? ($8\ am + 5 \equiv 13$) Obviously, it exceeds the clock bounds, which is 12, but intuitively we say it is 1:00 pm. The mathematical interpretation of this, that the remainder of dividing 13 by 12, is one, and this is the typical definition of modulo function that we are going to use to reflect a meaningful value for out-off bounds ranks. Therefore,

$$rank(x_{i+1}) = mod\big((x_i + s), |D|\big) + 1 \tag{4.10}$$

where s represents the integer value of the Lèvy flights steps/jumps, and 1 is the minimum rank value if $mod\big((x_i + s), |D|\big) = 0$. Eventually, the ranked new solution, $rank(x_{i+1})$, reversed to its assigned rank values or $value\big(rank(x_{i+1})\big)$ form that we could use to determine the objective optimization function directly.

4.4.3 Neighbourhood Search

X.-S. Yang (2013) expect that CS intensification could be improved by using local search Lèvy flights or hybridization CS with other local optimization algorithms (e.g. Tabu Search). The primary purpose of enhancing the intensification feature of any meta-heuristic is to ensure that the obtained solution is not a local optimum, and there was no possible global optimal solution left behind the current best solution. However, the proposed ADCSA turns the optimization problem into a pure permutation problem, as a result of using the rank-value approach. The ranked solution has an integer representation that we could permute to produce a new ranked solution. Based on this, four different permutation operators employed in ADCSA to improve the search of the current best solution neighborhood.

a.) Random permutation

Random permutation operator is selecting randomly two elements of the solution vector and reverses the order of the other elements in between. Let's that we have a six dimensions solution vector as follow:

$$x = [1\ 2\ 3\ 4\ 5\ 6]$$

So, a possible permutation is:
Before permutation:

$$x = [1\ \mathbf{2}\ 3\ 4\ \mathbf{5}\ 6]$$

After permutation:

$$x = [1\ \mathbf{5}\ 4\ 3\ \mathbf{2}\ 6]$$

b.) Swap (mutation)

Swap operator also knows as mutation, selects randomly two elements and switch over their positions in the solution vector.

Before swap:

$$x = [1\ 2\ \mathbf{3}\ 4\ 5\ \mathbf{6}]$$

After swap:

$$x = [1\ 2\ \mathbf{6}\ 4\ 5\ \mathbf{3}]$$

c.) Insertion

The insertion operator is randomly selecting an element of the solution vector and insert it randomly between the other two elements.

Before insertion:

$$x = [1\ 2\ 3\ 4\ \mathbf{5}\ 6]$$

After insertion:

$$x = [1\ \mathbf{5}\ 2\ 3\ 4\ \ 6]$$

d.) Bit flip

Bit flip operator selects a random element (bit) of the solution vector and changes its rank order randomly.

Before bit flip:

$$x = [1\ 2\ 3\ 4\ 5\ 6]$$

After a bit flip:

$$x = [1\ 2\ \mathbf{2}\ 4\ 5\ 6]$$

Random permutation and swap operators were used in solving a well-known benchmark problem of structural engineering of composite laminated design optimization by Genetic Algorithm (GA), (R. Le Riche & Haftka, 1993). Whereas, insertion and bit flip operators were used efficiently, as a part of an improved Tabu search algorithm in searching of the neighbourhood of large design space optimization problems (He, de Weerdt, & Yorke-Smith, 2019). Lastly, these operators were integrated into ADCSA structure in the way that they will not be activated until a certain number of successful runs is reached.

4.4.4 Convergence Criteria

ADCSA designed to stop after a certain number of successful runs without any solution improvement is reached. Otherwise, it continues searching for the optimal solution until the predefined maximum number of iterations is exceeded.

4.5 Numerical Experiments

In order to examine the performance of the proposed ADCSA in solving discrete structural optimization problems, two different case studies were selected from the literature. The first case study is a benchmark problem used here as a numerical experiment to validate the solution accuracy of ADCSA and to compare its performance with other meta-heuristics in the literature. Thus, a benchmark problem of a composite laminated plate subjected to bi-directional compression loading was used in this experiment (R. Le Riche & Haftka, 1993). To ensure the robustness of ADCSA, another discrete structure case study of customized I-beam gantry crane subjected to yield criteria (Ali Ahmid et al., 2017).

4.5.1 Validation Numerical Experiment

The benchmark optimization problem of a composite laminated plate subjected to compression loading is extensively used in the literature to investigate the performance of a new or modified meta-heuristics (Aymerich & Serra, 2008; Rubem Matimoto Koide et al., 2013). In a more recent study, the same benchmark problem used to compare the performance of five different meta-heuristics (A; Ahmid, Thien-My, & Van Ngan, 2019). The original optimization problem introduced by (R. Le Riche & Haftka, 1993) for a laminated rectangular plate simply supported. The in-plane compression loading conditions were applied in the direction of both axes x, y, see Figure 1.1 The optimization objective is maximizing the critical buckling loading capacity of the plate subjected to design and manufacturing constraints. Moreover, the number of plies, N_p, and thickness of each ply, t_p, are imposed while the fiber orientation of each plies group, θ_p, needs to be chosen from a discrete domain of available orientations, $D = [0°; \pm 45°;\ 90°]$.

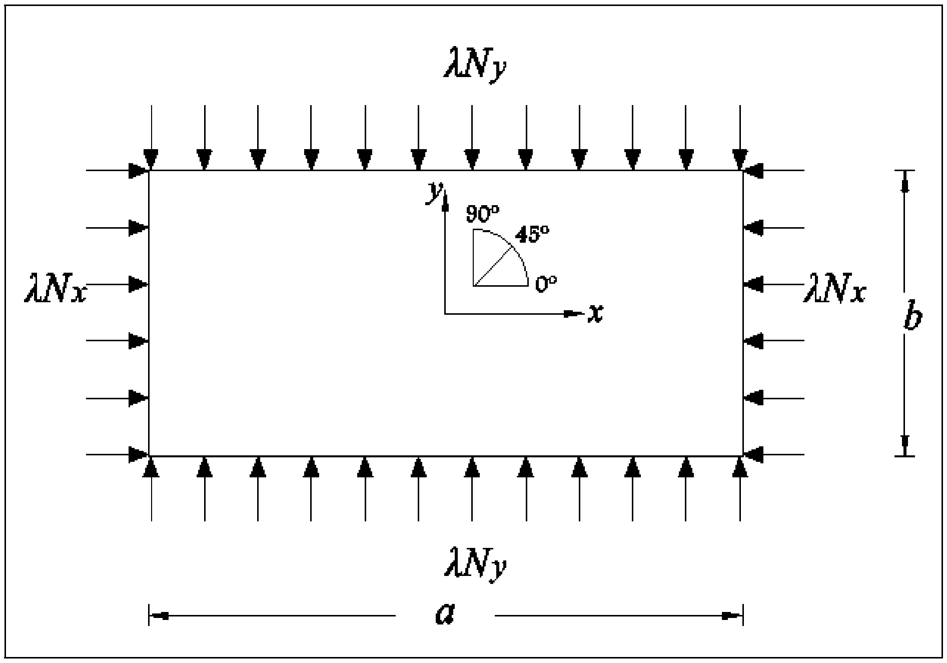


Figure 4.2 Simply supported plate subjected to biaxial loading (A; Ahmid et al., 2019)

The design variable vector, X, is formed by the number of plies groups that meet the symmetry and balanced constraints. The symmetrical laminate means that both sides about the mid-plane have the same number of the plies and this reduces the number of plies, to be optimized, into the half of the total number of plies, Np=2 .Balanced laminate is symmetrical one where each group of two plies, with same fiber orientation, on one side has a similar group on the other side, and this downsize the number of the optimized plies to another half. Thus, the final number of design variables (optimized plies) will be equal to $N_p = 4$ .The formula of buckling load factor λ_b , which developed according to Classical Lamination Theory (CLT), has been used implicitly to determine the objective function of critical buckling load, λ_{cb} , (R. Le Riche & Haftka, 1993), Therefore,

$$\lambda_b(\mathrm{p},\mathrm{q}) = \pi^2 \frac{\left[D_{11}\left({}^{p}/_{a}\right)^4 + 2(D_{12}+2D_{66})\left({}^{p}/_{a}\right)^2 + D_{22}\left({}^{q}/_{b}\right)^4\right]}{\left({}^{p}/_{a}\right)^2 N_x + \left({}^{q}/_{b}\right)^2 N_y} \tag{4.11}$$

where D_{ij} is the bending stiffness, N_x, N_y are in-plane compression loads in x, y. The variables p , q denote the buckling modes in both x, y directions. The critical buckling load factor λ_{cb}, is defined as the minimum obtained value of $\lambda_b(\mathrm{p},\mathrm{q})$, (S. S. Rao, 2009).

The orthotropic material properties, dimensions, and loading conditions of the composite laminated plate used in this experiment are listed in Table 4.1 and Table 4.2.

4.5.1.1 Experiment Setting

The proposed ADCSA code written in Matlab 2019b programming language. In addition, the other two discrete variants of presented by A Kaveh and Bakhshpoori (2013) and Loubna et al. (2017) were implemented and programmed using Matlab 2019b. All implemented variants tested on the same PC-machine with the listed specifications in Figure 4.3. The experiment initialized with the same number of nests, $n_{Nest} = 100$, and same discovery rate, $P_a = 0.25$. For each variant, the experiment has been repeated 200 times to overcome the stochastic behaviour of meta-heuristics as recommended in the original reference, (R. Le Riche & Haftka, 1993).

```
=====================================================================
User: Ali Ahmid ……………………………………………………………………..15-Feb-2020 00:24:25
=====================================================================
        Machine Information:
        CPU Processor: Intel(R) Core (TM) i7-4790 CPU @ 3.60GHz
        CPU clock speed: 3601 MHz
        CPU Cache size (L2): 1024 KB
        Number of physical CPU cores: 4
        Installed physical memory (RAM): 16 GB
        operating System Type: Windows
        Operating System Version: Microsoft Windows 7 Enterprise
=====================================================================
```

Figure 4.3 The specifications of PC-machine used in the current comparison study

Furthermore, an assessment of three different random initialization methods of Discrete Uniform Distribution (DUD), Latin hypercube (LHC) and hybrid DUD-LHC were conducted. The results demonstrated slightly better performance of ADCSA when LHC used to generate the initial population, see Figure 4.5. Thus, it used to generate the initial population of ADCSA in the executed validation experiments.

4.5.1.2 ADCSA Performance Assessment Criteria

In the literature, there were different measures used to assess meta-heuristics performance. Elapsed time is not only the measure used to evaluate the computational solution cost where the success rate (or reliability) and the average number of runs required to find the optimal solution (price) were commonly used too. Furthermore, normalizing the solution price, price/ reliability, could reveal valuable information about the solution cost (A; Ahmid et al., 2019; R. Le Riche & Haftka, 1993). Lastly, where such an optimization problem has multi-optimal solutions, the term of practical optima is used. It is devoted to considering the near-optimal solutions of 0.1% error to the best-known optimal solution (Aymerich & Serra, 2008).

4.5.2 Customized I-Beam Gantry Crane Problem

The customized I-beam gantry crane design is another structural design problem that started to attract the attention, (Ali Ahmid et al., 2017; Alhorani, 2020; Pavlovic et al., 2018). The original problem statement says that for a welded I-beam profile, gantry crane built by welding three different steel plates that have the same length, but they were diverse in their thickness and width. The live loading condition was applied to the crane, see Figure 4.4.The nomenclature of different crane dimensions and loads are given as: $\mathrm{b}_1, \mathrm{t}_1$ and b_2 t_2 are lower and upper flanges widths and thicknesses respectively, while h, t_3 are the width and thickness and width of the web. W_1represents the crane weight and W_2 is the live load. Lastly, L is the crane span, and x is the distance of W_2 measured from the crane left end. The optimization objective is reducing the crane weight by minimizing the crane cross-section area, which is defined by:

$$A_{cs} = b_1 . t_1 + b_2 . t_2 + h . t_3 \quad (4.12)$$

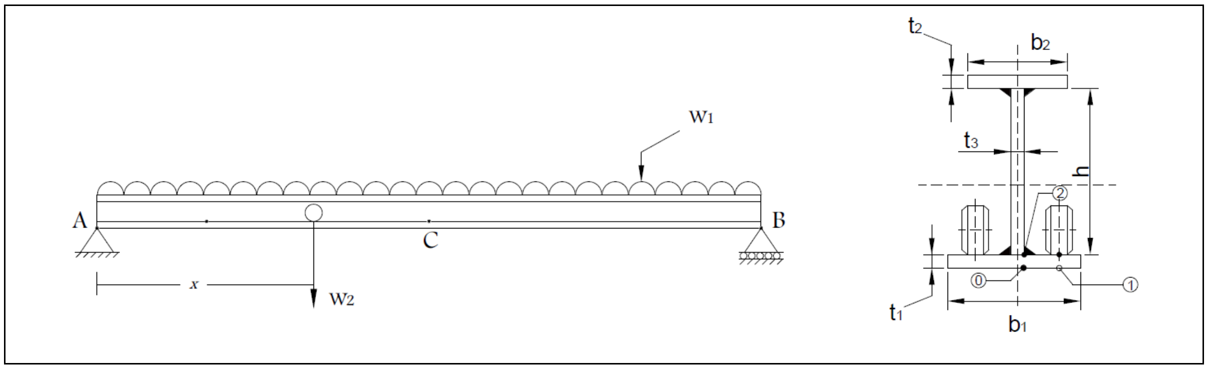


Figure 4.4 The crane beam dimensions and loading conditions, (Ali Ahmid et al., 2017)

The crane design is subjected to bending and buckling criteria, which result in a set of design constraints. Hence,

$$g_1 = \sigma_{\mathrm{comb_{max}}} - \sigma_{\mathrm{Tallowed}} \leq 0 \quad (4.13)$$

$$g_2 = 1.9 - f_{\mathrm{Buckling}} \leq 0 \quad (4.14)$$

$$g_3 = h/t_3 - 260 \leq 0 \quad (4.15)$$

$$g_4 = b_2/2t_2 - 260/\sqrt{\sigma_y} \leq 0 \quad (4.16)$$

$$g_5 = \delta_v - L/600 \leq 0 \quad (4.17)$$

$$g_6 = (\Delta\sigma)_{\mathrm{comb\text{-}max}} - \Delta\sigma_{\mathrm{allowed}} \leq 0 \quad (4.18)$$

These constraints were imposed by using the exterior penalty function that transforms the objective function, A_{cs}, into:

$$F(X, r_g) = A_{cs}(X) + r_g\left[\sum_{j=1}^{m}(max\{0, g_j(X)\})^2\right] \quad (4.19)$$

where $X = [b_1\ t_1\ b_2\ t_2\ h\ t_3]$ and r_g is penalty multiplier for inequality constraints g_i.

Eventually, the material used for the crane is 350W structure steel with yielding strength S_y=350 MPa, density ρ=7850 $\frac{kg}{m^3}$, Young's modulus E=200 GPa, shear modulus G=77 GPa and Poisson's ratio ν=0.3. The dimensions intervals and loads of the crane optimized here are:

$$
\begin{array}{ll}
b_1 \epsilon \, [150{:}\,10{:}\,490] & , \text{(mm)} \\
t_1 \, \epsilon \, [6{:}\,2{:}\,74] & , \text{(mm)} \\
b_2 \, \epsilon \, [150{:}\,10{:}\,490] & , \text{(mm)} \\
t_2 \;\; \epsilon \, [6{:}\,2{:}\,74] & , \text{(mm)} \\
h \;\; \epsilon \, [600{:}\,20{:}\,1280] & , \text{(mm)} \\
t_3 \;\; \epsilon \, [2{:}\,36] & , \text{(mm)} \\
L \; = 8 & , \text{(m)} \\
W_2 = 10{,}20{,}40 & , \text{(ton)}
\end{array}
$$

4.6 Results and Discussions

The obtained results of the validation experiment and the case study of customized I-beam gantry crane are illustrated and discussed in the following subsections.

4.6.1 Validation of Experiment Results

The results of the three initialization methods, they mentioned in section 4.4.1, were statistically compared and depicted in standard division graph in Figure 4.5. Moreover, the two variants of discrete CS by A Kaveh and Bakhshpoori (2013) and Loubna et al. (2017) where implemented, as described, and they were given two abbreviations, RDCS and ADCS, respectively. Consequently, they were applied for the same experiment with the same number of experiments. The number of 30 runs without improving was used as convergency criteria to break the variant searching loop. The proposed ADCSA was also examined with the same experiment setting, and the obtained results of all discrete CS were illustrated in Figure 4.5 to Figure 4.8. Finally, the summary of the comparison of different published results and the proposed algorithms were listed in Table 4.1.

Table 4.1 Comparison of different performance measures for ADCSA and other meta-heuristics

Meta-heuristic	Price	Reliability, %	Normalized Price	Elapsed Time, sec
ADCSA	41	98%	42	6
ADCS (Lubona 2017)	71	56.5%	126	18
RDCS (Kaveh 2013)	243	93.5%	259	118
GA (LeRiche 1993)	371	98.9%	375	NA
GA (Ahmid 2019)	252	88%	286	16
ACO (Ahmid 2019)	88	76.5%	115	4

The results reveal that the proposed ADCSA outperforms the other presented DCS algorithms and other meta-heuristics in literature as solving algorithm for the composite laminated plate. ADCSA exhibited a fast convergence rate where it needs around 12 iterations to find the optimal solution. Moreover, ADCSA delivers an accurate solution with So, the accuracy of the proposed ADCSA is examined, and it has shown significant performance in solving the NP-optimization problem of structural engineering by 98% reliability (successful rate) at 41 iterations solution cost.

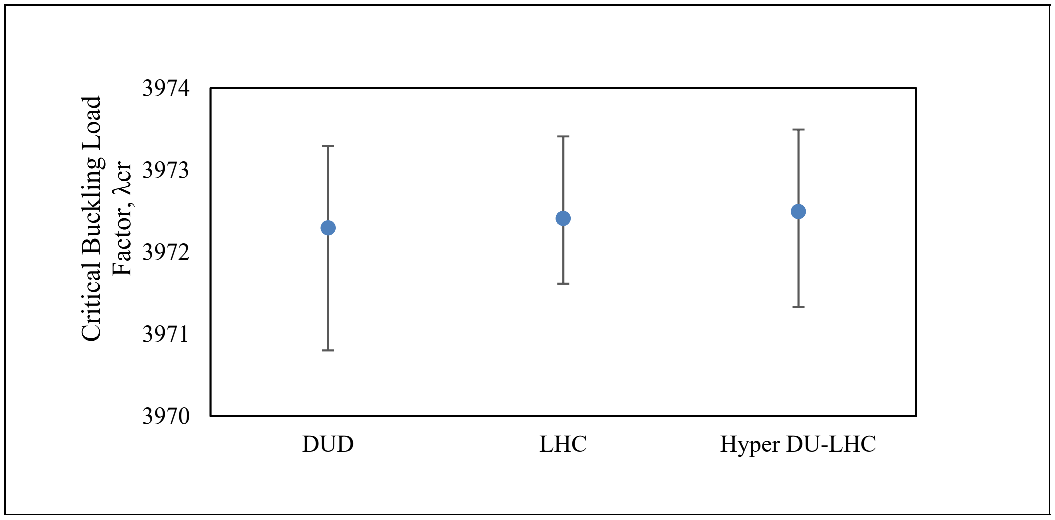


Figure 4.5 Standard deviation plot for different initialization methods

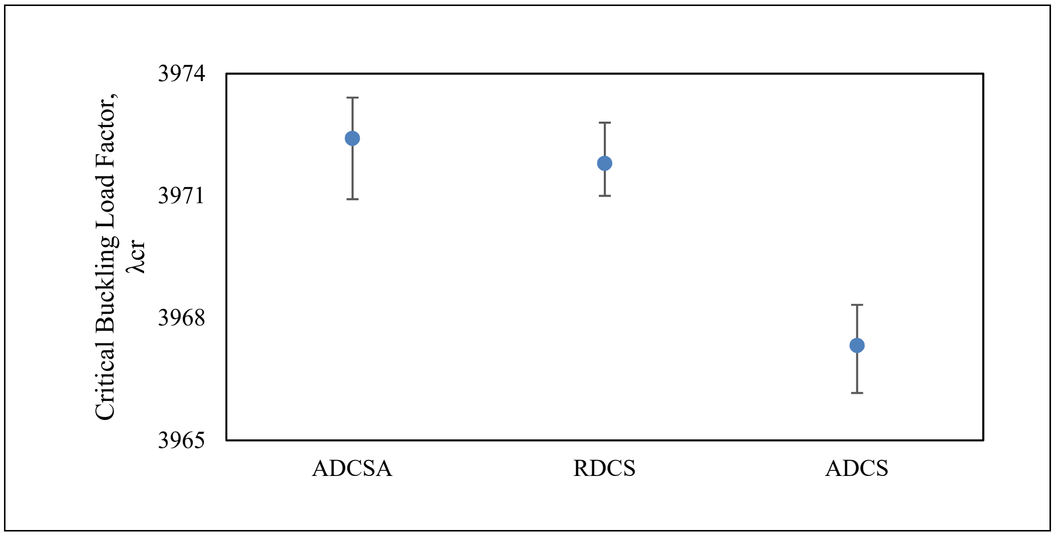


Figure 4.6 Standard deviation plot for λ_{cr} of different discrete CS algorithms

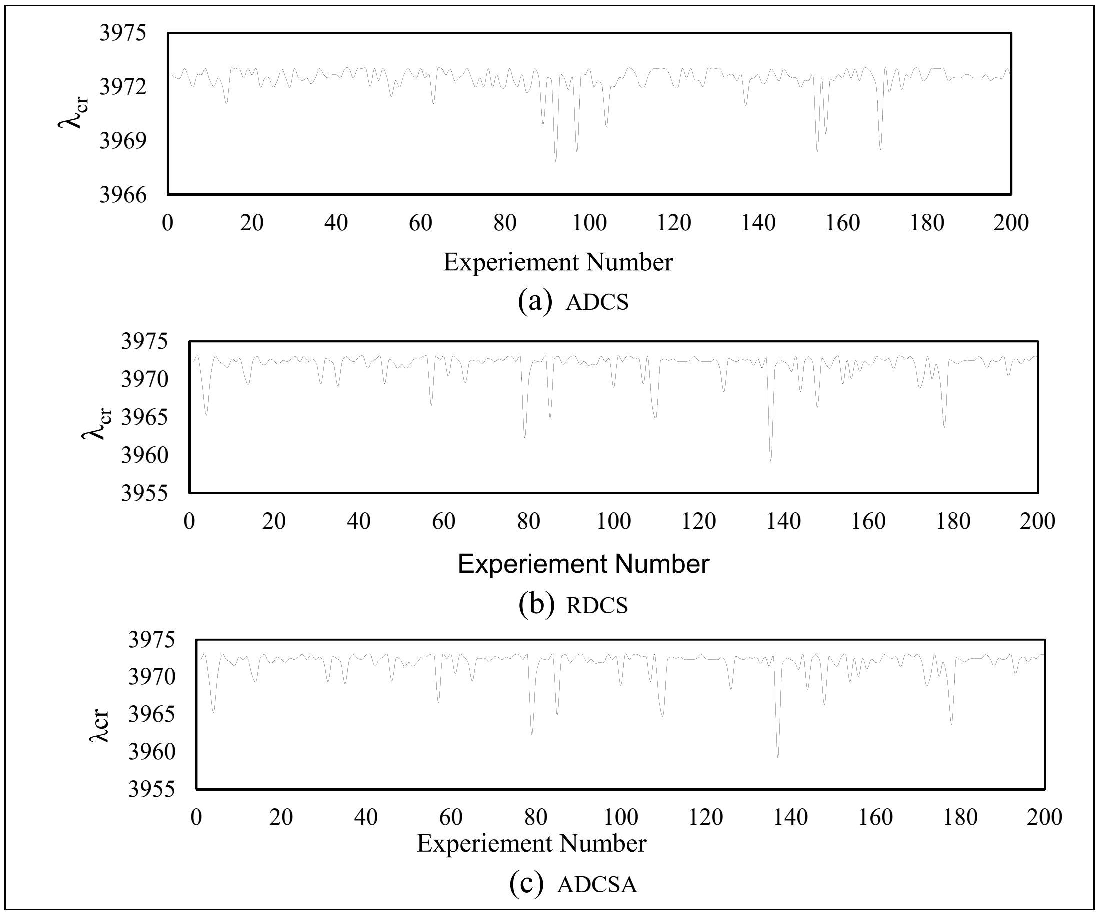


Figure 4.7 The number of experiments vs. critical buckling load for ADCSA, RDCS, ADCS

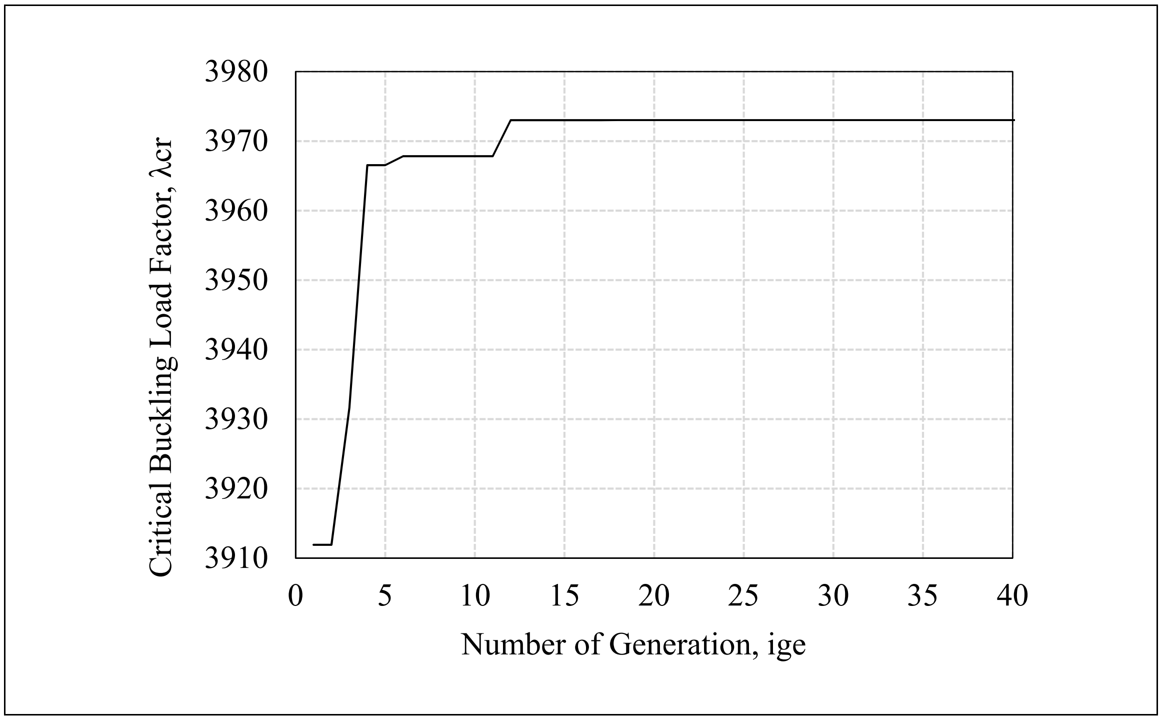


Figure 4.8 ADCSA meta-heuristic Convergence in the first successful run

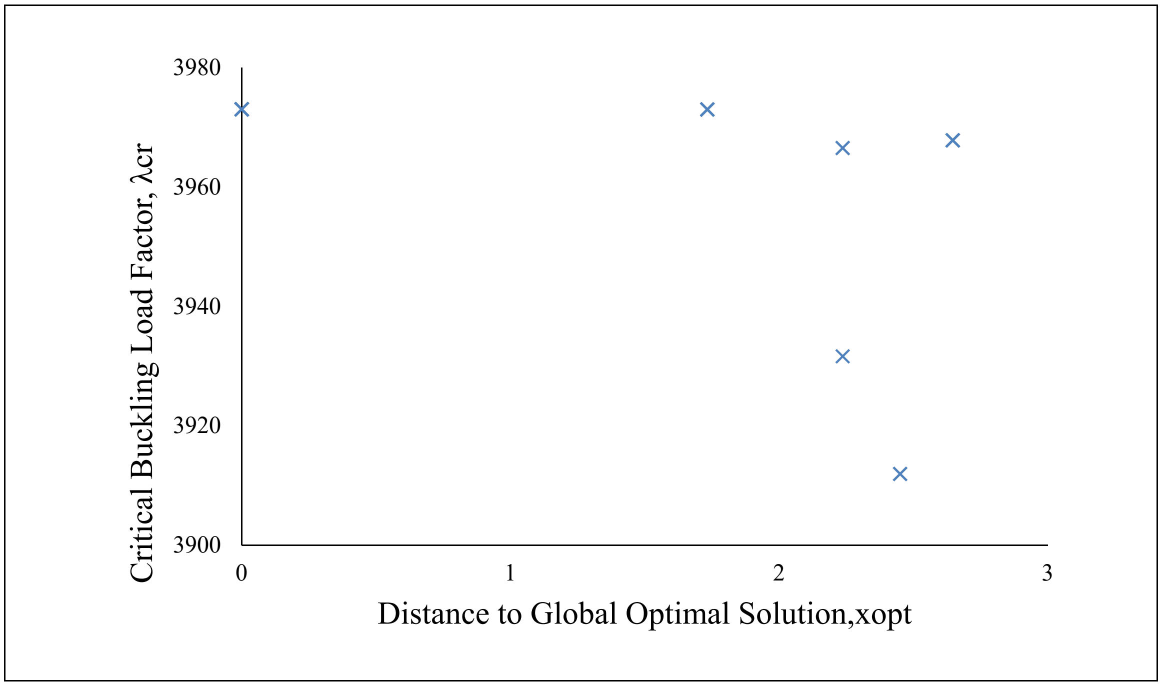


Figure 4.9 Distance to global optimal for ADCSA meta-heuristic first successful run

The next section is devoted to demonstrating the results of applying ADCSA to other structural engineering optimization problems.

4.6.2 Customized I-beam Gantry crane Results

The obtained results of discrete crane design optimization using ADCSA were compared to previously published one of the continuous optimization approaches. Furthermore, both results compared to an equivalent standard I-beam profile for the same optimization strength constraints (CISA). The comparison study produced nine different examples of 8 m crane, where the three different types of crane beams were subjected to three live loads of 10,20 and 40 tons.

The results of this comparison listed in Figure 4.10 to Figure 4.17 and the abbreviations CC, DC and ES are mentioning the three different types of I-beams crane: Continues optimized Custom (CC), Discrete optimized Custom (DC) and Equivalent Standard I-beam (ES) respectively. The results obtained here reveal that the crane cross-section profile of the discrete optimization approach followed the same configurations pattern achieved using a continues optimization approach, see Table 4.2. It always shows narrow and thick lower flange, wider and thinner upper flange and tall and very thin web, see Figure 4.17. Furthermore, both approaches of optimal custom crane did not violate any imposed constraints for the three live loads. At the same time, the equivalent standard I-beam failed to remain within the strength limits of tension, and fatigue stresses constraints for the 40-ton case, see Figure 4.10 and Figure 4.11. The lateral buckling of 10-ton live load almost reached the limit for the three I-beam types, see Figure 4.12. The local buckling of the top flange became critical for CC I-beam, while it was never critical for DC or ES I-beam types. On the other hand, the web slenderness was not critical for any CC I-beam loading cases while it reaches the limits in the first loading case (10-ton) for the other two types, see Figure 4.13. Finally, the results reveal that the discrete optimization approach could reduce the weight from 61 - 69 % of the equivalent standard I-beam crane structure.

Table 4.2 Different optimal solution configuration of customized I-beam gantry crane

Design Approach		**t_1** (mm)	**b_1** (mm)	**t_2** (mm)	**b_2** (mm)	**t_3** (mm)	**h** (mm)	**Area** (m^2)
CC	**10 tons**	27.82	150.01	6.99	194.2	3	608.64	.00736
	20 tons	37.88	150.16	8.38	220.18	3.19	826.46	.0102
	40 tons	52.62	150.04	9.08	252.14	4.38	1137.03	.0152
ES	**10 tons**	56.9	270.3	56.9	270.3	31.5	826.46	.0513
	20 tons	54.1	305.2	54.1	305.2	30	796.8	.0572
	40 tons	40	550	40	550	16	1120	.0621
DC	**10 tons**	54	150	34	170	3	620	.01574
	20 tons	46	170	10	200	14	760	.0177
	40 tons	46	190	16	290	10	1080	0.02418

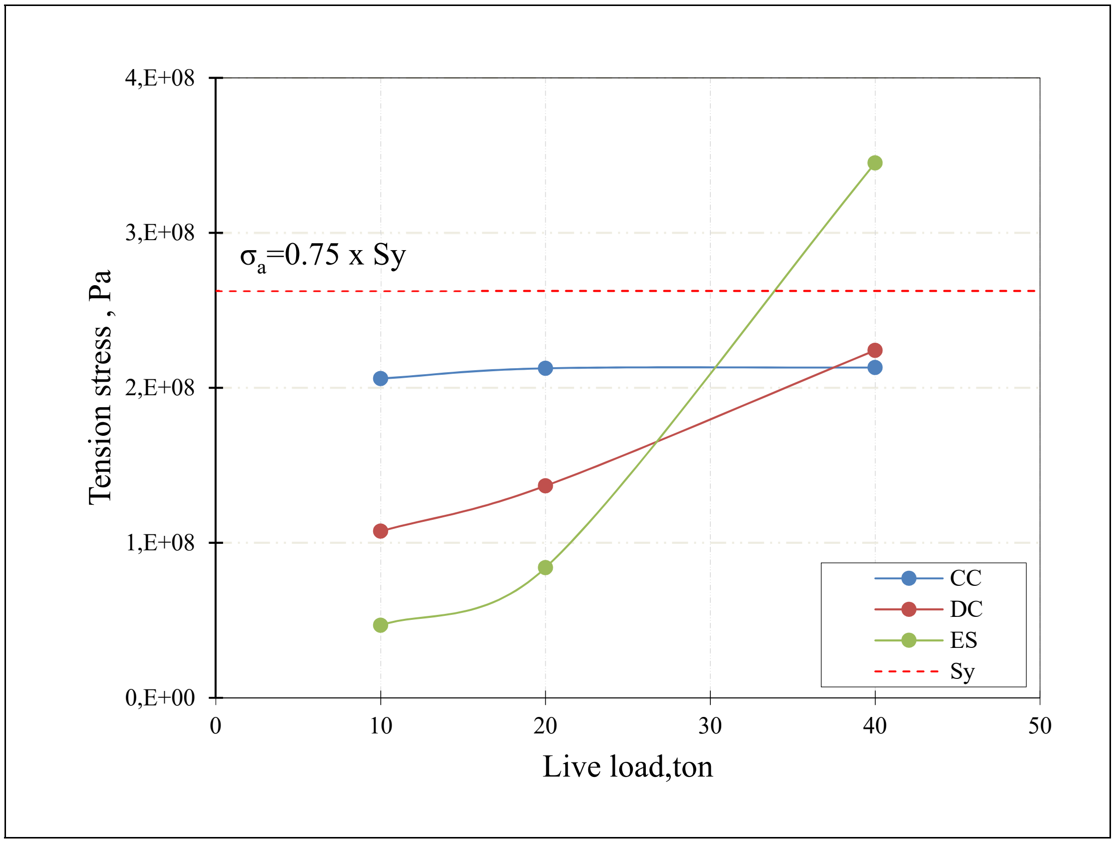


Figure 4.10 The tension stresses vs. live loads for different types of I-beam crane

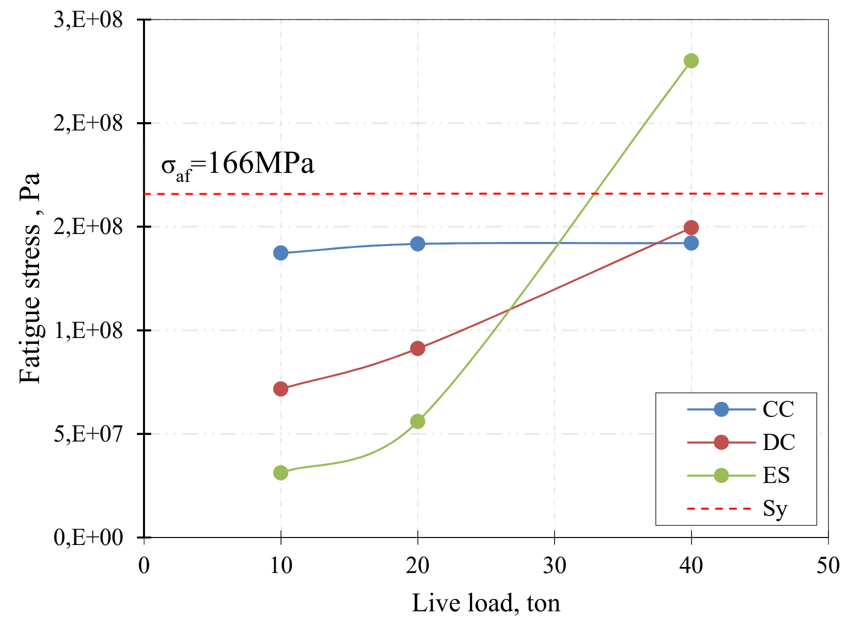


Figure 4.11 The fatigue stresses vs. live loads for different types of I-beam crane

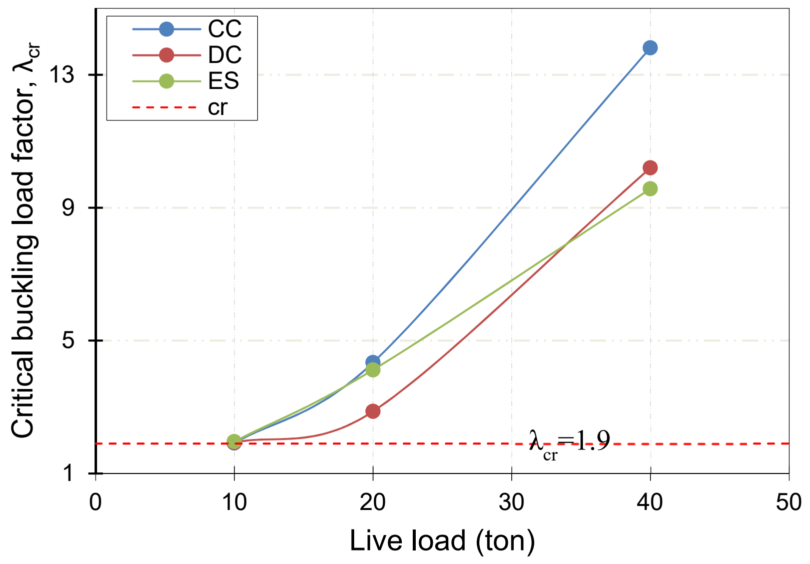


Figure 4.12 The critical buckling load factor vs. live loads for different types of I-beam crane

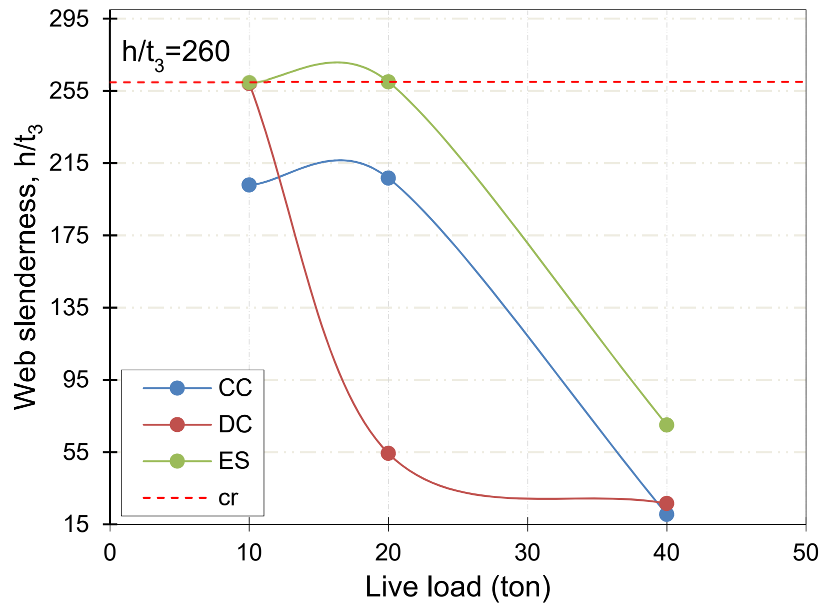


Figure 4.13 The web slenderness vs. live loads for different types of I-beam crane

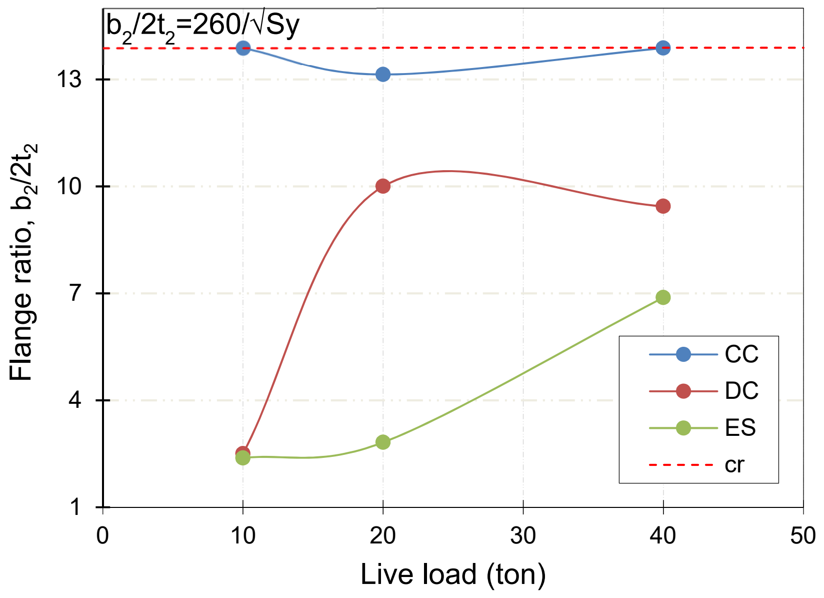


Figure 4.14 The flange ratio vs. live loads for different types of I-beam crane

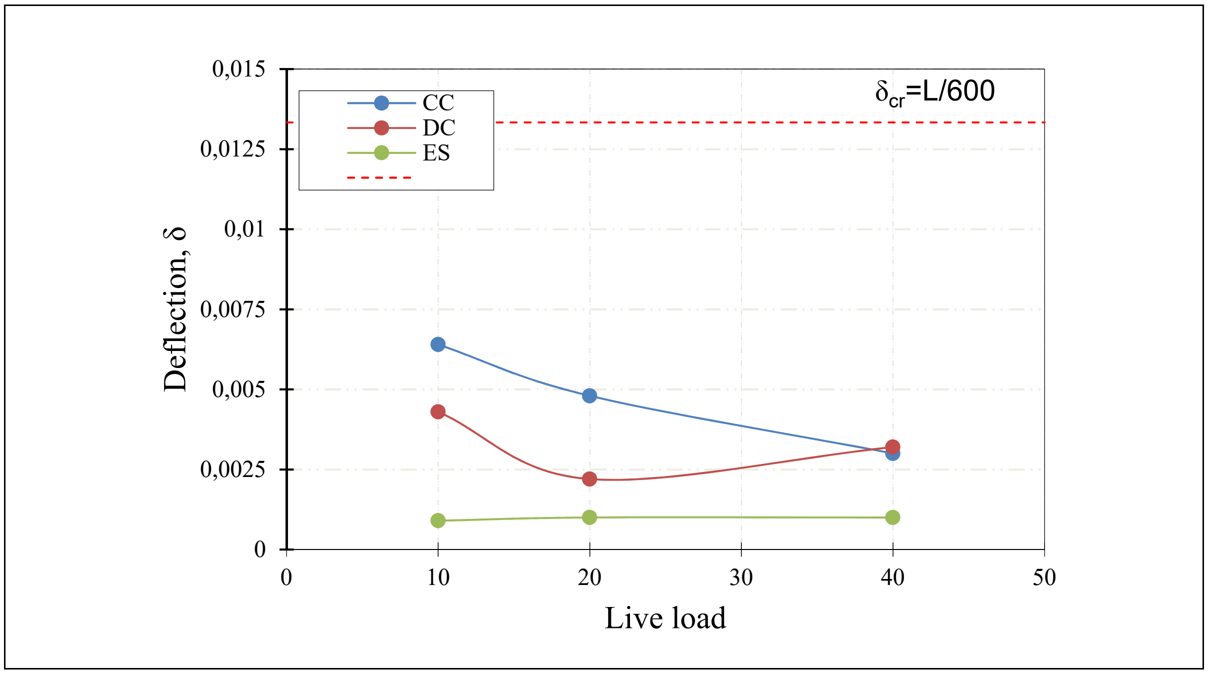


Figure 4.15 The deflection vs. live loads for different types of I-beam crane

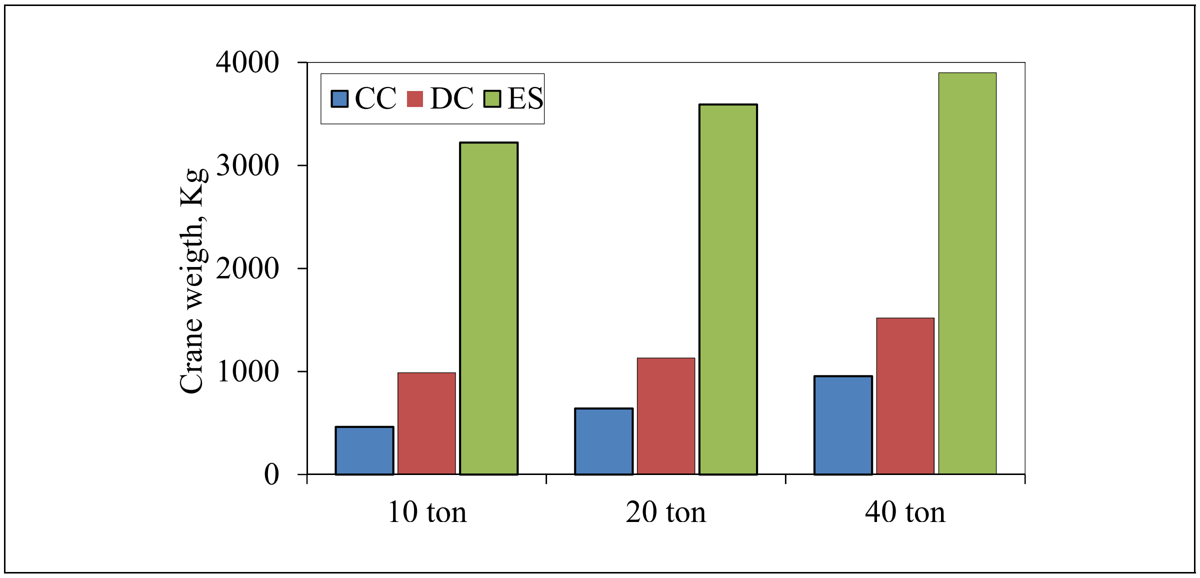


Figure 4.16 The crane weight vs. live loads for different types of I-beam crane

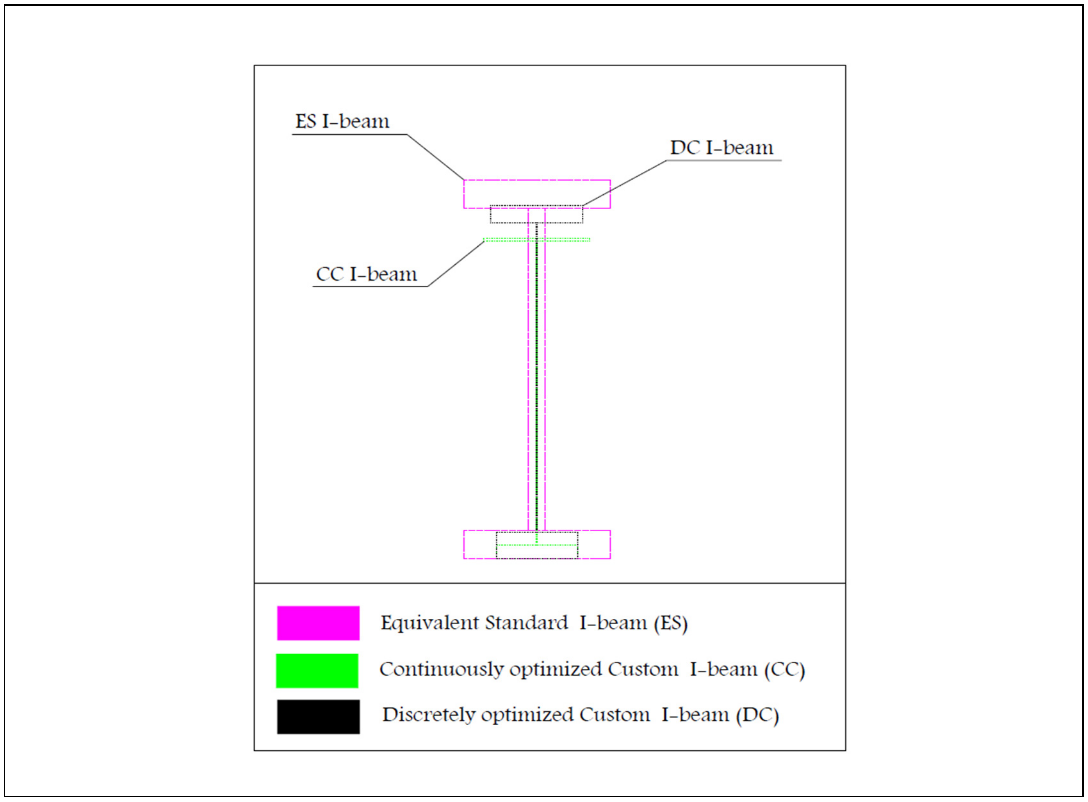

ES I-beam
DC I-beam
CC I-beam
Equivalent Standard I-beam (ES)
Continuously optimized Custom I-beam (CC)
Discretely optimized Custom I-beam (DC)

Figure 4.17 The three different types of I-beam crane for the case of 8 m x10 ton

4.7 Conclusion

A new variant of the Adapted Discrete Cuckoo Search Algorithm (ADCSA) presented and examined for two different case studies of maximizing the critical buckling load of composite laminated plate and a customized I-beam gantry crane design optimization. The validation results demonstrated a high accuracy of the ADCSA solution at a reasonable cost. Furthermore, the initialization methods experiment conducted here illustrated a slight effect of the initial population generation on the performance ADCSA. The use of the LHC sampling approach improved the reliability slightly compared to ADCSA initialized using DUD or Hybrid DUD-LHC. The results obtained for the customized I-beam gantry crane shown that the crane cross-section profile of the discrete optimization approach followed the same configurations pattern obtained using a continues optimization approach. It always shows narrow and thick lower flange, wider and thinner upper flange and tall and very thin web. Additionally, the saving in the cross-section area is noticeable compared to the equivalent standard I-beams. Eventually, the proposed ADCSA has been applied for two different structural optimization problems so far and examining it for other engineering problems could be prospective work. Furthermore, investigating the different initialization methods on the proposed algorithm to find better performance deserves a try.

CHAPTER 5

ENHANCED HYPER CUBE FRAMEWORK ACO FOR STRUCTURAL COMBINATORIAL OPTIMIZATION PROBLEMS

A.Ahmid [a] , T. M. Dao [b] and V. N. Lê [c]

[a,b,c] Department of Mechanical Engineering, École de Technologie Supérieure, 1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper submitted to Elsevier Computers & Structures journal, December 2020

5.1 Abstract

Many structural combinatorial optimization problems are hard to solve within the polynomial computational time or NP-hard problems. Therefore, developing new optimization techniques or improving existing ones still grab attention. This paper presents an improved variant of the Ant Colony Optimization meta-heuristic called Enhanced Hyper Cube Framework ACO (EHCFACO). This variant has an enhanced exploitation feature that works through two added local search movements of insertion and bit flip. In order to examine the performance of the improved meta-heuristic, a well-known structural optimization problem of laminate Stacking Sequence Design (SSD) for maximizing critical buckling load has been used. Furthermore, five different ACO variants were concisely presented and implemented to solve the same optimization problem. The performance assessment results reveal that EHCFACO outperforms the other ACO variants and produces a cost-effective solution with considerable quality.

Keywords: Combinatorial optimization; Ant Colony Optimization (ACO); Buckling load factor; Composite laminate.

5.2 Introduction

Combinatorial optimization is devoted to the mathematical process of searching for optimal solution (maxima or minima) of an objective function with a discrete domain of decision variables. The possible number of solutions for a combinatorial optimization problem is equal to $[D]^n$, where D is the discrete design domain vector and n represents the number of design variables (R. Le Riche & Haftka, 1993). Therefore, the optimization problem becomes more computationally difficult to be solved when the number of design variables increases. Accordingly, many combinatorial optimization problems are hard to solve within deterministic polynomial time (or NP-hard). A Travelling Salesman (or TSP) is a typical example of this type of optimization problem where the number of cities to be visited is given and the shortest path is needed to be determined (França, Sosa, & Pureza, 1999). As the number of cities increases, the number of possible solutions increases too and this leads to the computational complexity of the problem, where it is not possible to enumerate all these solution possibilities with the limited computation resources, such as memory size or processor speed. Hence, to solve such problems, many optimization techniques have been developed.

The Ant Colony Optimization (ACO) algorithm demonstrated a significant performance improvement in solving NP-hard combinatorial optimization problems. The Traveling Salesman Problem (TSP) is a good example of such problems and it is solved using an early version of ACO (M Dorigo, 1991). The improvements in subsequent ACO algorithms focused on enhancing the algorithm variants to yield better searching and computational performance. As a result of the improved algorithm performance, many new applications of ACO appeared, such as the probabilistic TSP using estimation based ACO in Weiler, Biesinger, Hu, and Raidl (2015) work. Gambardella and Dorigo (2000) used a hybrid ACO with a novel local search tool to solve the Sequential Ordering Problem (SOP). Zheng, Zecchin, Newman, Maier, and Dandy (2017) introduced a novel approach of ACO to optimize a water distribution system design. Furthermore, the optimization of the space truss design, using improved ACO, is demonstrated by A Kaveh and Talatahari (2010).

In the field of Stacking Sequence Design (SSD) of composite laminate, ACO has been used to determine the optimal stacking sequence design for maximizing the critical buckling load factor or natural frequency (Rubem Matimoto Koide & Luersen, 2013). Aymerich and Serra (2008) examined the ACO performance in solving stacking sequence design problem and he compared a modified standard ACO performance with Genetic Algorithm (GA) and Tabu Search (TS).He found that ACO performed much better than GA and TS in terms of solution cost and quality. Rama Mohan Rao (2009) presented a hybrid ACO-TS algorithm to optimize the stacking sequence of a composite laminate subjected to bidirectional compression loading. He concluded that ACO is an effective optimization technique if combined with an appropriate local searching tool. Mark W. Bloomfield et al. (2010) conducted a comparison study of three meta-heuristics of GA, ACO, and Particle Swarm Optimization (PSO) to determine the optimal stacking sequence composite laminate. Based on the results of this comparison study, ACO found to outperform GA and PSO algorithms in the field of stacking sequence design (SSD). This remarkable performance of ACO in solving such NP-hard combinatorial optimization is expected where it designed to solve discrete optimization problems (França et al., 1999).

An investigation of five different ACO variants in the field of composite laminate stacking sequence design has been carried out in the current study. Besides, a new optimization approach has been proposed to solve this problem. The new approach uses an improved version of a well-known ACO algorithm variant called Hyper Cube Framework ACO (HCFACO). A popular NP-hard combinatorial optimization problem of composite laminate stacking sequence has been considered to demonstrate the new algorithm performance (R. Le Riche & Haftka, 1993). Furthermore, a performance assessment criterion has been developed using different measures, such as performance rate, reliability, normalized price, and Fitness–Distance correlation. The proposed EHCFACO algorithm performance has been compared with other ACO algorithm variants.

The rest of this paper is structured as follows: firstly, it presents a review of standard ACO followed by a brief description of other considered ACO variants; secondly, the new optimization approach is explained in detail; then, the performance assessment criterion is introduced, followed by the numerical experiments section; next, the results of the performance survey are discussed; finally, the paper concludes with a summary of the study's research contributions, limitations, and prospective directions for future research.

5.3 Ant Colony Optimization Algorithms (ACOs)

M Dorigo (1991) developed the basic Ant Colony Optimization (ACO) algorithm, or Ant System (AS), which is a metaheuristic approach inspired by the collaborative work of ants in finding the source of food. The ants cooperate to find the best possible path from their colony to the food source. During their searching tour the ants communicate by depositing a certain amount of a substance, called the pheromone, on their way to the food site. The following group of ants tends to follow the paths with higher pheromone concentration. Over time, the less selected paths gradually lose their information due to the pheromone evaporation process.

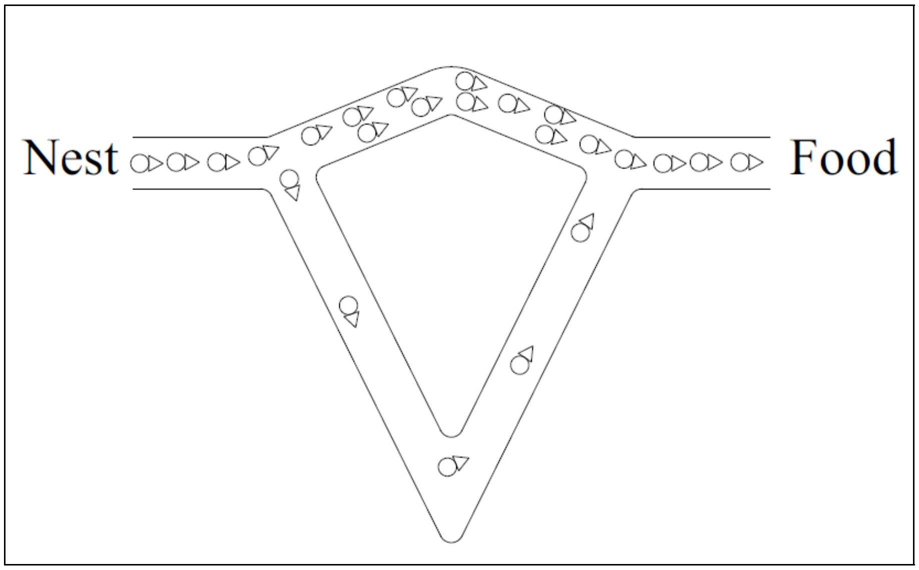


Figure 5.1 Cooperative search of ants by pheromone trails

The virtual ants travelling and selecting paths can be interpreted as a probabilistic selection of certain nodes, which they are part of the solution, in the path based on the pheromone value. The ACO general procedure is illustrated in Algorithm 5.1.

Algorithm 5.1 Ant Colony Optimization procedure

```
Initialization
While (termination criteria not satisfied) Do
    Construct Solutions Table by Ants
    Local Search (optional)
    Global Pheromones Updating
End ACO algorithm
```

To understand the mathematical interpretation of ACO, there is a need to go through each step of the ACO procedure shown in Algorithm 5.1. ACO starts with initial values of the pheromone trail τ_0 , set to a small value for all ant trails as this gives all nodes j of the design variable i, an equal probability of selection. Next, each ant starts to construct its own solution by applying the rule of selection, which has the following general form:

$$p_{ij}^{(k)} = \frac{\tau_{ij}^{\alpha} . \eta_{ij}^{\beta}}{\sum_{j \epsilon N_i^k} \tau_{ij}^{\alpha} . \eta_{ij}^{\beta}} \text{ , } \forall\, j \in N_i^k \tag{5.1}$$

$p_{ij}^{(k)}$ represents the probability of selecting the path $i\,j$ for the k^{th} ant, τ_{ij} is the updated pheromone trail, η_{ij} denotes the value of heuristic information for each feasible solution s, $N_i^{(k)}$ indicates the neighbourhood nodes of the k^{th} ant, when it located at node i and α, β are the amplification parameters for pheromone trials and the influence of heuristic information on the algorithm behaviour respectively (M. Dorigo, Birattari, & Stutzle, 2006). At the end of each tour all the pheromone trails are updated through two steps of pheromone evaporation and depositing, according to the following formula:

$$\tau_{ij}^{k(t+1)} = (1 - \rho).\, \tau_{ij}^{k(t)} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k(t)} \tag{5.2}$$

where ρ is the evaporation rate, $\rho \in (0,1]$, and $\Delta\tau_{ij}^{k(t)}$ is the amount of deposited pheromone by ant $k(t)$ that could be determined as :

$$\Delta\tau_{ij}^{k(t)} = Q/_{L_{k(t)}} \tag{5.3}$$

where Q is a constant and $L_{k(t)}$ represents the distance travelled by ant $k(t)$. Eq.(5.3) is the basic form of the pheromone trail updating which used to solve TSP optimization problem and it could be implemented in more general form:

$$\Delta\tau_{ij}^{k(t)} = \begin{cases} \xi.\dfrac{f_{worst}}{f_{best}}, & if\ i,j\ \in global\ best\ solution \\ 0, & otherwise \end{cases} \tag{5.4}$$

where f_{worst} , f_{best} are the worst and the best values of the objective function f obtained by N ants in tour t and x is the global pheromone scaling factor (S. S. Rao, 2009). Eventually, the ACO loop continues until one of the termination conditions is met.

In SDD optimization problem, the thickness of each ply (equivalent to the distance between the cites in TSP) is assumed to be constant, so the heuristic information value, η_{ij}, will be constant all over the ant tours t which simplifies the probability of selection, in Eq. (5.1), into:

$$p_{ij}^{(k)} = \frac{\tau_{ij}^{\alpha}}{\sum_{j\epsilon N_i^k}\tau_{ij}^{\alpha}} \text{ , } \forall\, j \in N_i^k \tag{5.5}$$

Local search

The procedure of the ACO algorithm includes the option of improving the intensification feature of the ACO algorithm by adding some local search algorithms or movements that could improve the search of the solution neighbourhood (França et al., 1999) .

5.3.1 Elitist Ant Colony (EACO)

Gambardella and Dorigo (2000) introduced an improved version of the ACO algorithm that uses the elitism strategy. The idea behind this strategy is a reinforcement of the best solution path found once the algorithm is initialized. The rule of pheromone updating for EACO is written as follows:

$$\tau_{ij}^{k(t+1)} = (1-\rho).\tau_{ij}^{k(t)} + \sum_{k=1}^{n} \Delta\tau_{ij}^{k(t)} + e.\Delta\tau_{ij}^{k(t_{best})} \tag{5.6}$$

where

$$\Delta\tau_{ij}^{k(t_{best})} = \frac{f_{best}}{\sum_{k=1}^{n} f_i} \tag{5.7}$$

The reinforcement of selection probability of the best path (t_{best}) occurs by adding the value of $e.\Delta\tau_{ij}^{k(t_{best})}$ where e is the weighting parameter and it represents the number of elitist ants (Rama Mohan Rao, 2009).

5.3.2 The Rank-Based Ant Colony Optimization (RBACO)

Bullnheimer, Hartl, and Strauss (1997) proposed a new extension of the ACO that enhances the performance of the original EACO by ranking the ants based on their path length. The deposited value of pheromone decreases according to its rank index, μ. Moreover, only the best ants, σ, will be updated which prevents the over concentration of pheromones on local optima paths chosen by other ants . Hence, the pheromone updating rule of RBACO is:

$$\tau_{ij}^{k(t+1)} = \rho.\tau_{ij}^{k(t)} + \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu} + \sigma.\Delta\tau_{ij}^{k(t_{best})} \tag{5.8}$$

5.3.3 Max-Min Ant Colony (MMACO)

Previous ACO algorithms used the strategy of reinforcing only the best-found paths. This strategy could cause the excessive increase of pheromone values on optimal local paths causing all other ants to follow this path. To overcome this drawback, Stützle and Hoos (2000) proposed a modified version of ACO that limits the pheromone values to a specific interval, $[\tau_{min};\ \tau_{max}]$. In addition, the initialization of pheromone value is set to the upper limit of the pheromone interval, with a small evaporation rate to increase the algorithm search diversification. The pheromone rule is:

$$\tau_{ij}^{k(t+1)} = \rho.\tau_{ij}^{k(t)} + \Delta\tau_{ij}^{k(t_{best})} \tag{5.9}$$

and $\tau_{ij}^{k(t)} \in [\tau_{min};\ \tau_{max}]$

where $[\tau_{min};\ \tau_{max}]$ values are determined by the following formulas:

$$\tau_{max} = \frac{1}{(1-\rho)}.\frac{f_{worst}}{f_{best}}$$

$$\tau_{min} = \frac{\tau_{max}.(1-\sqrt[n]{P_{worst}})}{\left(\frac{n}{2}-1\right).\sqrt[n]{P_{worst}}}$$

where p_{best} denotes the probability of the best solution, it has a value greater than 0, while n represents the number of ants.

5.3.4 Best-Worst Ant Colony (BWACO)

Zhang, Wang, Zhang, and Chen (2011) presented BWACO as an extension of MMACO, where the algorithm exploitation capability benefits from both, best and worst solutions. During the search tour, the pheromone trail update uses the positive return of the best solution and the negative one generated by the worst solution. The pheromone updating rule can be written as:

$$\tau_{ij}^{k(t+1)} = \left[\rho.\tau_{ij}^{k(t)} + \Delta\tau_{ij}^{k(t_{best})}\right]_{\tau_{min}}^{\tau_{max}} - \lambda.\Delta\tau_{ij}^{k(t_{best})} \tag{5.10}$$

where, λ is a coefficient that has value within [0,1] interval and it could be noticed that BWACO became MMACO if $\lambda = 0$.

5.3.5 Hyper Cube Framework ACO (HCFACO)

The different algorithms of ACO build a limited hyperspace of the pheromone values. The Hyper Cube Framework of ACO algorithms, proposed by Blum in 2001, generates a binary convex hull hyperspace from pheromone values for the feasible solutions. In other words, the values of the pheromone vector, $\tau = [\tau_{1,}\tau_2, \tau_3, \ldots.., \tau_n]$, are limited to the interval $[0,1]$, and this is carried out by changing the pheromone update rule. The following formula expresses the rule of pheromone updating in HCFACO:

$$\tau_{ij}^{k(t+1)} = (1-\rho).\tau_{ij}^{k(t)} + \rho.\sum_{k=1}^{n} \varDelta\tau_{ij}^{k(t)} \qquad (5.11)$$

where:

$\varDelta\tau_{ij}^{k(t_{best})} = \frac{f_{best}}{\sum_{k=1}^{n} f_i}$, and n is the number of ants follow the same best path.

HCFACO algorithms overcome the undesirable problem of different behaviour of standard ACO algorithms when the same objective function is scaled, which affects the algorithm robustness. Also, it reduces the search effort and improves the algorithm search diversification (França et al., 1999) . Lastly, it is worthwhile to mention that the HCF update rule is not limited to standard ACO algorithm (or Ant System AS) as it can also be used with MMACO, where the maximum and minimum limits of MMACO pheromone trail are set to be 0 and 1 respectively (Blum & Dorigo, 2004).

5.4 Enhanced Hyper Cube ACO Algorithms

Dorigo experimentally observed that using local search techniques can improve the overall performance of the ACO (M Dorigo, 1997; França et al., 1999). Local search can be carried out by hybridizing the ACO with local search algorithms such as Tabu search or using permutation operators to explore the solution neighbourhood (Aymerich & Serra, 2008;

Katagiri, Hayashida, Nishizaki, & Guo, 2012). The commonly used operators in SSD optimization problem are two-points permutation and swap. Two-points permutation means selecting two bits in the solution string and reversing the order of the bits in between (R. Le Riche & Haftka, 1993). The swap operator is used to switch the position of two randomly selected bits of the solution string (Jing, Fan, & Sun, 2015).

The HCFACO algorithms presented here adopted two other permutation operators to perform the algorithm enhancement. The first operator is called a single point mutation, which is used successfully with Permutation Genetic Algorithm (R. Le Riche & Haftka, 1993). The second operator is inspired by using one of the Tabu Search movements named the insertion (França et al., 1999). The proposed Enhanced HCFACO procedure for standard ACO (Ant System AS) and max-min ACO is listed in Algorithm 5.2. The Enhanced HCFACO Algorithms starts by defining the standard ACO parameters such as the maximum number of iterations $(Iter_{max})$, number of ants (n_{Ants}) , number of design variables (N_V), the initial pheromone trail (t0) and evaporation rate (r). In addition, the solution convergence rate counter is imposed $[Iconv_{max}]$ and its value determine whether the convergence rate is slow or fast. When the ACO loop starts, all solution edges have the same deposited pheromone trail τ_0 , which gives all nodes the same probability of selection to be a part of the feasible solution. The artificial ants, $k = 1{:}\, n_{Ants}$, start building the solution table, $S_i(n_{Ants}, N_V)$, by randomly choosing a node di on their way to build the solution vector $S_i(k, N)$. Next, the evaluation of the solutions table is carried out by calling the objective function, and the obtained values are stored in $f(ige, S_i(1{:}\, n_{Ants}, N_V\ ))$ matrix. The best solution of the stacking sequence design has the maximum value of the objective function listed in $f(ige, S_i)$ matrix of the current iteration. The best solution of each iteration ige is stored in the best solution matrix $S^*(ige)$. Thereafter, the global pheromone trail update is performed as described in the Hyper Cube Framework of ACO in Eq.(5.11).

The local search actions are enforced as soon as the best solution of the current tour is determined. Following this, a comparison of the generated solutions with the best solution obtained so far is made. The best solution matrix is then updated if any improvement is detected. Finally, the HCFACO loop continues until the termination criteria are met. The global optimal solution is determined as the best solution matrix member with the maximum value of the objective function.

Algorithm 5.2 Enhanced HCFACO procedure

```
Initialization
    - ACO parameters:$[n_{Ants}, Iter_{max}, N_V, \alpha, \rho, \tau_0]$
    - Convergence rate counter: $[Iconv_{max}]$
    - initialize ACO loop counter: $[ige = 0]$

While (termination criteria not satisfied) Do
                        $ige = ige + 1$
    - Solution table construction:
                        $P_i = \tau^{\alpha}$
            $Si(n_{Ants}, N_V) = RouletteWheelSelection(\frac{P_i}{\sum P_i})$
    - Solution evaluation:
                    $f(ige, Si(1 : n_{Ants}, N_V))$
                    $S^*(ige) = max(S_i(ige))$
                $if\ S^*(ige) < max(S^*(1 : ige - 1))$
                    $S^*(ige) = max(S^*(1 : ige - 1))$
                end
- Apply Local Search:
                        • Single point mutation.
                        • Insertion.
                $if\ S^*_{LS}(ige) > S^*(ige)$
                    $S^*(ige) = S^*_{LS}(ige)$
                $end$
Global pheromone trail updating: Eq. (5.11)
End (While)
Optimal solution $S_{opt} = max(S^*(1 : ige))$
End Enhanced HCFACO algorithm
```

5.5 Performance Evaluation

The time required by an algorithm to find the global optima is widely used to evaluate its performance (Talbi, 2009). However, a single performance measure cannot reflect the effectiveness of the algorithm in exploring the design space or determining solution quality. In the current study, three different groups of performance measures have been applied to ensure a fair evaluation of the proposed algorithm.

5.5.1 Computational Effort

In addition to the elapsed time, literature has shown that other measures can be used to measure computational effort. The first is the Price PS, which is defined as the number of objective function evaluations within a search run and reflects the computational cost of the search process. The second measure is Practical Reliability (PR) and, it is defined as the percentage of runs that achieve Practical Optima (PO), at a specific run. Practical optima is defined as the solution with 0.1% error in the best possible solution (R. Le Riche & Haftka, 1993). The last is the normalized price nPS ,which is defined as the ratio of price and practical reliability (Kogiso, Watson, Gürdal, & Haftka, 1994; R. Le Riche & Haftka, 1993; Malan & Engelbrecht, 2014). Finally, the Performance Rate measure P_{rate}, is also considered to link the computation effort with the number of function evaluations (Talbi, 2009).

$$P_{rate} = \frac{number\ of\ successful\ runs}{\begin{pmatrix} number\ of \\ function \\ evaluations \end{pmatrix} . \begin{pmatrix} total\ number \\ of\ runs \end{pmatrix}} \tag{5.12}$$

5.5.2 Solution Quality

The solution quality of an algorithm can be measured by determining the absolute error between the current solution and the best-known global solution (Kogiso et al., 1994; R. Le Riche & Haftka, 1993; Malan & Engelbrecht, 2014).

$$Q = \left| \frac{S^* - S_{opt}}{S_{opt}} \right| .100 \tag{5.13}$$

5.5.3 Fitness Landscape Analysis

The design space of a combinatorial optimization problem can significantly affect the search performance of an algorithm. The notion of Fitness-Landscape appeared in literature as an answer to the question of ”what the design space looks like?”. The Fitness-Landscape is defined by the feasible solutions set, the objective function (fitness) and the structure of the solution neighbourhood. To find the connection between the Fitness Landscape and the problem hardness, T. Jones and Forrest (1995) introduced a Fitness Landscape - Distance Correlation (FDC) to determine the hardness of optimization problems to be solved using Genetic Algorithm (GA). The distance mentioned here is defined as the number of movements that should be imposed on a Solution S_i to eliminate dissimilarity with the optimal solution S_{opt} . The proposed correlation by Jones is computed using the correlation factor, r :

$$r(F,D) = \frac{CFD}{\sigma_F . \sigma_D} \tag{5.14}$$

where CFD indicates the $Cov(F,D)$ and σ_F, σ_D are the standard deviation of F and D respectively. The values of the correlation coefficient r are limited to interval $[-1,1]$ where negative values are desirable for maximization and indicate better searching performance. Finally, using the scattering of fitness versus the distance to the global optima can reveal valuable information about FDC of the optimization problem solved by an algorithm (Malan & Engelbrecht, 2014; Stützle & Hoos, 2000).

5.6 Numerical Experiments

To demonstrate the performance of the new approach we selected a well-know NP-hard combinatorial optimization problem in filed of composite laminated structures. The optimization objective is maximizing the critical buckling load of composite laminated plate exposed to bidirectional compression loading. The decision variables are the fiber orientation of each composite layer (lamina) which form the optimal stacking sequence of the laminate (a group of layers). To employ ACO as an optimization algorithm for SSD optimization problem, there is a need to understand specific problem characteristics such as solution representation, constraints, and objective function formulation. In meta-heuristic algorithms, the solution (stacking sequence) takes the form of a bit string that consists of a combination of plies with the available angle fiber orientations (e.g. $0°, \pm 45°$ and $90°$).The different solutions have integer coding with 1,2 and 3 numbers, which represent the three possible fiber orientations, respectively. For instance, the laminate with $[2132231]_s$ stacking sequence describes the laminate of $[\pm 45,\ 0_2, 90_2, 45, \pm 45, 90_2, 0_2\ ]s$ fiber orientations.

The simplicity of using an integer representation along with significant performance gains, made it the most widely used method in meta-heuristic optimization algorithms for composite laminated design. The buckling load factor lb for simply supported rectangular laminated plate subjected to bi-axial loading is determined as follows:

$$\lambda_b(\mathrm{p,q}) = \pi^2 \frac{\left[D_{11}\left({}^{p}/_{a}\right)^4 + 2(D_{12} + 2D_{66})\left({}^{p}/_{a}\right)^2 + D_{22}\left({}^{q}/_{b}\right)^4\right]}{\left({}^{p}/_{a}\right)^2 N_x + \left({}^{q}/_{b}\right)^2 N_y} \tag{5.15}$$

where D_{ij} denotes the bending stiffness, N_x is the axial loading in x-direction, N_y is the axial loading in y-direction, p and q are the number of half waves in x, y directions. The critical buckling load factor λ_{cb} is defined as the minimum obtained value of $\lambda_b\ (p, q)$. The critical values of p and q are linked to different factors such as laminate material, a number of plies, loading conditions and the plate aspect ratio. In uniaxial loading and simply supported plate, the critical buckling load is happening when $p\ = 1$ whereas in biaxial the critical buckling loads it needs to be determined as the minimum value of $\lambda_b(p, q)$ (R. Le Riche & Haftka, 1993; Rama Mohan Rao, 2009). Finally, the constraints in stacking sequence optimization with constant laminate thickness t could be imposed as follow:

- Symmetry constraint is enforced by optimizing half of the laminate.
- Balancing constraint is enforced by selecting θ_2 for the standard fiber orientation set of 0 , ± 45 ;and 90.

- Only $N/4$ ply orientations are needed to describe laminate as a result of balancing constraints. Contiguity constraint is handled by using the penalty parameter (β).
- And the critical buckling load factor objective function f_{obj} could be formulated as:

$$f_{obj} = (1 - \beta).\max(\lambda_{cb}(p, q)) \tag{5.16}$$

To compare the performance of the proposed algorithm alongside the other ACO algorithms; we implemented all the algorithms presented here using MATLAB R2019b software. The benchmarking problem from the literature of stacking sequence design optimization is accredited to Le Riche and has been used by previous studies (Jing et al., 2015; R. Le Riche & Haftka, 1993). The original problem describes a simply supported plate subjected to an in-plane biaxial loading as shown in Figure 5.2.

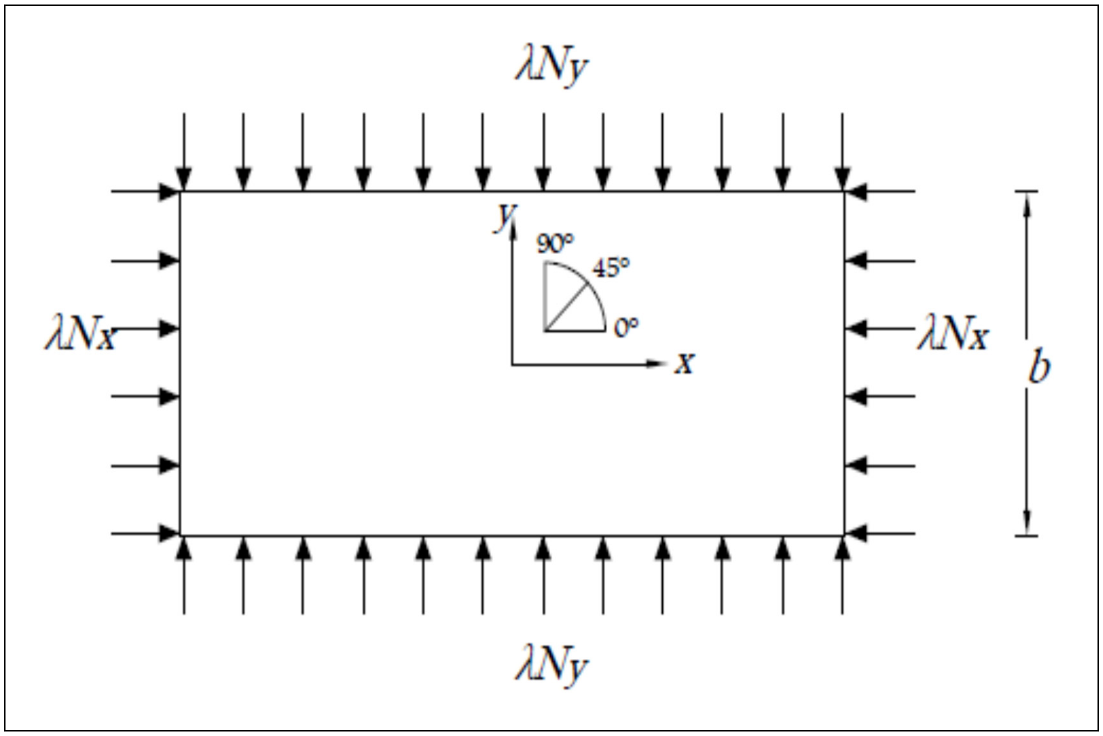


Figure 5.2 Simply supported plate subjected to biaxial loading

The thickness of each ply t_i is assumed constant, and the ply orientations are limited to 0 , ±45 and 90 sets of angles. The number of plies N_L is constant. The required properties, dimensions, and loading conditions are listed in Table 5.1 and Table 5.2. The objective function is set to maximize the critical buckling load. The constraints are integrated into the solution (e.g., balanced laminate, symmetrical, etc.). The implemented ACO algorithms were executed on the same computer for the same number of experiments; $N_{exp} = 200$. This number is used to overcome the stochastic behaviour of meta-heuristic algorithms (R. Le Riche & Haftka, 1993). Furthermore, this number of experiments is conducted over ten different random generating seeds of $301, 2,50,75, 111, 200,167, 225 ,11$ *and* 25. Then the average of the performance measures values was used in the comparison of different ACO algorithms.

Table 5.1 Graphite-epoxy lamina's properties

$E_1(GPa)$	$E_2(GPa)$	$G_{12}(GPa)$	v_{12}
127.59	13.03	6.41	0.3

Table 5.2 Graphite-epoxy lamina's geometrical and loading data

N_L	$t(mm)$	$a(mm)$	$b(mm)$	$N_x(N/m)$	N_x/N_y
64	0.127	508	254	175	1

Lastly, all ACO algorithms were examined at two different levels of convergence rate, slow and fast. The slow rate enforces the algorithm searching loop to stop after 56 iteration without improvement, while the fast rate needs just 10 iterations to be terminated (R. Le Riche & Haftka, 1993).

5.6.1 ACO Parameters Setting

To ensure a fair assessment of the ACO algorithms performance, the following standard ACO parameters were assumed for all implemented algorithms:

number of Ants $n_{Ants} = 25$, the maximum number of iterations $Iter_{max} = 1000$, evaporation rate $r = 0.1$, the parameter of the pheromone trail relative importance $\alpha = 1$, initial one trail $\tau_0 = 0.004$ (except for MMACO and BWACO algorithms were $\tau_0 = 1$). Best solution probability $P_{best} = 0.05$ for MMACO and BWACO Algorithms and lastly the coefficient of worst solution pheromone trail $\lambda = 0.6$ for BWACO algorithm only.

5.6.2 Termination Criteria

All algorithms will stop as soon as one of the following conditions are satisfied:

- If there is no improvement in the solution after 10 (fast rate) or 56 (slow rate) iterations.
- If the number of iterations exceeds 150 and the best solution is equal to the worst solution (means all artificial ants following the same path).
- If a maximum number of iterations have been generated.

5.7 Results

The case study described in the previous section has been optimized using nine different algorithms: standard ACOA, EACO, RBACO, MMACO, BWACO, HCF/EHCF for both ACO and MMACO algorithms. Analysis of the algorithm's performance will be divided into two parts. First, the performance of ACO algorithms with Hyper Cube Framework will be assessed. The second part is dedicated to the comparison of EHCFACO algorithm with the rest of ACO algorithms.

5.7.1 Hyper Cube Framework ACO Algorithms Results Analysis

Referring to section 5.3.5, the Hyper Cube Framework (HCF) can be applied for both versions of the standard ACOA and MMACO. Hence, this part of the analysis is devoted to determining which version of both ACO algorithms, with HCF and EHCF, could exhibit better performance? The performance measures for the original ACOA,MMACO, HCFACO, HCFMMACO, EHCFACO, and EHCFMMACO are listed in Table 5.3. The performance values listed in Table 5.3 reveal that applying HCF to the ACOA has positively affected the overall performance of ACO. The average practical reliability increased by $22 - 56\%$ and the normalized price declined from 51.17 to 36.91 for fast convergence rate and from 181.12 to 94.24 for slow one. The performance rate doubled at slow rate while remain the same for the fast. The FDC correlation coefficient r decreased slightly for both levels of convergence.

Table 5.3 The performance measures of Hyper Cube Framework ACO algorithms

Performance measure	**Convergence rate**	**ACOA**	**HCFACO**	**EHCFACO**	**MMACO**	**HCFMMACO**	**EHCFMMACO**
Elapsed time, t_s, min	Slow	0.87	.01	2.16	1.12	0.96	1.34
	Fast	3	4.15	6.86	4.62	6.68	4.39
Reliability %	Slow	35.71	77.1	89.6	16	13.17	54.36
	Fast	36.45	93.15	98.95	87.7	91.05	98.25
Normalized price, nP_s	Slow	51.17	36.91	28.92	152.23	169.26	52.2
	Fast	181.12	94.24	81.49	118.53	117.3	98.25
Performance rate, P_{rate}	Slow	0.0196	0.0272	0.0347	0.0068	0.0063	0.0206
	Fast	0.0056	0.0106	0.0123	0.0088	0.0090	0.0106
Quality %	Slow	99.69	99.93	99.96	98.28	98.57	99.81
	Fast	99.69	99.97	99.98	99.96	98.57	99.98
Fitness-Distance Correlation, r	Slow	-0.80	-0.67	-0.79	-1	-1	-0.82
	Fast	-0.84	-0.71	-0.73	-0.82	-0.77	-0.75

Further improvement of HCFACO performance is acquired when the proposed local search movements are imposed. The average practical reliability became more than twice of the standard ACO and the normalized price decreases more to hold at 28.92 instead of 51.17 and 81.49 instead of 181.2 for both convergence rates. The performance rate is slightly increased and the FDC correlation coefficient r is partially improved. On the contrary, HCFMMACO performed poorly compared to the original MMACO. However, the performance of MMACO has improved when the Enhanced HCF is applied, but the computational effort became more costly. Based on these results, we conclude that EHCFACO delivers an inexpensive solution with significant performance. Eventually, the solution convergence of the algorithms mentioned above has been plotted Figure 5.3, for the same selected experiment (seeds=75).

5.7.2 Other ACO Algorithms Results Analysis

All ACO algorithms have been successfully found the best-known value of the maximum critical buckling load factor, λ_{cb}. A different samples of optimal SSD obtained by different ACO algorithms were listed in Table 5.4. The solution convergence of EACO,RBACO and BWACO for a selected experiment (seeds=75) has been graphically illustrated in Figure 5.4.

Table 5.4 The optimal stacking sequence for 64 ply laminates subjected to biaxial loading without contiguity constraint ($N_y = N_x = 1\ and\ a/b = 2$)

Algorithm	**Optimal stacking sequence design**	**Critical buckling load factor, λ_{cb}**
EACO	$[2333332333323333]_s$ $[\pm45°/90°_{10}/\pm45^{°}/90°_8/\pm45°/90°_8]_s$	3973.01
RBACO	$[2333323333333332]_s$ $[\pm45°/90°_8/\pm45°/90°_{18}/45°]_s$	
BWCACO	$[3333232323222222]_s$ $[90°_8/\pm45°/90°/\pm45°/90°/\pm45°/90°\pm45_2]_s$	
EHCFACO	$[3333322322232222]_s$ $[90°_{10}/\pm45°_2/90°/\pm45°_3/90°/\pm45°_8]_s$	

It is observed from Table 5.4 that the optimal stacking sequence design followed the same pattern of switching between two groups of 90_2 and ±45 fiber orientations which confirm the results of previous studies(Jing et al., 2015; R. Le Riche & Haftka, 1993). On the other hand, the solution convergence plot in Figure 5.4 illustrate that both RBACO and BWACO algorithms develop gradual search trends on their way to the optima whereas EACO and

EHCFACO algorithms smoothly converge to the global optima. Further, the numerical experiments confirm the fluctuation of ACO algorithms in finding the global optimal solution due to their stochastic nature, as illustrated in Figure 5.5.

According to the introduced performance assessment criteria in section 5.5, the average values of different performance measures of reliability, performance rate, solution quality, normalized price, and searching effort coefficient are determined for EACO, RBACO, BWACO at fast and slow convergence rates. These results, alongside with EHCFACO results, are plotted in Figure 5.6 to Figure 5.12 to provide a sensible comparison of the performance evaluation of the proposed algorithm with other ACO algorithms. The average practical reliability of the algorithms is introduced in Figure 5.6. Both EACO and RBACO algorithms show low practical reliability values, with 10% and 8% respectively, while BWACO presented better value at the slow rate of convergence but it poorly performed at the fast rate. The EHCFACO algorithm exhibited a significant reliability values of 89.6-98.95%. Furthermore, it demonstrated the highest performance rate measure (0.012-0.035), see Figure 5.9. The solution quality results of the algorithms are summarized and depicted in Figure 5.10 which reveals that all ACO algorithms produce an excellent solution quality for this particular case study.

In terms of solution cost, the normalized price results plotted on the scatter chart that illustrated in Figure 5.7. As mentioned before, the normalized price measure reflects the balance between the solution cost and the reliability. So, it is quite clear that EHCFACO outperformed other ACO algorithms. BWACO comes second at slow convergence rate, whereas RBACO and EACO deliver a costly solution.

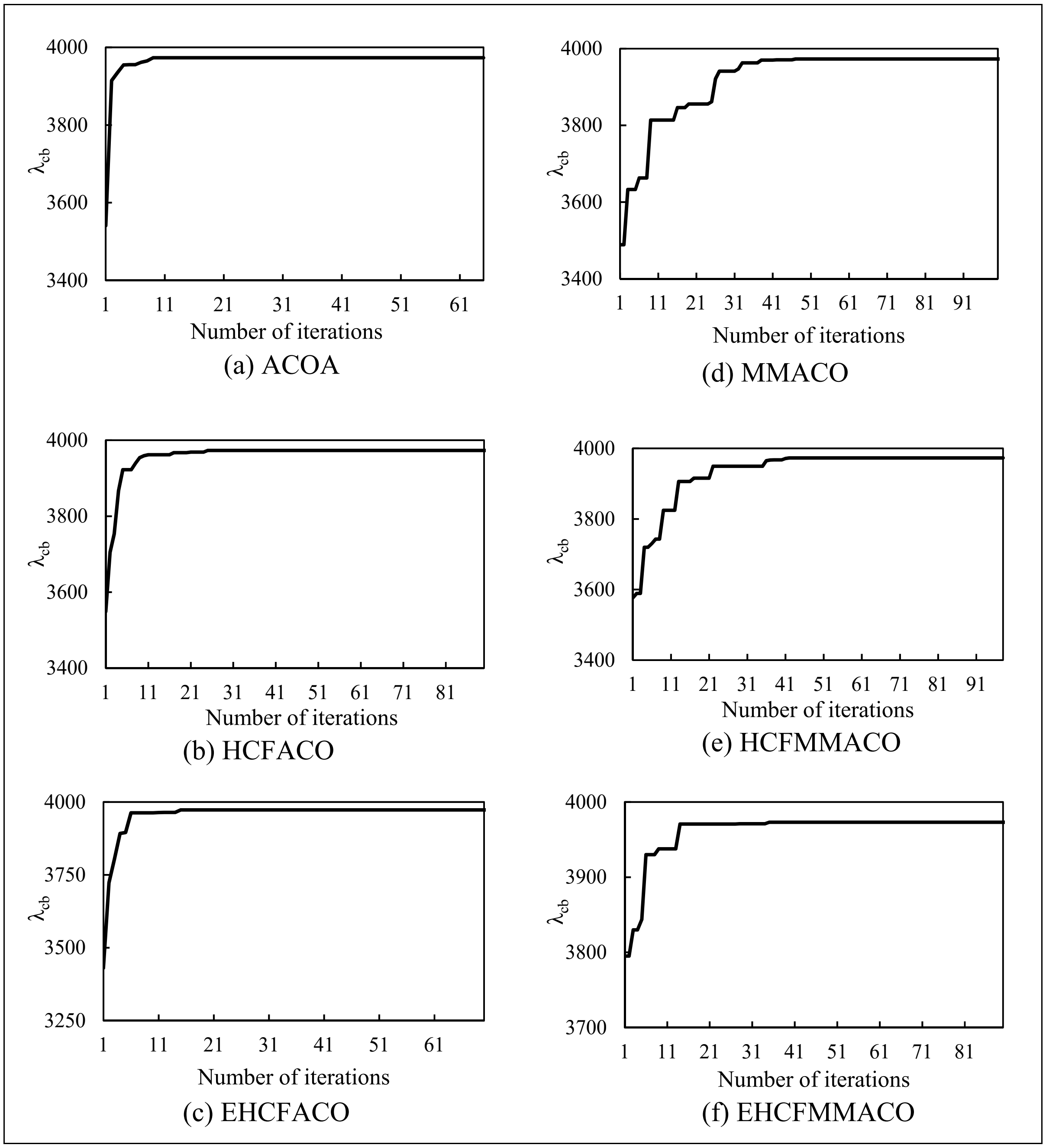
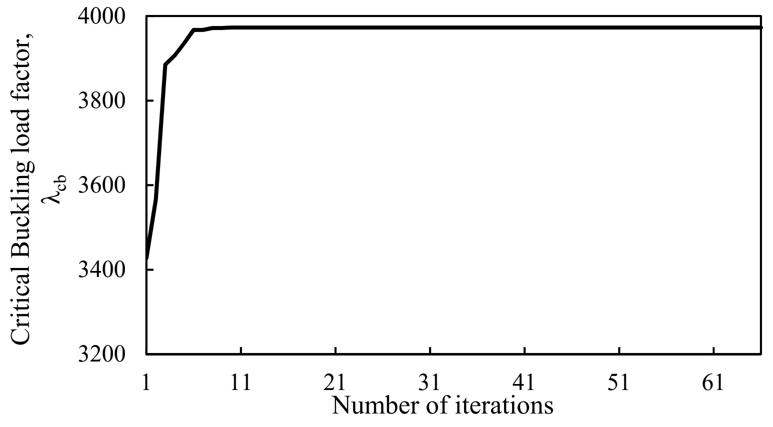
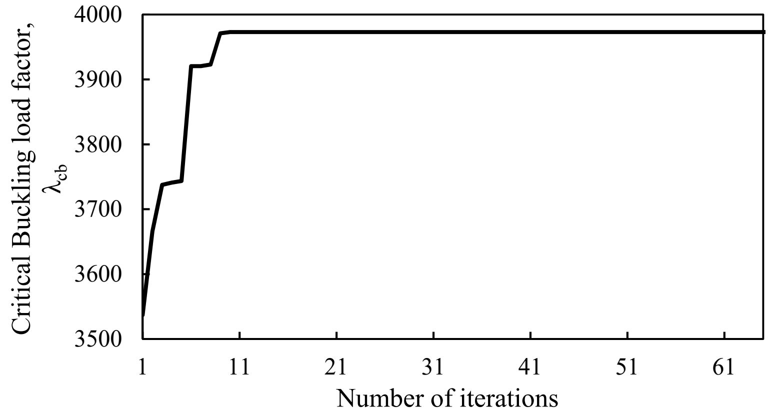


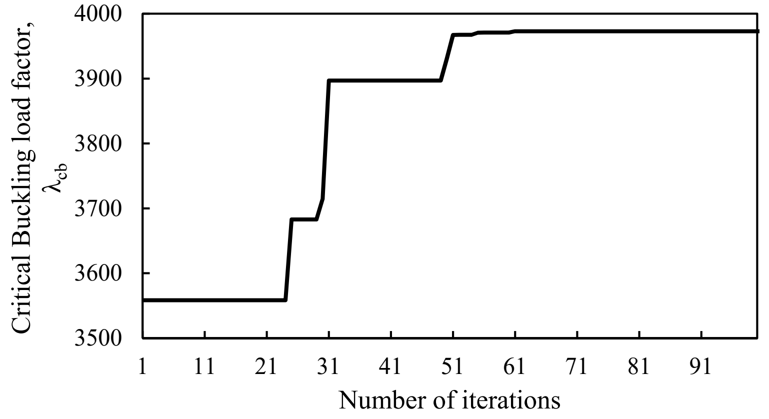
Figure 5.3 Solution convergence of ACO-MMACO and their Hyper Cube Framework variants



(a) EACO

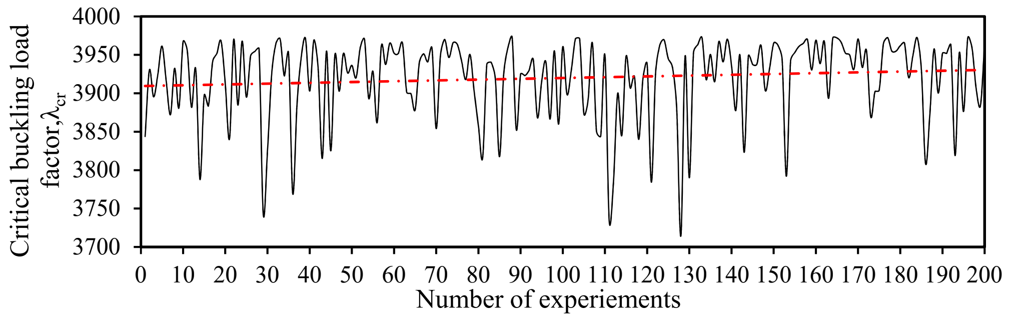


(b) RBACO

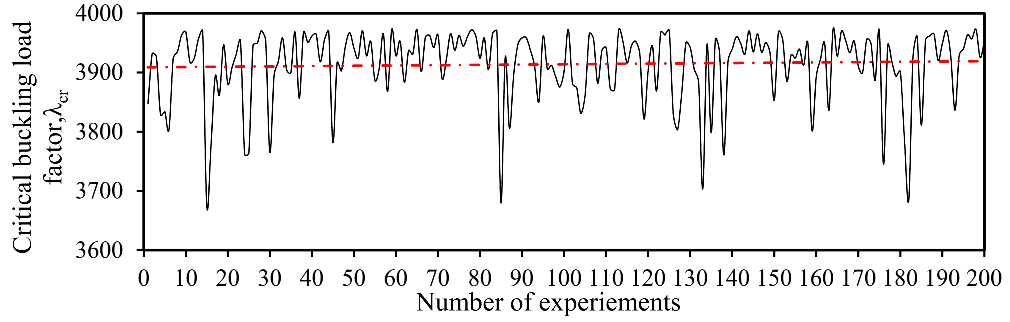


(c) BWACO

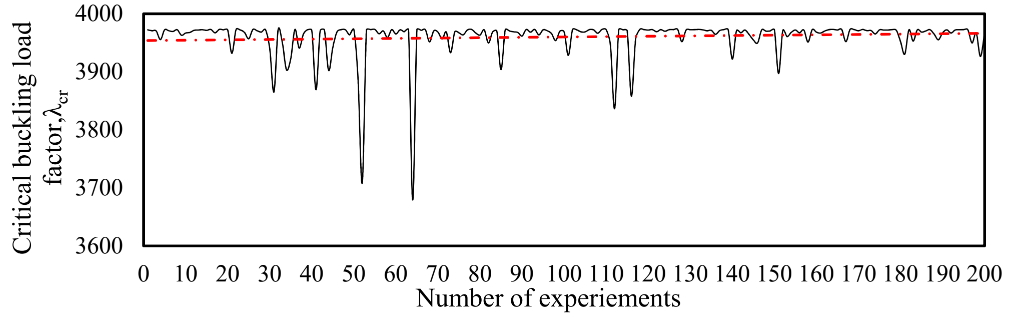
Figure 5.4 Solution convergence of EACO, RBACO and BWACO



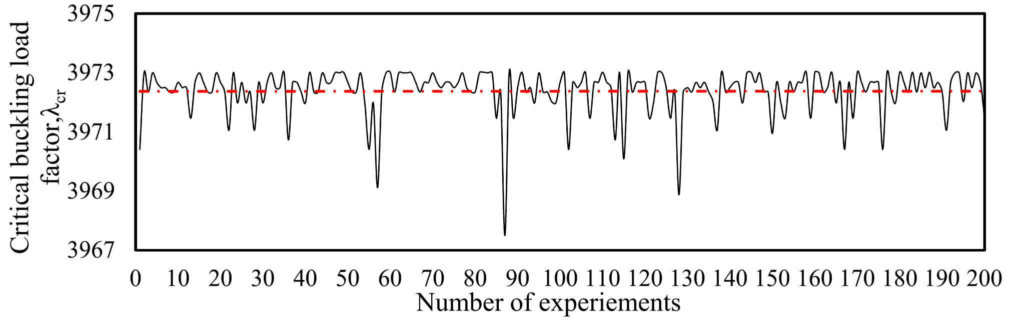
(a) EACO



(b) RBACO



(c) BWACO



(d) EHCFACO

Figure 5.5 Critical buckling load factor vs. number of experiments

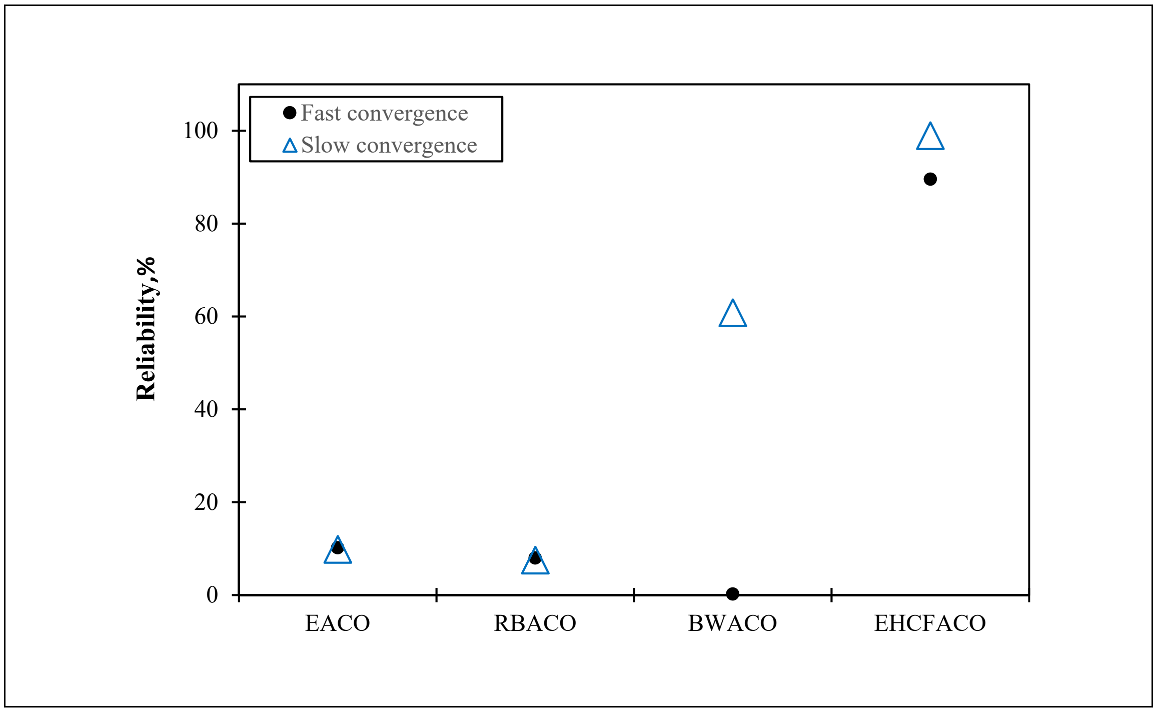


Figure 5.6 Reliability of EACO, RBACO, BWACO and EHCFACO algorithms

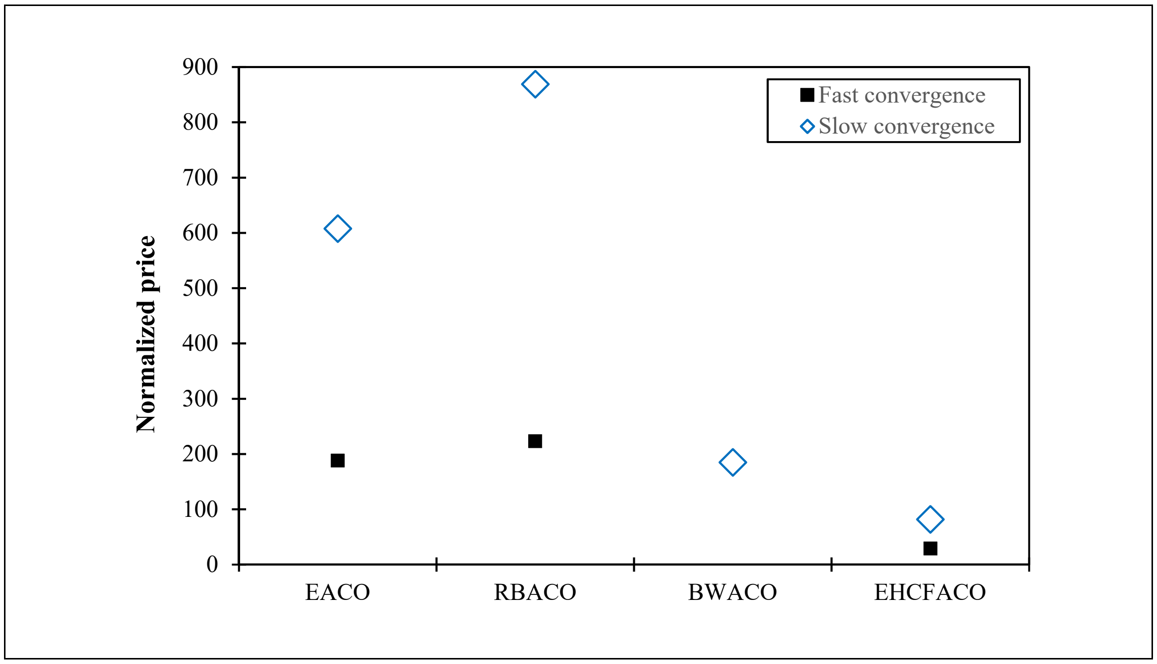


Figure 5.7 Normalized price of EACO, RBACO, BWACO and EHCFACO algorithms solutions

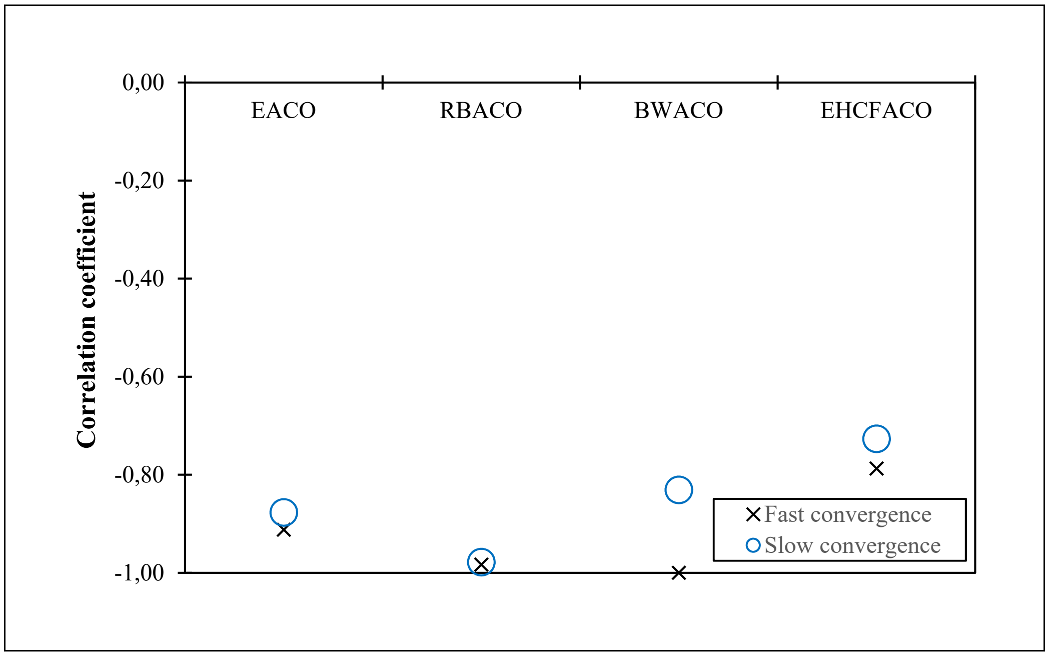


Figure 5.8 Correlation coefficient of EACO, RBACO, BWACO and EHCFACO algorithms

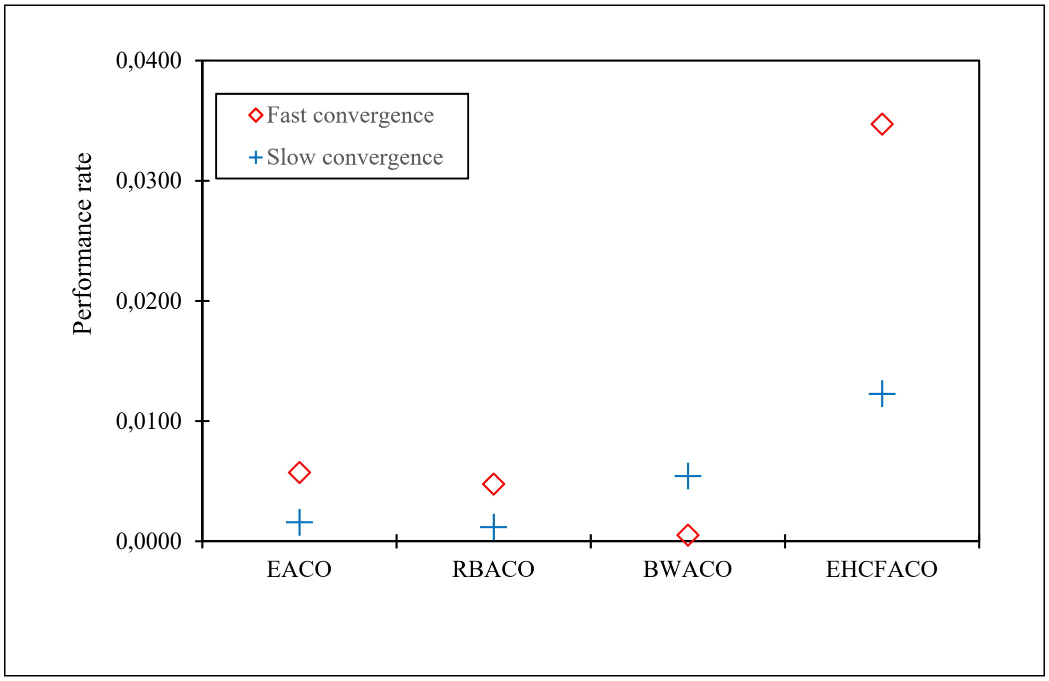


Figure 5.9 Performance rate of EACO, RBACO, BWACO and EHCFACO algorithms

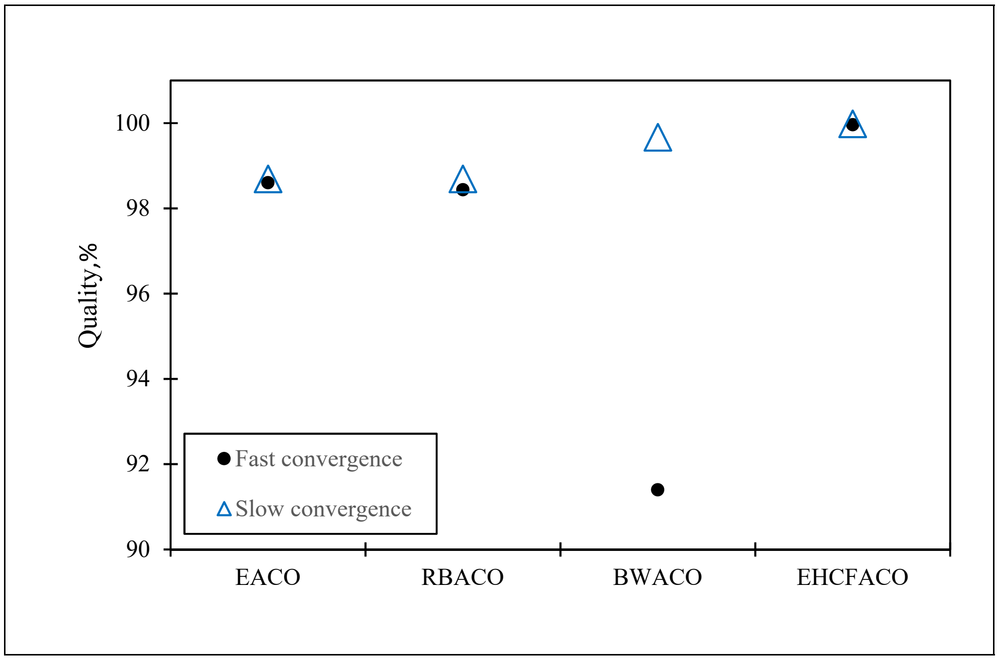


Figure 5.10 Solution quality of EACO, RBACO, BWACO and EHCFACO algorithms

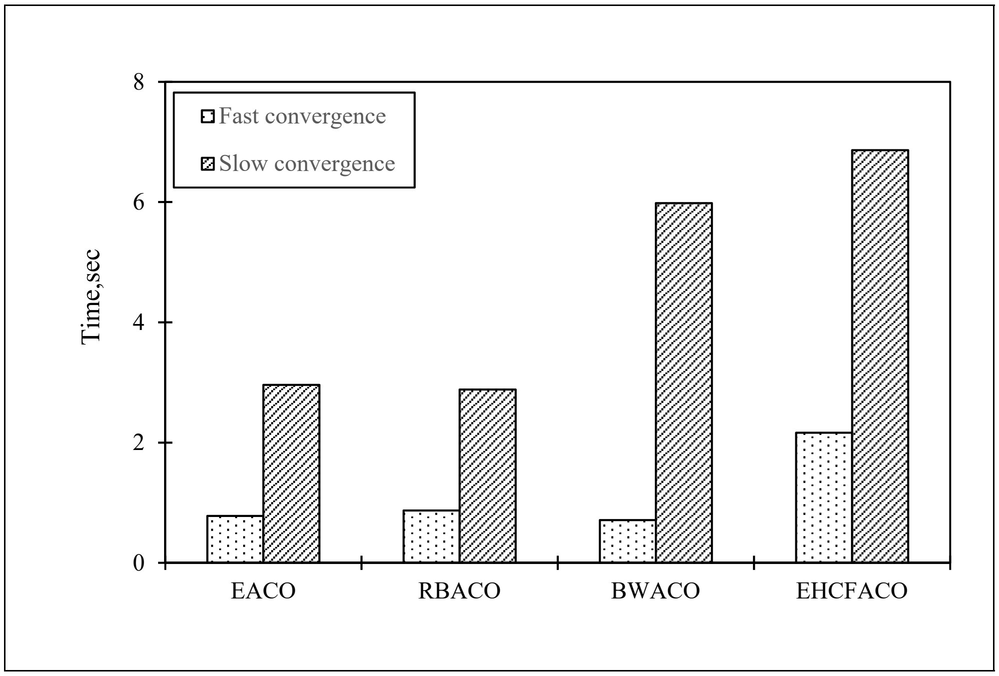


Figure 5.11 Elapsed time of EACO, RBACO, BWACO and EHCFACO algorithms to find the optimal solution

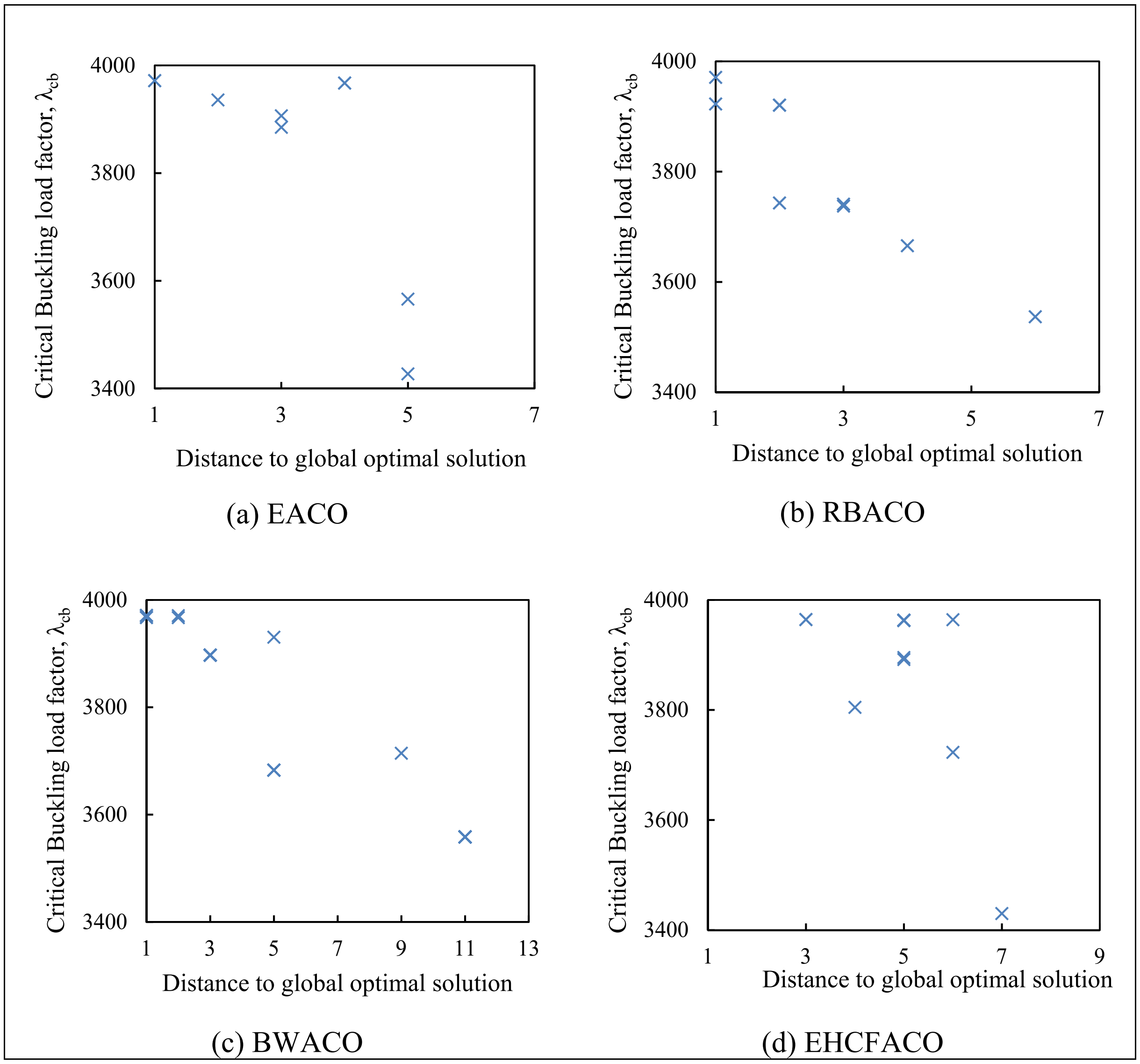


Figure 5.12 Fitness VS. Distance to global optimal solution of ACO Algorithms

5.8 Conclusion

Since many of the structural optimization problems are hard to solve within the polynomial computational time (NP), this study introduces a new optimization approach to solve the structural combinatorial optimization problems. The new approach uses an enhanced version of Hyper Cube Framework ACO (EHCFACO) that integrates two movements of insertion and bit flip to improve the local search feature of the original algorithm. A well-known benchmark case study of a composite laminated plate subjected to bi-directional buck ling loads has been selected to investigate the performance of the proposed algorithm. Furthermore, five different ACO variants were concisely presented and implemented to solve the same case study. General performance assessment measures, such as reliability, normalized price, . . . etc., have been determined for all presented ACO algorithms. It is observed that applying Hyper Cube Framework (HCF) to standard ACO has a significant influence on the overall performance of ACO. Furthermore, imposing local search movements, as an enhancement of exploitation effort, helped HCFACO to deliver a cost-effective solution. These improvements in ACO performance are in line with suggestions made by previous studies that rewarding HCF and local search movements the dominant factor in improving standard ACO algorithm performance (M. Dorigo et al., 2006; Marco Dorigo & Stützle, 2019). In general, the proposed EHCFACO outperforms the other ACO variants, where it offers a cost-effective solution.

CHAPTER 6

OPTIMIZATION OF PERFORATED COMPOSITE LAMINATED PLATE SUBJECTED TO UNCERTAIN GEOMETRICAL AND LOADING CONDITIONS

A.Ahmid [a] , T. M. Dao [b] and V. N. Lê [c]

[a,b,c] Department of Mechanical Engineering, École de Technologie Supérieure,
1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

Paper submitted to Elsevier Composite Structures, December 2020

6.1 Abstract

Nowadays, the real-life engineering applications are exposed to uncertainty influences of loading conditions or manufacturing errors. The traditional deterministic design optimization approaches are becoming more limited to offering cost-effective designs in similar circumstances. On the other hand, applying probabilistic design optimization techniques solved many deterministic approaches disadvantages. Unfortunately, the availability and accuracy of the probabilistic information of different design variables greatly affect the final design quality. The third option is the robust design (or uncertainty design) optimization, which demonstrated a significant performance in solving different practical engineering designs. One of the efficient methods of robust design optimization is the anti-optimization approach. The optimization process is carried out at two levels of optimization and anti-optimization to find robust optimal design configuration. However, in a lack of analytical solutions available for the objective function of a given design, the anti-optimization technique becomes a computationally expensive and less attractive option for the designer.

In the current study, a robust design optimization framework has been proposed to solve a novel optimization problem of a perforated composite laminated plate subjected to the uncertainty of buckling loading and location of the cut-out center point. A cutting-edge technique of Artificial Neural Network (ANN) is used here to develop a precise prediction of the objective function values which eliminate the negative impact of accompanied expensive function evaluation when the anti-optimization procedure is used. The obtained results revealed interesting findings and proved the worthiness of using the proposed framework to optimize such designs.

Keywords: Robust design, optimization, perforated composite laminated plate, buckling, Meta-heuristics, Cuckoo Search.

6.2 Introduction

The design of composite laminated structures is influenced by different uncertainties in loading conditions, geometric imperfections, material characteristics and any other designing or manufacturing parameters. The efficient design approach needs to consider the existence of such uncertainties to prohibit the different failure modes that may occur simultaneously, such as in the design of structures subjected to buckling loads (Lombardi & Haftka, 1998). Most of the prior researches have applied one or more of three different design optimization approaches of deterministic, probabilistic and robust design optimization. The deterministic design uses the factors of safety to deal with uncertainty influences. Still, it can result in an inefficient design that fails to spot one or more failure modes. Also, it becomes, even more worse when different failure modes are optimized against the design limits (Beck & Gomes, 2012).

The probabilistic approach has shown better results compared to the deterministic approach. Unfortunately, probabilistic design optimization is very sensitive to the accuracy and amount of statistics design data. The scarcity or inaccurate data leads to misfit the probabilistic distribution of uncertainty domains of the design variables (Lombardi & Haftka, 1998; Qiu & Wang, 2010). The robust design optimization became the preferred approach by engineers, where it eliminates the uncertainties influences via considering bounded uncertainty domains of the design variables (Isaac Elishakoff & Ohsaki, 2010; A Kaveh et al., 2019).

The anti-optimization approach is the common form of robust design optimization for structure applications, also known as a two-level optimization, and it was introduced originally by I Elishakoff et al. (1994). The anti-optimization levels create a nested optimization/anti-optimization loop where the top level is devoted to determining the optimal solution of a given design. In contrast, the bottom level is anti-optimize the obtained optimal solution to find the worst scenario case. The proposed approach was formulated to optimize ten-bar structure weight subjected to the uncertainty of loading, stress and displacements. The uncertainty of loading variations was limited to a multi-dimensional box uncertainty domain. Based on their work findings, the authors pointed out that the anti-optimization approach overcomes the numerical complexity that accompanied the probabilistic optimization.

Venter and Haftka (1996) introduced a two species Genetic Algorithm to reduce the computational effort of GA as an optimizer of two-level problems. They demonstrated the effectiveness of the improved algorithm by solving the anti-optimization problem of a composite laminate plate subjected to in-plane bi-directional compression loading in addition to the uncertain out-of-plane uniform load. The proposed algorithm exhibited a significant saving in computational effort. Adali et al. (2003) studied the maximization of the critical buckling load of a composite laminated plate subjected to uncertain loading conditions and lamina material type. Both deterministic and robust optimization approaches were examined. The authors concluded that deterministic critical buckling load factor values were less than these obtained by the robust optimization approach as a result of different stacking sequence design obtained by both approaches.

Liao and Chiou (2006) proposed a new method to minimize composite laminated structures weight subjected to sensitivity, stress concertation, thermal buckling load and uncertainty constraints. The proposed method considered uncertainties of design and non-design variables (e.g. ply orientation and material properties) to formulate a robust design problem. A Finite Elements Model (FEM) was used to determine the objective function at the optimization level. Simultaneously, an analytical solution of a simply supported composite laminated plate was used at the anti-optimize level. The proposed method showed a significant influence of the material properties' uncertainty on the optimal weight of the laminated plate. Following this evidence, the authors recommended using a big safety factor when the composite laminate structure was designed based on a deterministic approach. Han, Jiang, Gong, and Huang (2008) developed a method that uses the interval analysis with a hybrid numerical method to compute the transient response bounds of composite laminated plates to load and material properties uncertainties. The influence of different design variables uncertainty was investigated. The transient response bounds acquired by using first-order Taylor expansion together with interval extension. The results imply that the proposed method was confined to a small level of uncertainty applications. On the other side, the method could be extended to solve hybrid composite laminated structures. Qiu and Wang (2010) studied the solution of the anti-optimization problem that integrated a deterministic optimization technique with an interval analysis method to remedy the uncertainty influences of design variables. The interval set is generated based on the uncertain design variables. The results approved the superiority of robust design optimization compared to deterministic and probabilistic optimization.

Kalantari, Dong, and Davies (2017) conducted a robust multi-objective optimization study of a hybrid composite laminate. The minimization of the laminate weight and cost were the optimization objectives which confined to design variable and manufacturing uncertainty constraints. A modified GA algorithm was utilized to carry out an anti-optimization procedure. Their study found that the composite laminate weight and cost were increased when the uncertainty influences included. A Kaveh et al. (2019) suggested a robust optimization procedure that considers uncertain bi-directional buckling loading of a hybrid composite laminate plate. Two meta-heuristics of the GA and Quantum-inspired Evolutionary Algorithm (QEA) were integrated to handle the anti-optimization procedure. Two domains of uncertainty were examined for buckling loading for different aspect ratios. The obtained results were compared to those reported by Adali et al. (2003), and they demonstrated a good agreement.

Despite this considerable body of literature on the anti-optimization approach as a robust design technique for composite structures, such as beams or solid plates, far too little attention has been paid to the robust optimization of the perforated composite laminated plate. This could be explained by the computational cost associated with the objective function evaluation of such structures, where the analytical solution is not available. Furthermore, using GA as an optimizer of the anti-optimization problem dominates the previous studies reviewed here, even though GA is known as an expensive optimization algorithm.

However, the central thesis of the current study is finding how the anti-optimization approach could be applied to solve optimization problems that have an expensive objective function at a reasonable computational cost. Accordingly, a novel robust optimization framework has been introduced. To demonstrate the effectiveness of the proposed framework, a case study of a perforated composite laminated plate subjected to the uncertainty of buckling loading and location of the cut-out center point has been examined. The plate edges were simply supported, and the objective of the optimization is finding the optimal Stacking Sequence Design (SSD) that maximize the critical buckling load factor. An Artificial Neural Network (ANN) technique was used to eliminate the expensive cost that accompanied the buckling analysis of the perforated laminated plate. The ANN is trained based on data samples generated by the Latin Hypercube (LHC) plan. The design responses to these inputs were determined by the commercial FEA software of ANSYS workbench. Then, the trained ANN used to replace the expensive objective function in the anti-optimization procedure.

Eventually, it could be said that the proposed framework offered a cost-effective solution for the examined NP-hard optimization problem of the perforated laminated plate.

6.3 Problem Formulation

The design optimization of the perforated composite laminated plate for maximum critical buckling load has been studied using deterministic approaches. Hu and Lin (1995) conducted a deterministic optimization study of a rectangular plate with a circular cut-out located at the plate center. The optimization objective was maximizing the critical buckling load factor of the symmetrically laminated plate. In contrast, plate thickness, aspect ratio, fiber orientation and end conditions were the design variables of the optimization problem. One of their interesting concluded findings is the buckling loading capacity of the perforated laminated plate could be surpassing the similar solid plate, without cut-out, by tailoring the fiber orientation and cut-out diameter. Other deterministic studies carried out by D. Kumar and Singh (2012) investigated different design variables' effects on stability and failure strength of composite laminate subjected to unidirectional compression and in-plane shear loading conditions. Different cut-out shapes have been examined for a set of three fiber orientation of $[\pm 45°]_s, [0°_2\ \ 90°_2\,]_s$ and $[0°_2 \pm 45°\ 90°_2]_s$ configurations. They observed that a perforated laminated plate with a circular cut-out shape exhibits better strength utilization than a laminated plate with other shapes.

A similar case study will be discussed in the next sections, see Figure 6.1; but this time, it is optimized against the uncertainty of loads, diameter ratio and location of cut-out center points.

6.3.1 Anti-Optimization Problem Formulation

As mentioned in the introduction, the anti-optimization procedure consists two levels of optimization, at the top of the loop, and anti- optimization at bottom level of the loop. The objective of the optimization level is maximizing the critical buckling load, λ_{cr}, while the purpose of anti-optimization level is finding the worst-case scenario due uncertainty. Figure 6.1 illustrates a perforated composite laminated plate exposed to in-plane buckling loading conditions where N_x and N_y are uncertain loads belongs to uncertainty domain Up which is confined by:

$$U_p = \{(N_x, N_y)|\, N_x \geq 0, N_y \geq 0, N_x^p + N_y^p = 1\} \tag{6.1}$$

p represents the uncertainty domain exponent that gives the domain shapes of triangular, circular and rectangular for $p = 1,2,\infty$ respectively, see Figure 6.2. The values of $N_x, N_y\ \epsilon\ U_p$ need to be determined by an anti-optimization procedure such that critical buckling load factor, λ_{cr}, of the perforated laminated plate is minimized for all possible design variables configuration.

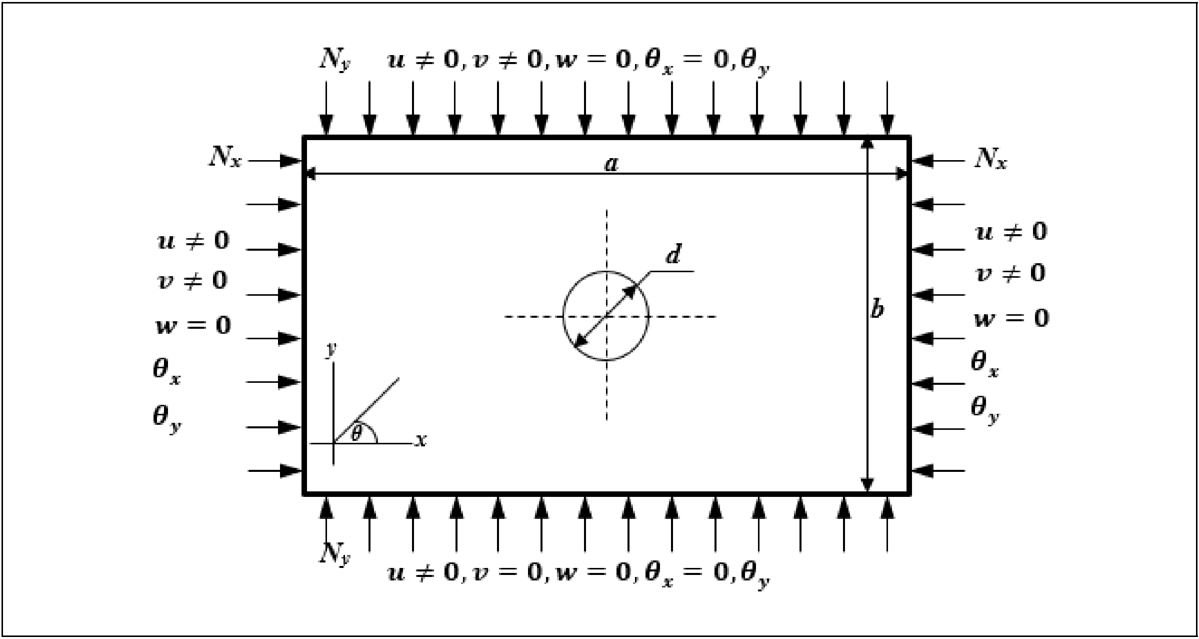

N_y $u \neq 0, v \neq 0, w = 0, \theta_x = 0, \theta_y$
N_x
a
$u \neq 0$
$v \neq 0$
$w = 0$
θ_x
θ_y
d
b
y
θ
x
N_y $u \neq 0, v = 0, w = 0, \theta_x = 0, \theta_y$

Figure 6.1 Loading and boundary conditions of the simply supported perforated laminated plate.

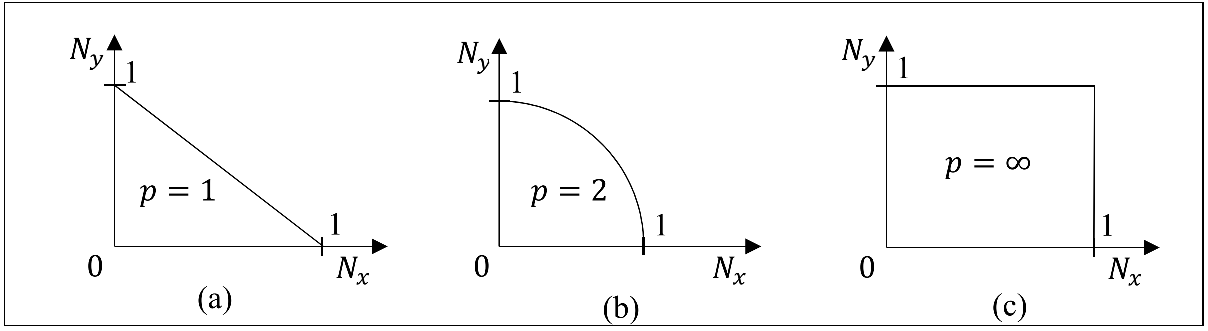


Figure 6.2 Different uncertainty loading domains.

The set of design variables consists of the fiber orientation angle θ_i, circular cut-out diameter (or diameter ratio) d/b and the cut-out center positions P_i. The other laminated plate design variables are imposed, such as width a, height b and ply thickness t_p. Thereby, the critical buckling load factor could be written as:

$$\lambda_{cr}(m,n) = \min \lambda(\ \theta_i, d/b\,, c_i, N, m, n) \tag{6.2}$$

Where m, n are buckling modes in x and y directions. The anti-optimization problem could be formulated to find uncertainty values of $N = (N_x, N_y) \epsilon\ U_p$ as follow:

$$\lambda_{cr}(\theta_i, d/b\,, c_i, N^*, m, n) = \underset{N \epsilon U_p}{min}\ \lambda(\,P_i, d/b\,, \theta_i, N, m, n) \qquad (6.3)$$

The Eq. (6.3) results in the worst-case scenario, which needs to be optimized to find the optimal design variables values that yield the maximum critical buckling load factor.

6.3.2 Optimization Problem Formulation

The current study focuses on optimizing the SSD that maximizes the buckling load of the perforated laminated plate subjected to uncertainty loading and dimensions conditions.

a.) Optimal solution representation

The manufactured laminate layers have limited sets of fiber orientations such as $0^{\circ}, \pm 45^{\circ}$, and 90°. Consequently, the design domain turns into a discrete one, and the optimization problem becomes NP-hard combinatorial problem. This type of optimization problem is known for its complexity and thus, selecting the suitable optimization method plays a major part in finding an inexpensive optimal solution with good accuracy. The Meta-Heuristics (MH) exhibited a significant performance in solving different SSD optimization problems of composite laminated structures. To some extent, the MH solutions for SSD optimization problems take the form of a bit string that consists of a combination of plies with the available orientations set. The simplicity of using an integer representation and its significant performance makes it the most widely used method in MHs for SSD optimization of composite laminated structures (R Le Riche & Haftka, 1995).

Figure 6.3 shows the interpretation of integer representation of laminate with $[3\ 3\ 3\ 2\ 1\ 1]_s$ stacking sequence where numbers 1,2 and 3 representing 0°, $\pm 45^{\circ}$, and 90° fiber orientations, respectively.

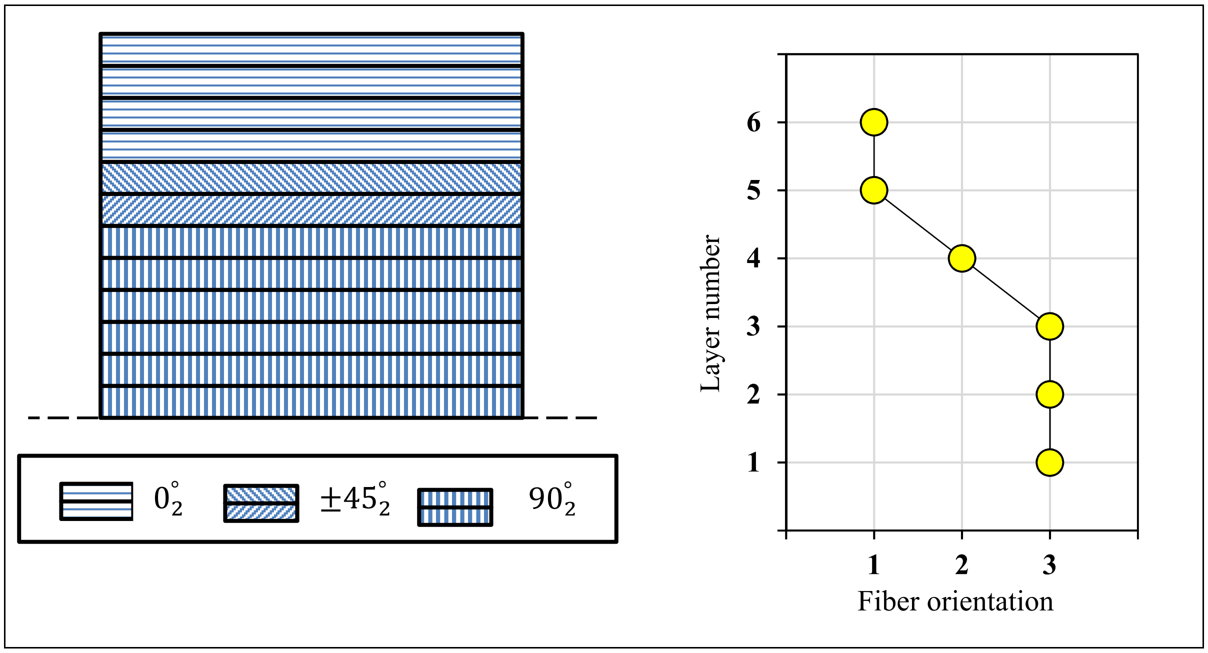


Figure 6.3 Solution representation of symmetrical and balanced laminate.

b.) The objective function

The objective function of maximizing critical buckling load factor for a laminate subjected to uncertainty conditions of buckling loading and location of cut-out center could be written as follow:

$$f_{obj} = \max\{\min[\lambda(P_i, d/b\,, \theta_i, N, m, n)]\} \tag{6.4}$$

where λ_{cb} is the critical buckling load factor that buckles a simply supported plate subjected to in-plane loads of λN_x and λN_y into m and n half-waves in x, y directions. The smallest value of $\lambda\ (P_i, d/b\,, \theta_i, N, \boldsymbol{m}, \boldsymbol{n})$ is considered the critical buckling load factor. The critical values of m and n are linked to different factors such as laminate material, a number of plies, loading conditions, and the plate aspect ratio. In uniaxial loading of a simply supported plate, the critical buckling load occurs when $m = 1$ whereas in biaxial critical buckling loads, it needs to be determined as the minimum value of $\lambda\,(P_i, d/b\,, \theta_i, N, \boldsymbol{m}, \boldsymbol{n})$, (Söyleyici, 2011).

c.) The constraints

The design of the composite should respect certain limitations of manufacturing and certain design considerations. In the literature, some rules have been proposed to improve the effectiveness of a laminate design for different applications (Javidrad, Nazari, & Javidrad, 2017; Kim, Kim, & Han, 2005). Manufacturing limitations could be the thickness of ply or fiber orientations that are limited to available manufacturing values. For example, the available fiber orientations are usually limited to values of $\pm 45°, 0°$, and 90° angles, whereas the ply thicknesses are limited to specific step values. Additionally, symmetrical and balanced laminate makes the manufacturing process simpler. Another notable design constraint is the congestion of the same orientation plies, which cause the undesirable effect of laminate crack propagation.

Generally, these constraints of SSD optimization problem, with constant laminate thickness, could be handled as follows:

- The symmetry constraint is enforced by optimizing half of the laminate.
- Balancing constraint is enforced by selecting θ_2 for the standard fiber orientation set of $0°, \pm 45°$, and 90°.
- Only $N/4$ ply orientations are needed to describe laminate because of balancing constraints.
- The contiguity constraint is handled using the penalty parameter (ρ), (R. Le Riche & Haftka, 1993).

The final objective function form, that includes the contiguity penalty parameter (ρ), could be written as:

$$f_{obj} = (1 - \rho).\,max\{min[\lambda_b(P_i, d/b\,, \theta_i, N, m, n)]\} \qquad (6.5)$$

6.4 The Proposed Uncertainty Optimization Framework

Figure 6.4 illustrates the proposed procedure to solve the uncertainty optimization problem of the perforated composite laminated plate. The procedure starts with the initialization step where the laminate information is given (e.g. material properties, geometry dimensions, ...etc.). The procedure starts with the pre-investigation of a certain set of design variables to determine the initial worst-scenario case. Then the obtained geometrical settings are transferred to the ANN block to generate inputs data sample that will be used to train the ANN. The generated ANN prediction function is used as the objective function of the optimization/anti-optimization of the perforated composite laminated plate. The loop of optimal/anti-optimal solutions will continue until the termination criteria are met.

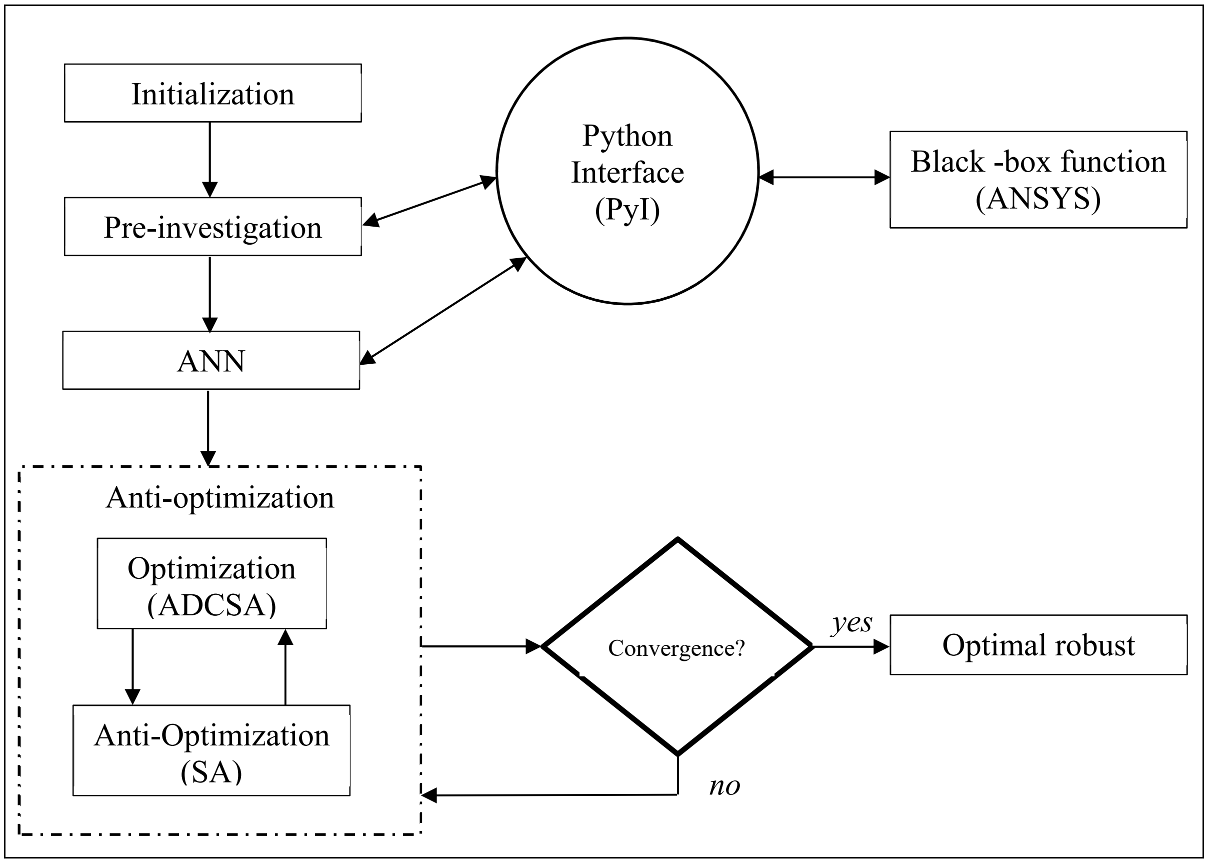


Figure 6.4 Proposed robust optimization procedure.

6.4.1 Black-Box Function

The black-box function term is used to describe the expensive objective functions that consume long computational time. Mostly, these functions values determined using black-box software (e.g. Ansys WB) or even the observations of experimental work.

Ansys Workbench is a well-known product design software, and it covers different types of engineering design problems. One of the recent ANSYS Workbench features is Ansys composite PrePost (ACP), which deals with designing those products built-up using layered composites. ACP gives the designer the ability to define the complex parameters of the composite structure interactively such as the layers number, fiber orientations, ply thickness and materials of each layer group.

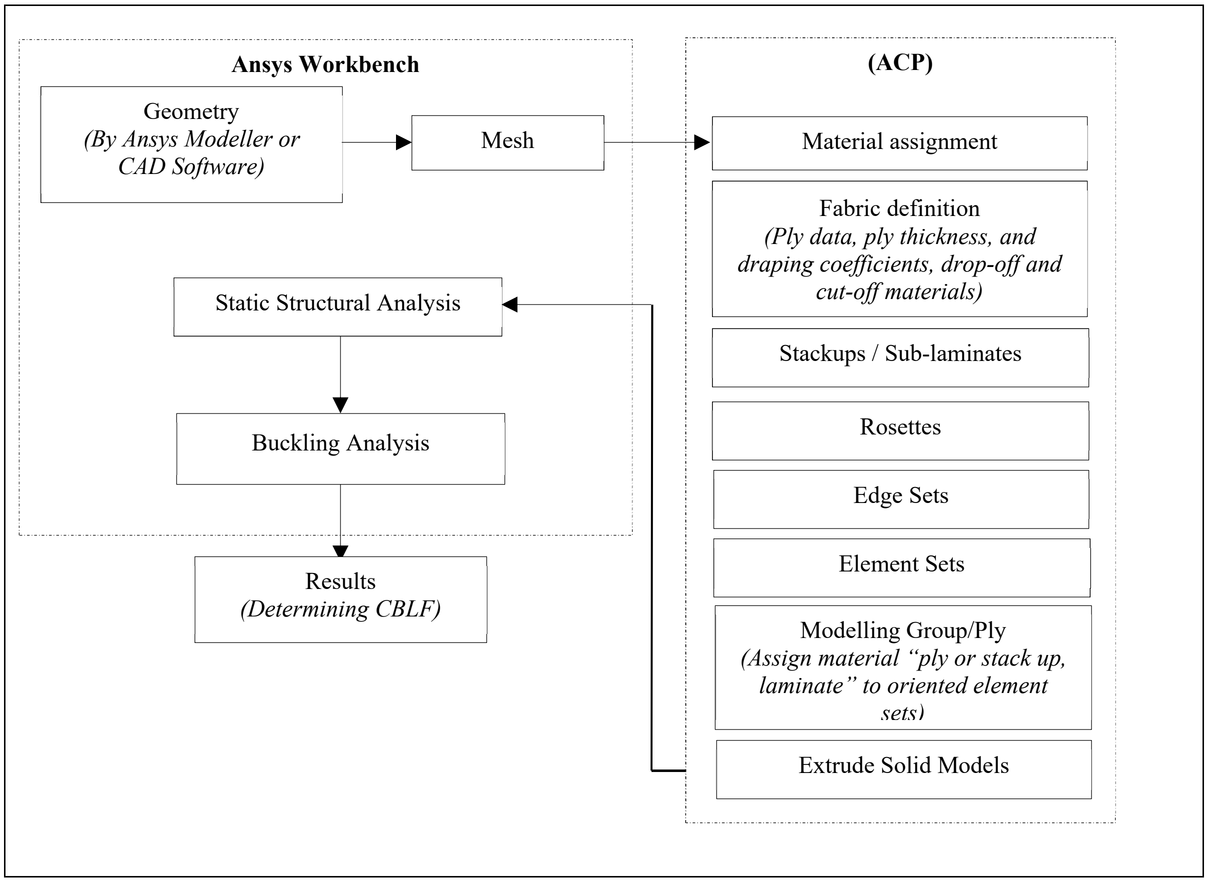

Ansys Workbench
Geometry
(By Ansys Modeller or CAD Software)
Mesh
Static Structural Analysis
Buckling Analysis
Results
(Determining CBLF)
(ACP)
Material assignment
Fabric definition
(Ply data, ply thickness, and draping coefficients, drop-off and cut-off materials)
Stackups / Sub-laminates
Rosettes
Edge Sets
Element Sets
Modelling Group/Ply
(Assign material "ply or stack up, laminate" to oriented element sets)
Extrude Solid Models

Figure 6.5 Ansys Composite Pre-post (ACP) model analysis flowchart
Taken from (Ahmid2019)

The definition of the laminated geometry is the basic step of the product design, and this geometry is used next to create the model mesh and to define the boundary conditions as well as the loading, see Figure 6.5. ACP has two processing modes, pre, and post, where all composite geometry definitions and meshing model could be created in the pre-processing mode, whereas the post-processing is evaluating the design analysis results. In the current study, Pre-post is used alone where there is no need to do a post-processing step, and the critical buckling load is determined through the buckling analysis.

6.4.2 Python Interface (PyI)

ANSYS Workbench affords the capability of recording all Graphical User Interface (GUI) actions. This feature is called journaling, and it creates a scripting file written in Python programming language. Running the journal file from the ANSYS Workbench menu or the command prompt will repeat all actions of the model creation. The user can modify and reuse this file in the way that he needs, (Ansys, 2015). This feature of journal scripting is employed here to create a set of x, y data set, where x represents the random inputs of the design vector generated using the LHC sample plan while y are their equivalent responses determined by ANSYS Workbench. The developed Python file is used to transfer the determined critical buckling load factor (or y) back into the Matlab code to fill the x, y data matrix that is used to train the ANN.

6.4.3 Artificial Neural Network (ANN)

ANN is a deep learning algorithm that simulates the neuron structure in the human brain, and it is originally presented by Hinton (1986). Even though the concept of deep learning and ANN presented a few decades ago, but it did not find the appropriate attention until recently. The main barriers confronted deep learning was the availability of data and the computational power which are needed to build and train the learning model. These barriers are demolished over time, and deep learning algorithms gain momentum due to the abundance of the data and computational power.

The neuron is the main block of an ANN algorithm, which creates a linear combination activated by a nonlinear function, see Figure 6.6. The neuron works as a node where input signals are loaded, and certain computations proceeded. The input signals, x_i, received from raw data or from another neuron in the prior layer. The weight (w_i) and bias (b) are the design parameters of the ANN, where weights are applied to the input signals followed by summation and then adding the bias.

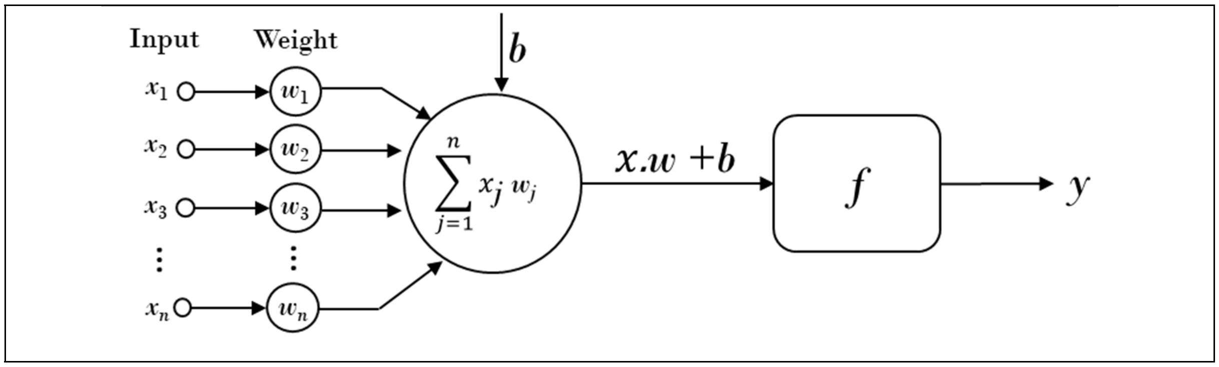


Figure 6.6 Typical structure of an artificial neuron.

Eventually, the obtained linear combinations are activated using a nonlinear function such as a sigmoid function to determine the output value (y_i). The deep learning model of an ANN is formed by connecting a specific number of these neurons to respect the net scheme. In the current study, the feedforward neural network scheme is used through the *fitnet* function in the Matlab program. This kind has three main types of layers; the first layer called the input layer, which contains the input data set, the second is the hidden layers that have a number of connected neurons, and the last one is the output layer that predicts the desired output, see Figure 6.7.

Based on the given data set, the net starts to find the connection between the input and output data that form a nonlinear relation. It occurs through a process called learning of the network, and it is carried out by using different training algorithms. Once the net learned how to find the output value of each input, it could be used to predict the output value of any other inputs.

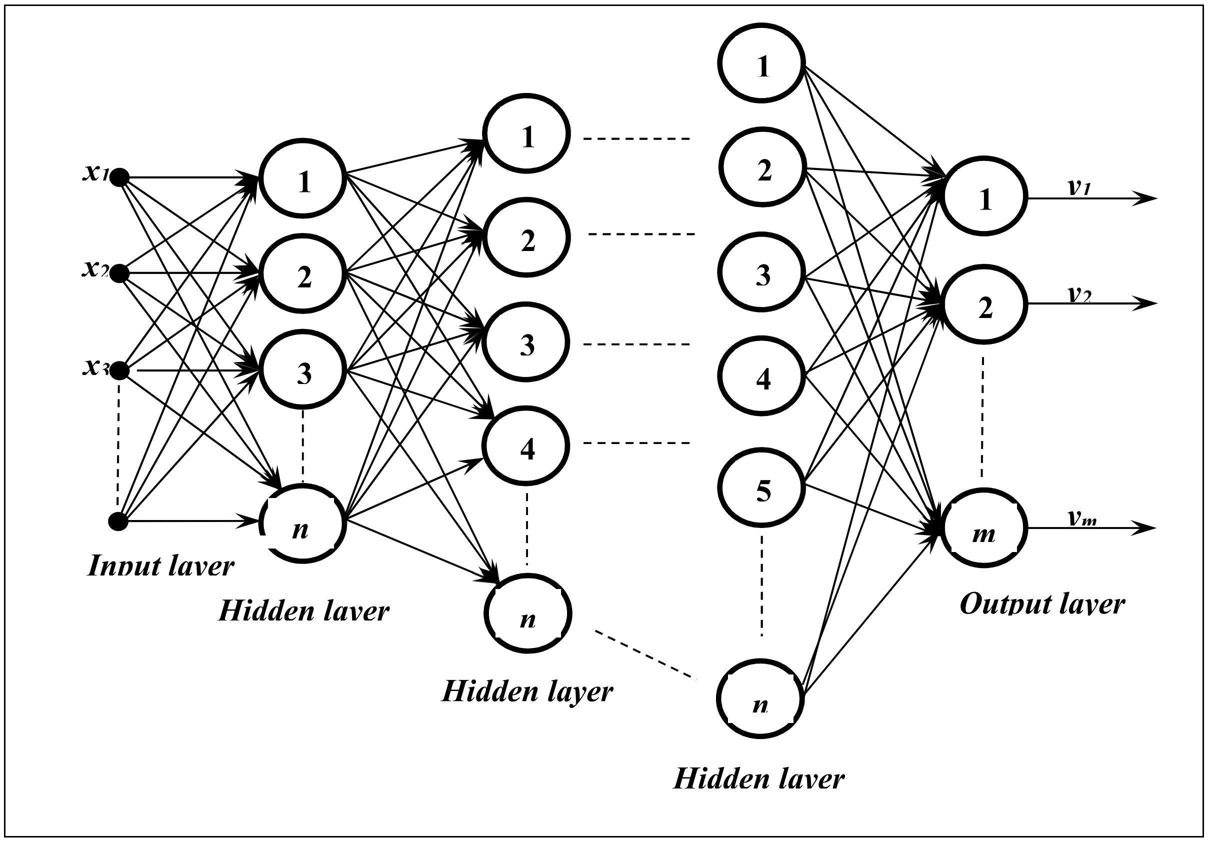


Figure 6.7 Structure of the feedforward neural network.

Figure 6.8 shows the different steps of building ANN used here, and each step is briefly explained in the following subsections.

a.) Data Sampling

The first step in developing a neural network model is collecting the data set of the designated problem. The data set consists of inputs, which need to be randomly generated, and outputs that need to be determined by the black-box function. Thus, generating the random sample of the inputs has great impacts on the quality of the data set created. There are a variety of sampling methods that could be used; here, a Latin Hypercube (LHC) design plan has been employed. LHC is a statistical scheme to generate a random sample of variables values that fulfill a certain criterion. LHC divides the range of each variable into a number of equal intervals. To respect LHC requirements, the number of interval needs to be the same for all variables. LHC takes random samples one at a time, knowing which samples have been taken so far. Thereby, LHC ensures sampling each variable range fairly with the least possible number of samples (Iman, Helton, & Campbell, 1981).

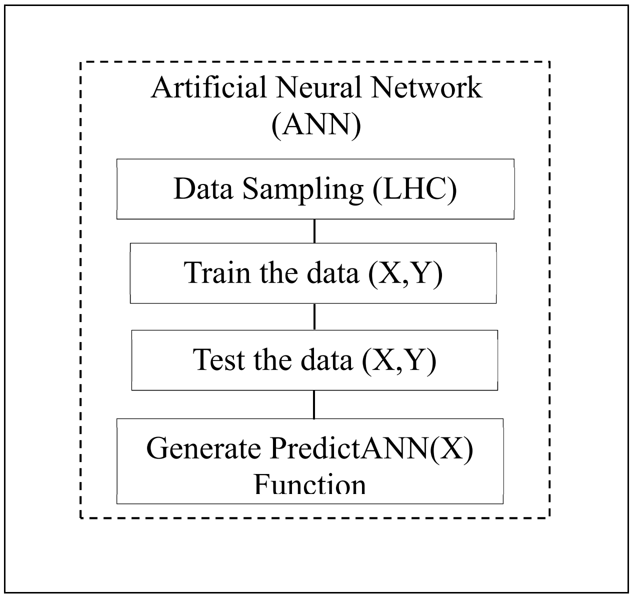

Artificial Neural Network (ANN)
Data Sampling (LHC)
Train the data (X,Y)
Test the data (X,Y)
Generate PredictANN(X) Function

Figure 6.8 ANN building flowchart.

However, to apply the LHC plan for the current design problem, there is a need to understand that the input vector,X, have a mixed type of design variables of real and integer numbers. The design vector consists of the center of the cut-out P_i, the diameter ratio d/b , stacking sequence configuration (or SSD) θ_i, and loading conditions N_i.

$$X = \left[P_i(x_i, y_i) \quad d_i/b \quad \theta_i \quad N_i(N_x^i, N_y^i)\,\right] \tag{6.6}$$

θ_i is a vector of integers between 1 and 3; they represent the available fiber orientations, and its length depends on the number of the laminate plies. As soon as the input samples were created, they sent to the black-box function program, ANSYS, to find the corresponding output values described in section 6.4.2.

b.) ANN Training and Testing

When the data set becomes available, it will be split into two groups of training data and testing data. The training data dedicated to training the ANN to learn the relationship between the input and output values; thus, it could predict the output of any other input values. The learning process is carried out using different algorithms that consist of a certain set of blocks designed to adjust the neuron's weights and bias. The training data usually hold about 60-90% of the whole data set while the remaining data used to test the solution generalization of the network (I. N. Da Silva, Spatti, Flauzino, Liboni, & dos Reis Alves, 2017). In the current study, 85 percent of the data set used to train the network while 15 percent used for the network solution testing. The training algorithm used here is called Bayesian Regularization backpropagation, which is known for its well-generalized networks.

6.4.4 Anti-optimization Procedure

As mentioned in the introduction, the anti-optimization procedure has two levels of optimization and anti-optimization, which work to find the robust optimal SDD that considers the uncertainty influences of buckling loading and location of the cut-out center. Two different MHs were selected to conduct this process, and they were introduced concisely in the following sections.

a.) Adapted Discrete Cuckoo Search Algorithm (ADCSA)

The original Cuckoo Search algorithm (CS) is designed to solve unconstrained continuous optimization problems (X.-S. Yang & Deb, 2009). CS demonstrated a significant performance in solving several NP-hard optimization problems such as TSP, scheduling and truss weight optimization problems. Since CS presented in 2009 by Yang, various improvements and modifications were presented. Some of the developed CS variances tried to deal with the discreteness nature of real-life applications. Unfortunately, most of those discrete CS variants were designed to solve an appointed optimization problem. For instance, the Binary CS (BCS) is anticipated by Gherboudj et al. (2012) to solve the knapsack problem. However, other variants were introduced as more general-purpose variants such as Discrete CS (DCS) by A Kaveh and Bakhshpoori (2013), which uses the approach of rounding to the nearest discrete value for the newly generated solutions. More recently, Loubna et al. (2017) suggested another discrete CS variant based on a rank-value approach where the new solutions determined based on the integer-order (rank) of each possible discrete value in the design domain. This variant applied to the discrete problem of image recognition, and the obtained results reveal the efficiency of the rank-value approach.

Algorithm 6.1 Adapted Discrete Cuckoo Search Algorithm (ADCSA)
Taken from (A. Ahmid, Thien-My, & Le, 2020)

```
Initialization:
        - Initial and evaluate a random population of $n$ host nests ($xi, fi$) using Latin
          HyperCube (LHC) random generator.
While (convergence not met) Do:
        - Generate a new Cuckoo (population) randomly by Lévy flights.
        - Evaluate the new Cuckoo fitness ($f_i$).
        - Choose a nest among $n$ ($say, j$) randomly.
    If $fi > fj$
        - Replace the j with the new solution.
    end
        - Rank the solutions and find the current best.
        - Discard Pa fraction of worst solutions.
        - Substitute the discarded solutions by new ones generated by Lévy flights.
    If (maximum successful runs number exceeded)
        - Do permutation, swap, insertion and bit flip for the current best solution.
    end
        - Update the best solution.
End
```

Here, a most up-to-date variant of CS called Adaptive Discrete CS Algorithm (ADCSA), by (A. Ahmid et al., 2020), was used to carry out the optimization level of the anti-optimization procedure. ADCSA bears three main modifications to the original CS. First, it is using Latin Hypercube (LHC) sampling method to generate the initial population; the second is presenting discrete Levy flight representation and finally improve the neighbourhood search of the best solution through four different permutation movements. The proposed ADCSA pseudo-code is listed in Algorithm 6.1.

b.) Simulated Annealing (SA)

SA is a trajectory-based MH that mathematically analogy the thermal annealing process of metals. The concept of SA introduced by Metropolis in the early 1950s' and since then, it has examined to solve a wide range of optimization problems (Kirkpatrick, Gelatt, & Vecchi, 1983). SA proved a significant performance in solving numerous optimization problems, and here is used to perform the anti-optimization level job of finding the worst-case scenario. The typical structure of the SA optimization algorithm is shown in Algorithm 6.2.

Algorithm 6.2 Simulated Annealing Algorithm procedure
Taken from (A; Ahmid et al., 2019)

```
Initialization:
        - Initialize SA parameters $(T, c, n)$
        - Generate an initial random solution.
        - Evaluate the initial solution.
While (termination criteria not satisfied) Do
        - Generate a new solution from the current solution vicinity.
        - Calculate the current solution of energy.
        - Calculate the new solution energy.
        - Compare both solutions energy.
        - Update the current solution with the biggest.
        - If the number of iterations > $n$
        - Reduce the temperature, $T$, by reduction factor $c$.
End SA algorithm for combinatorial optimization problems
```

a.) Convergence criterion

The anti-optimization block continues to find the optimal and anti-optimal solutions until the difference between both solutions,Δf, remains constant for a certain number of iterations or the maximum number of anti-optimization iterations is reached.

$$\Delta f = f_{opt} - f_{anti} \tag{6.7}$$

6.5 Numerical Experiment

The optimization framework developed here was examined for a case study of perforated composite laminated subjected to biaxial buckling loading, see Figure 6.9. The laminate has an aspect ratio $a/b = 1:2$, where $a = 558\ mm$ and $b = 279\ mm$; the number of plies $N_p = 24$, ply thickness $t_p = 0.1163\ mm$, the edge distance $e = 3.h_p$ and the diameter of the cut-out $d = 0.2,0.3,\ldots,0.8.b\ (mm)$. The edges are simply supported, and the biaxial loading $N(N_x, N_y) \in [0,1](N/m)$. The laminate material is Epoxy-Graphite, $E_{11} = 132.58\ GPa$, $E_{12} = 10.8 GPa$, $G_{12} = 5.7 GPa, \nu_{12} = 0.24$ and $\nu_{23} = 0.49$.

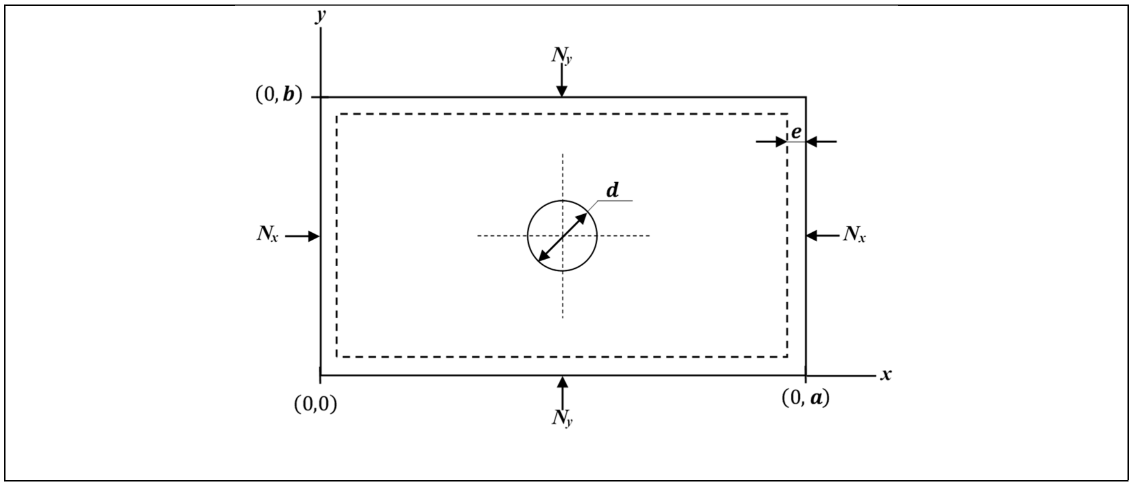


Figure 6.9 Main dimensions and loading conditions of 24 ply perforated Graphite-Epoxy laminate.

The different framework blocks were programmed using Matlab R2019b program, while ANSYS Workbench 2020R2 was used to determine the Black-box function values. The numerical experiment started by creating an FEA model using ANSYS, then a mesh convergence check is conducted. This step is followed by pre-investigation for specific loading conditions with a set of different cut-out diameters and locations to identify the spot of the possible critical cut-out configuration, which will be used later as the initial worst case. Finally, based on previous step outcomes, the ANN is created, and the anti-optimization procedure is activated to determine the optimal robust SSD.

6.5.1 Finite Element Modelling

The laminate buckling analysis goes through three steps; it starts by generating the laminate model geometry and defining the material properties in ANSYS ACP. Next, the static structural analysis takes place and finally carrying out the eigenvalue buckling analysis. The geometry of the laminate FE model was created, and the material properties were defined according to the given laminate information. Creating the model in ACP starts with sketching the principle geometry of the laminated plate, which is a simple 2D rectangular, and then a surface body is created using this sketch. Then, it is followed by creating the surface body that is needed to implement the meshing model. The SELL181 element type was used, and the number of elements initialized by just 90 elements and gradually increased up to 384 elements where a satisfied convergency rate was met, see Figure 6.10.

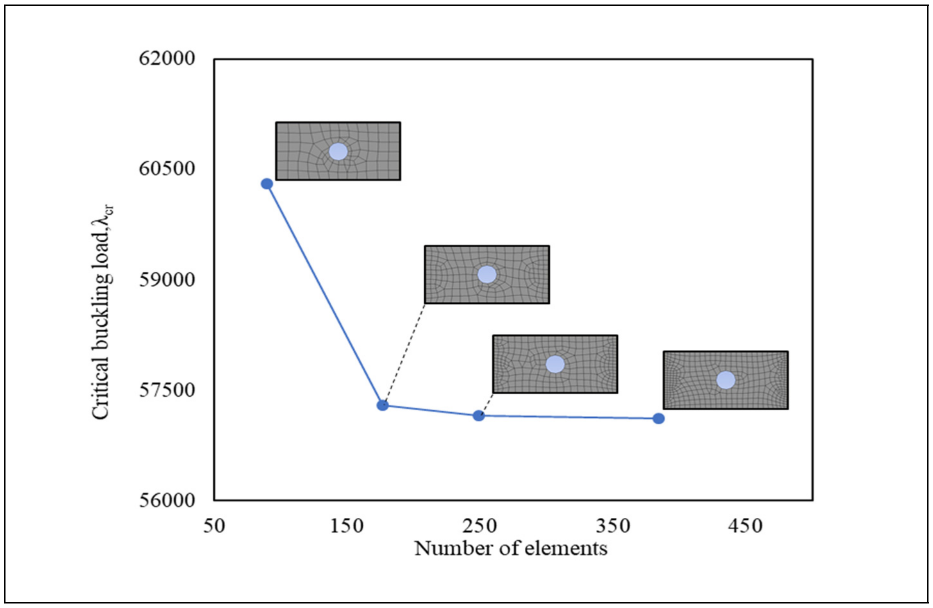


Figure 6.10 The mesh size convergency.

6.5.2 Preliminary Investigation

Three different laminates with SSD of angle ply, cross-ply and quasi-isotropic were examined to develop a draft idea about where is the initial worst-case scenario could occur. Nine different cut-out centers were selected while the cut-out diameter varies from $0.2b$ to $0.80b$ with an increment of $0.1b$, see Figure 6.11. Each laminate configuration is subjected to three different loading conditions. First is maximum unidirectional loading in x-direction; second is unidirectional loading in y-direction and, finally, bidirectional loading.

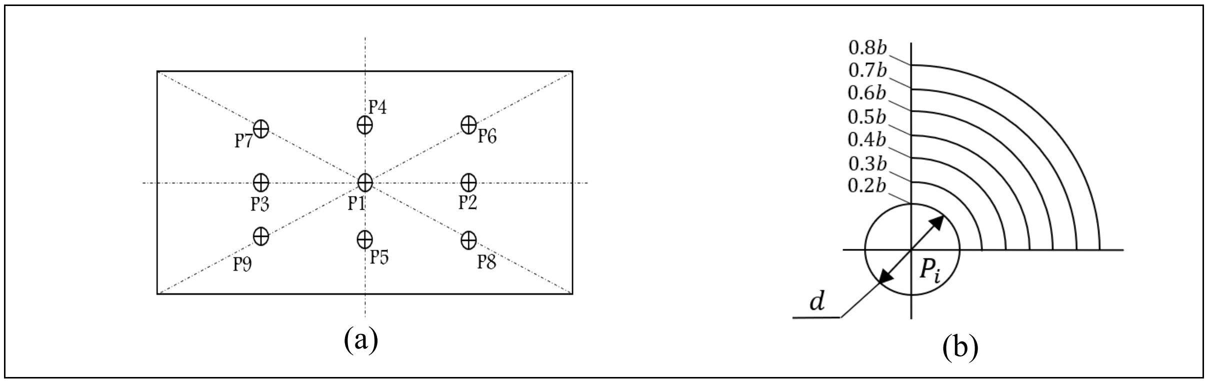


Figure 6.11 Selected cut-out centers and diameter aspect ratio.

These figures are quite revealing in several ways. First, the different examined stacking configurations followed almost the same pattern for the three loading conditions; for instance, in the unidirectional loading case, in the x-direction, the high d/b gives bigger values of λ_{cr} and this pattern is the same for all designated stacking configurations. The second interesting observation was that the most minimum λ_{cr} found for the case of unidirectional loading in y-direction rather than to be in case of bidirectional loading.

Overall, these results indicated that the worst-case scenario occurred at P_2 for $d/b = 0.8$, consequently, the ANN will be developed to this specific configuration, and the obtained prediction function will be used next as the objective function in the anti-optimization procedure.

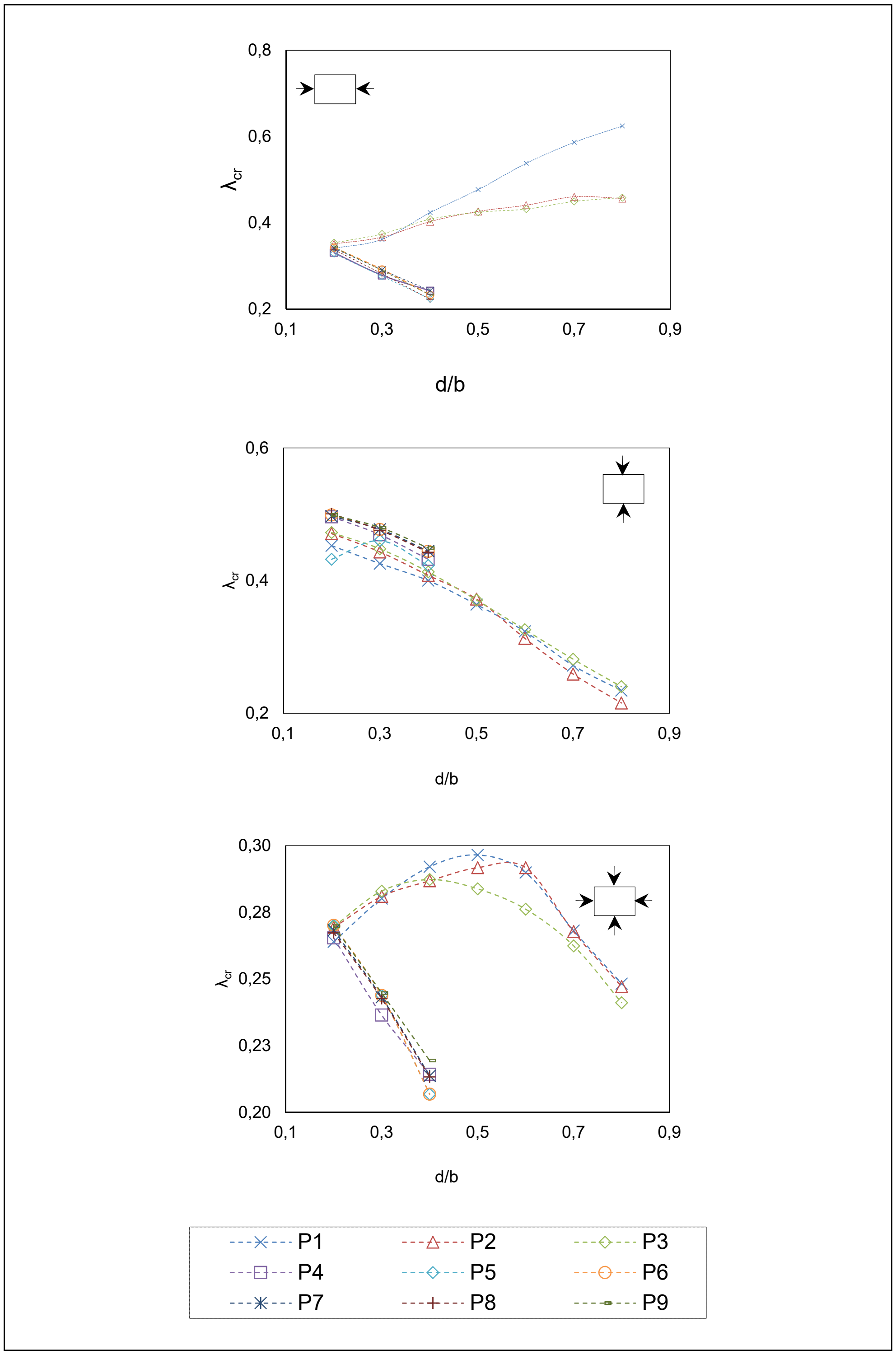


Figure 6.12 Critical buckling load factor vs. Diameter ratio for $[\pm 45°]_{6s}$ laminate under different loading conditions

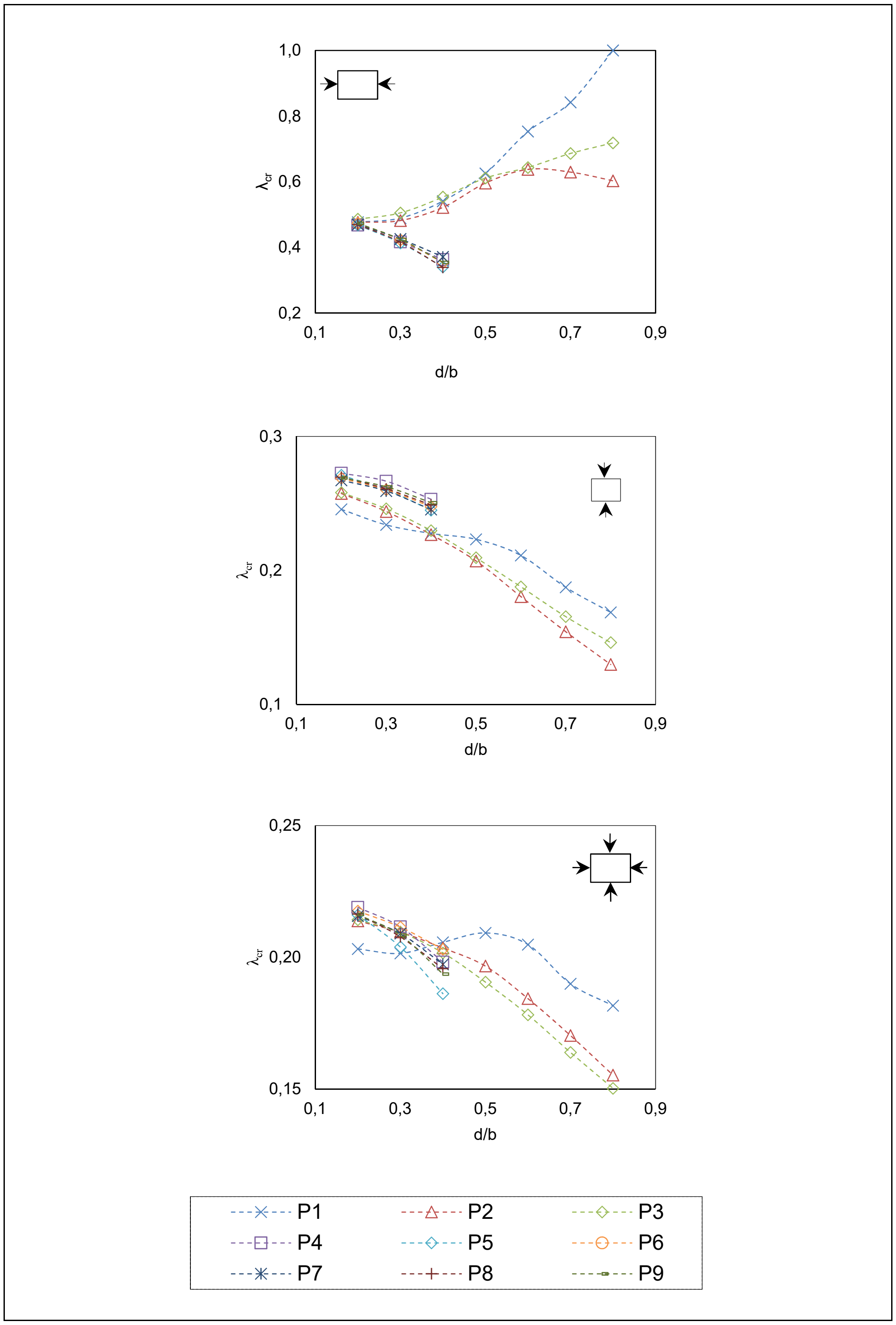


Figure 6.13 Critical bucking load factor vs. Diameter ratio for $[0°_2\ 90°_2]_{3s}$ laminate under different loading conditions

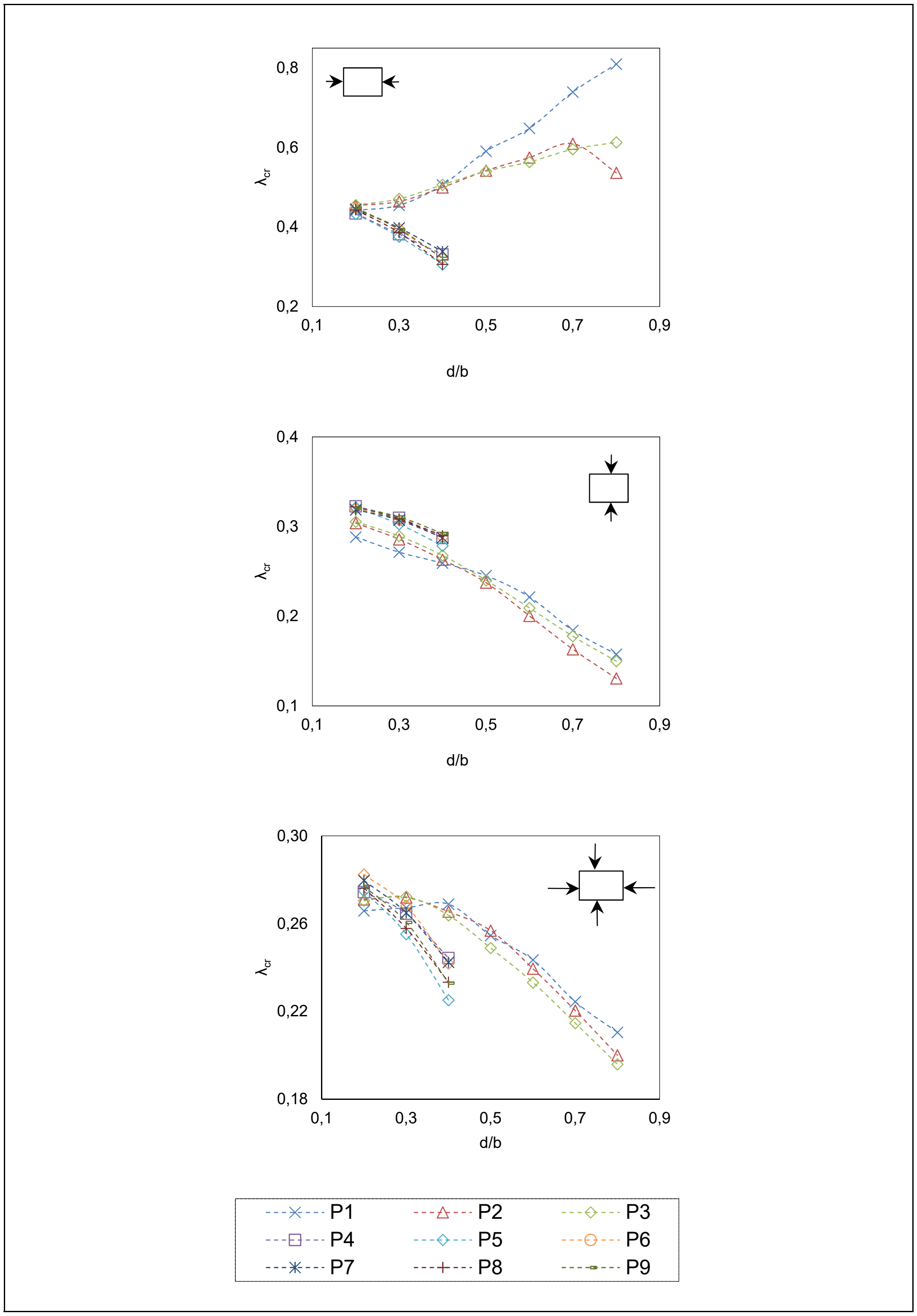


Figure 6.14 Critical bucking load factor vs. Diameter ratio for $[\pm 45^{\circ}\ 0^{\circ}{}_2 90^{\circ}{}_2]_{2s}$ laminate under different loading conditions.

6.5.3 ANN Settings

As stated in section 6.4.3 (a), the design vector has mixed types of numbers that need to be fairly handled to obeys the LHC plan criteria. Thus, all design vector variables are sampled simultaneously using the *lhsdesign* function in Matlab. Then, the sampling values of the fiber orientations (SSD) vector are interpreted to their equivalent integer values using the classification concept that is widely used in Machine Learning (ML) (Montáns, Chinesta, Gómez-Bombarelli, & Kutz, 2019). The sample size used here is 1000 observations. The equivalent responses,Y, of the generated sample determined according to the described procedure in section 6.4.3.

Regarding the ANN structure, the preliminary trials imply that ANN with three hidden layers and ten neurons each will perform perfectly for this specific problem. The performance goal of the ANN is set to 1e-06 and epochs to 3000. The developed prediction function is then verified using a new set of design vectors, see Figure 6.16.

Eventually, the anti-optimization loop is activated, and the tour to find the robust optimal SSD continues until 50 consecutive equal differences, $\Delta f,$ of optimal and anti-optimal solutions are obtained or maximum number of iterations, $I_{max} = 1000$ is reached.

6.6 Results

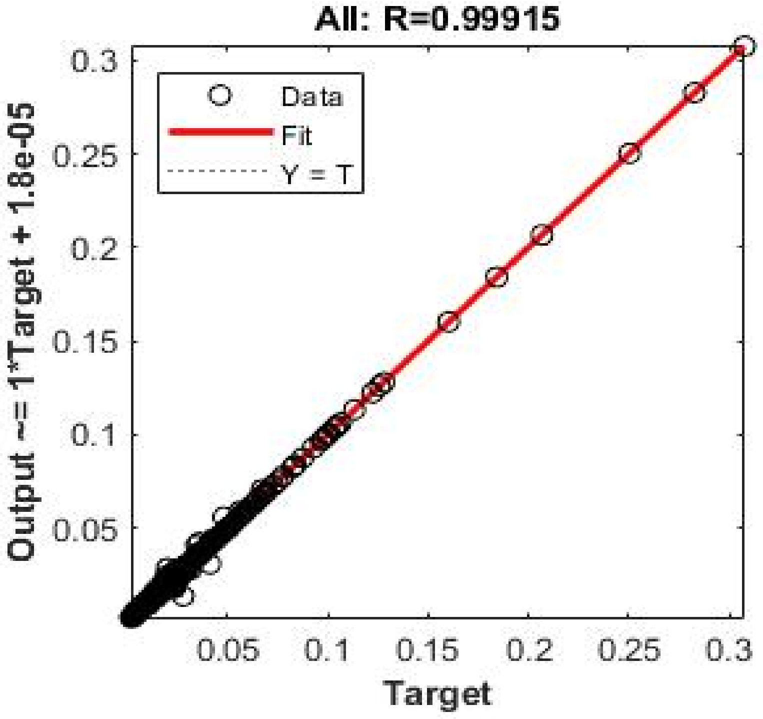
The trained ANN performance and their Mean Square Error (MSE) are plotted in Figure 6.15 (a) and (b). The validation of the ANN conducted using a new sample of design vectors which already evaluated using ANSYS software. The ANN prediction function and ANSYS results were compared and plotted in Figure 6.15. The comparison reveals a reasonable agreement between both results were the average error determined was around $\pm 4\%$.

The anti-optimization procedure started with an evaluation of the solution at the initial worst scenario (P_2 in the preliminary investigation) and trying to find the new optimal solution using

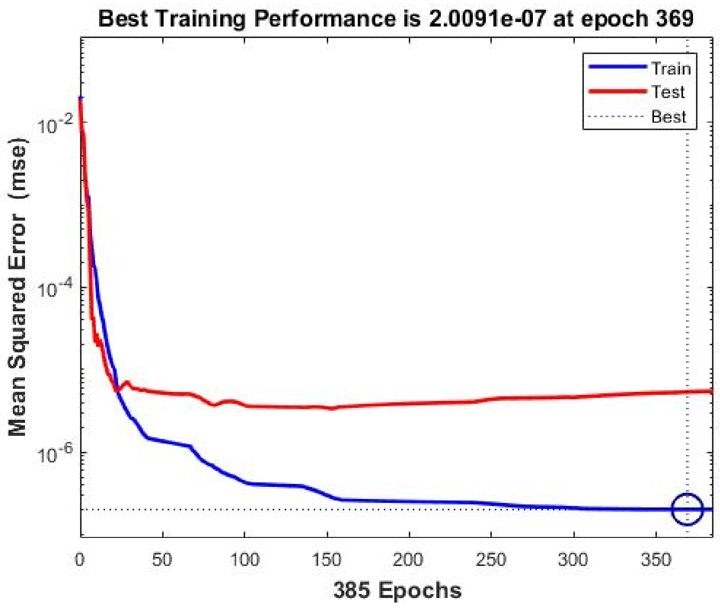
ADCSA and store it into the optimal solution matrix. The optimization level is searching the design space to find the optimal SSD that maximize the critical buckling load factor,λ_{cr}, for cut-out center and loading conditions of P_2. Then, the anti-optimization level starts a new search to determine the new location of the cut-out and new loading conditions that will minimize λ_{cr} obtained by optimal SSD. This loop continues until the mentioned convergency criteria are met. In the current experiment, the maximum number of iterations is exceeded, and the gained results of the anti-optimization procedure were plotted into Figure 6.17.

The experiment results show that the optimal SSD has 17 stacking sequence configurations, see Figure 6-18. Also, it could be noticed that the 90_2° fiber orientation is dominating all optimal SSD found here. This tendency to include more stacking sequence with fiber orientation 90_2° can be explained because of the existence of high loading in the y-direction. Moreover, the SSD 2 in Figure 6.18 repeatedly appeared during the anti-optimization procedure, see Figure 6.19

In addition to SSD 2, two other optimal solutions occupied 16% and 17% of the appearance during the anti-optimization loop, see Figure 6.19. Consequently, those three appointed SSD solutions were investigated more by plotting their cut-out center points and plotting their anti-optimization procedure results, see Figure 6.20 to Figure 6.25.Those figures have shown that most critical locations of the cut-out were near to preliminary investigation worst-case $P2$. Furthermore, the critical loading conditions occurred when the loading conditions in the y-direction are high, which is in line with preliminary investigation results.

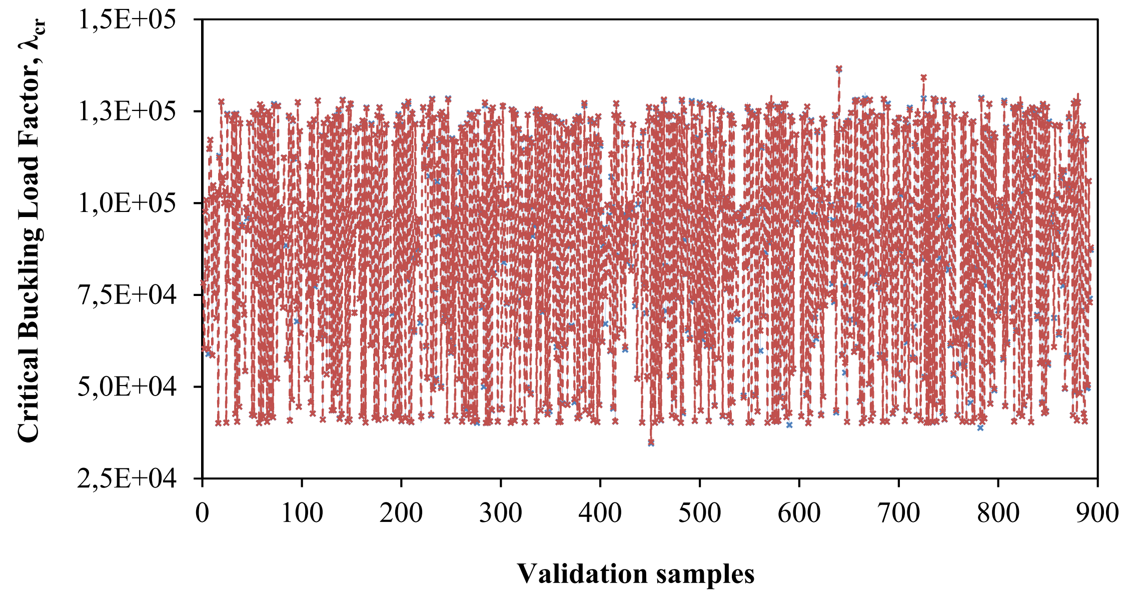


(a)



(b)

Figure 6.15 Data set regression and Mean Square Error (MSE) results using forwardfeed ANN



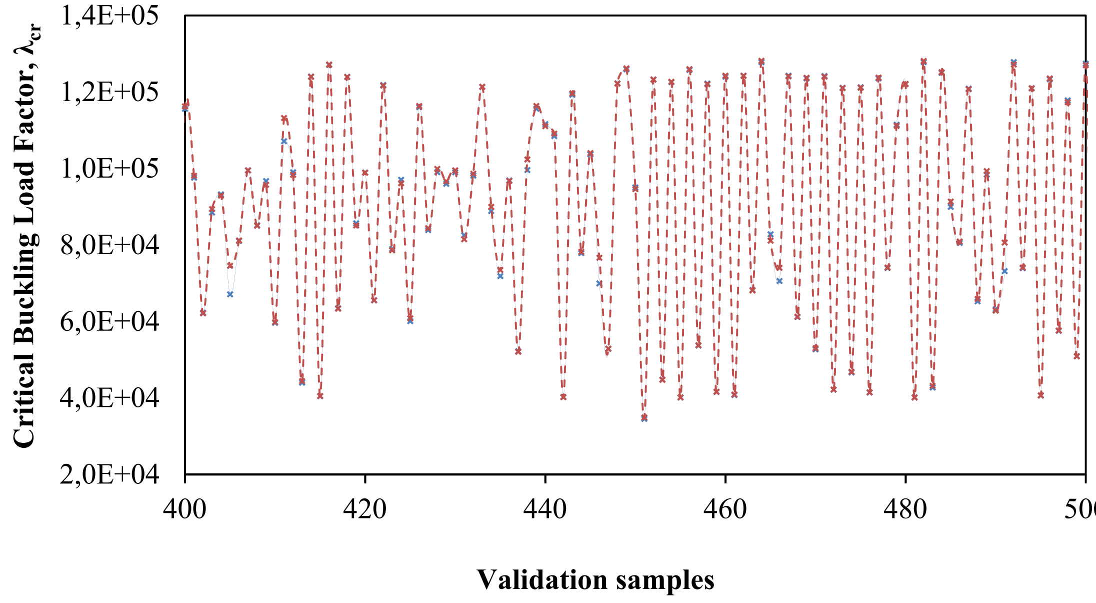


Figure 6.16 Comparison of ANN and Black-box function (ANSYS) results for the validation sample

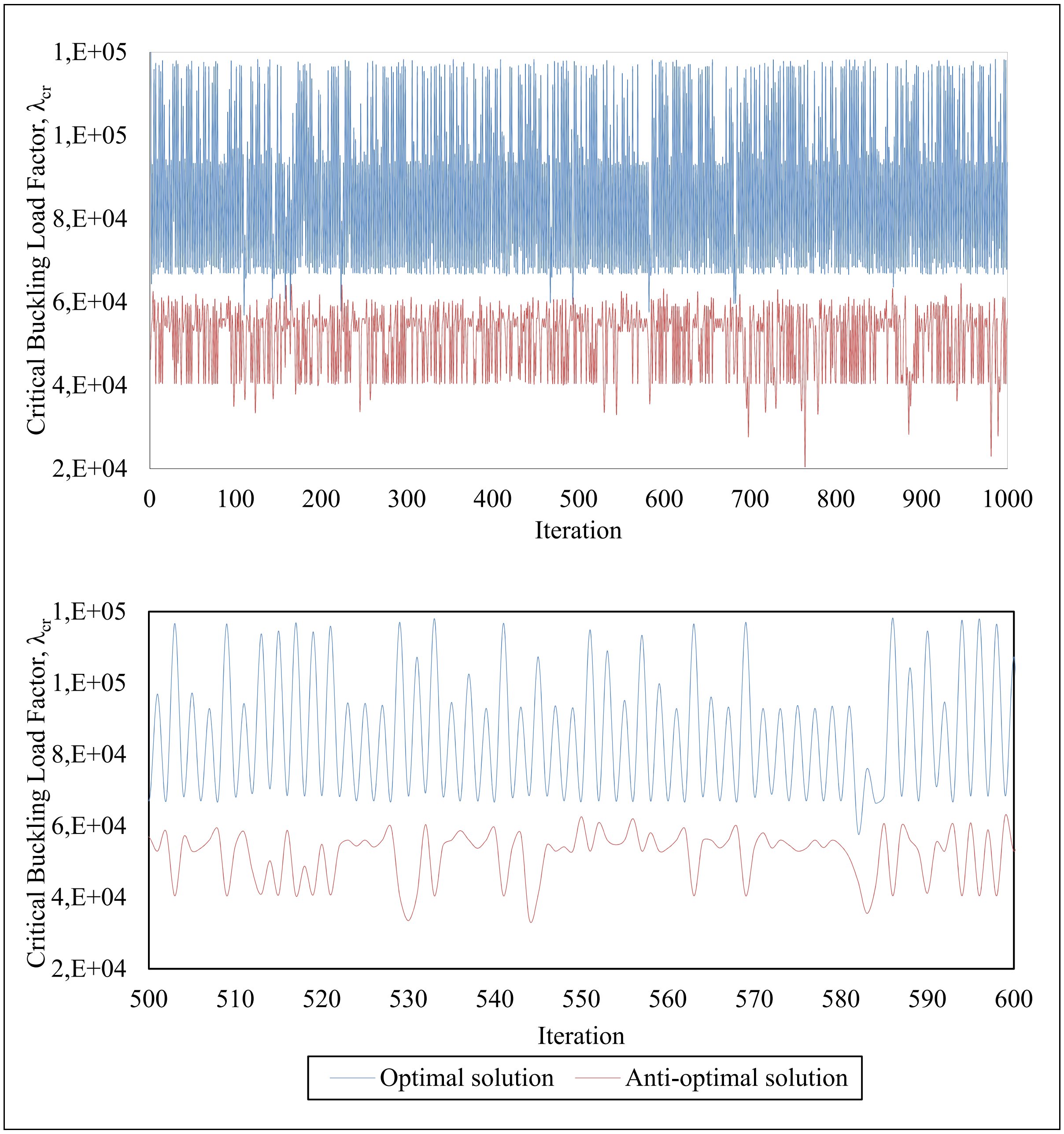


Figure 6.17 The anti-optimization procedure iteration vs. critical buckling load factor

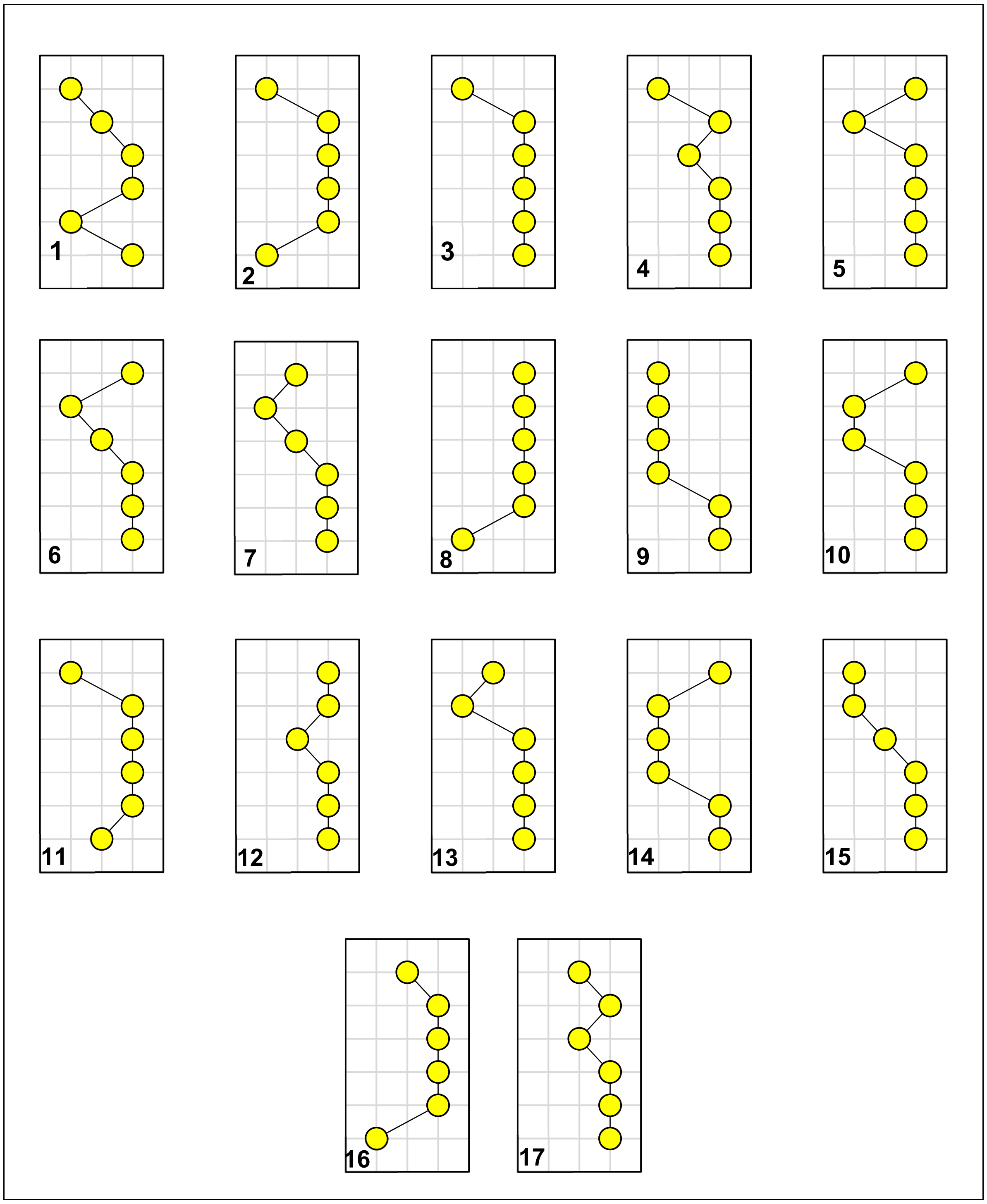


Figure 6.18 The SSD arrangements of the main optimal solutions.

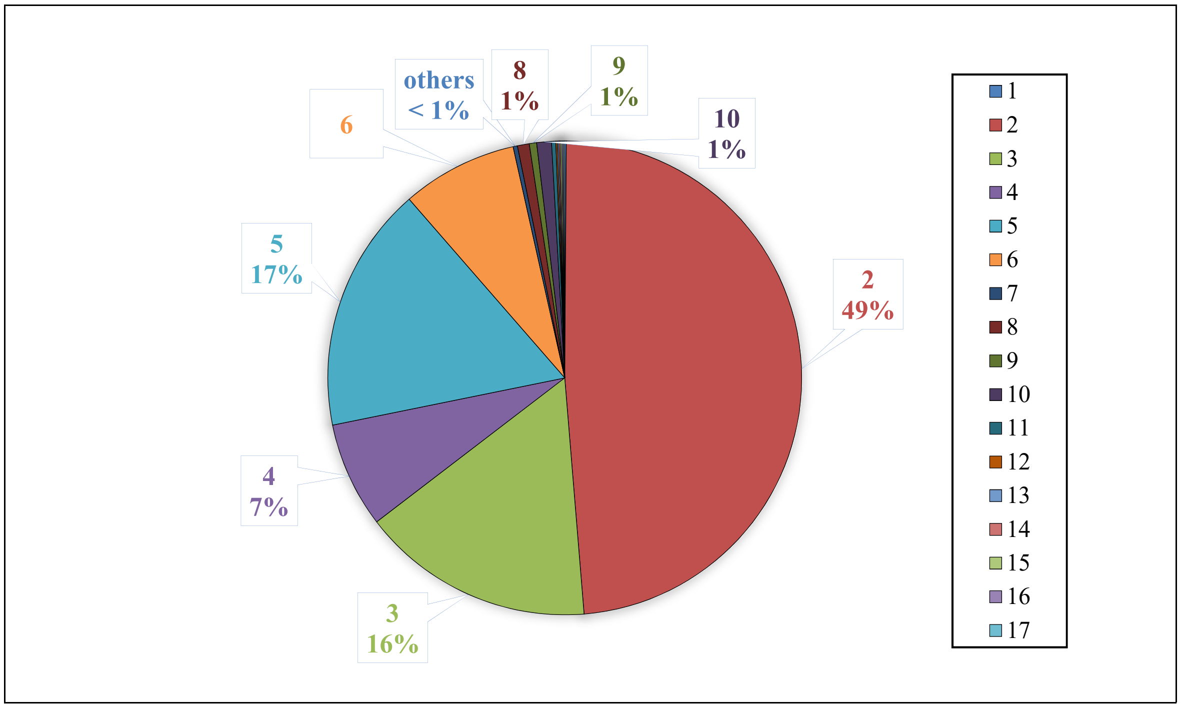


Figure 6.19 Main optimal SSD percentage breakdown.

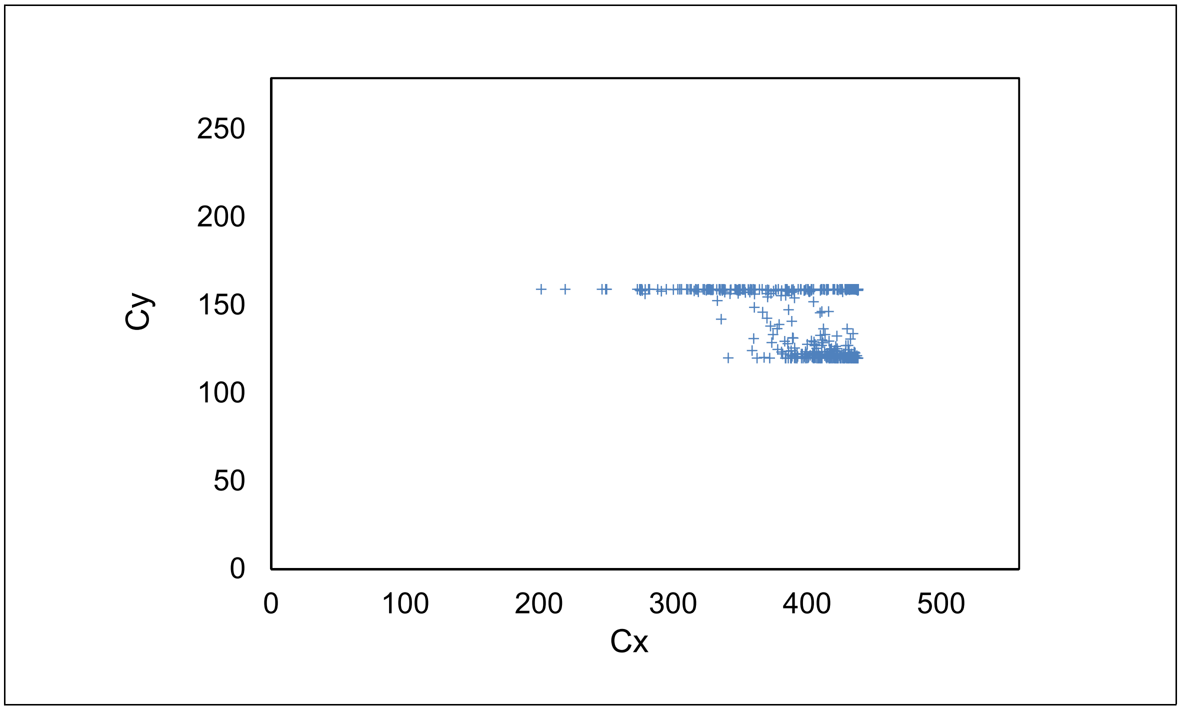


Figure 6.20 Cut-out center points of SSD2

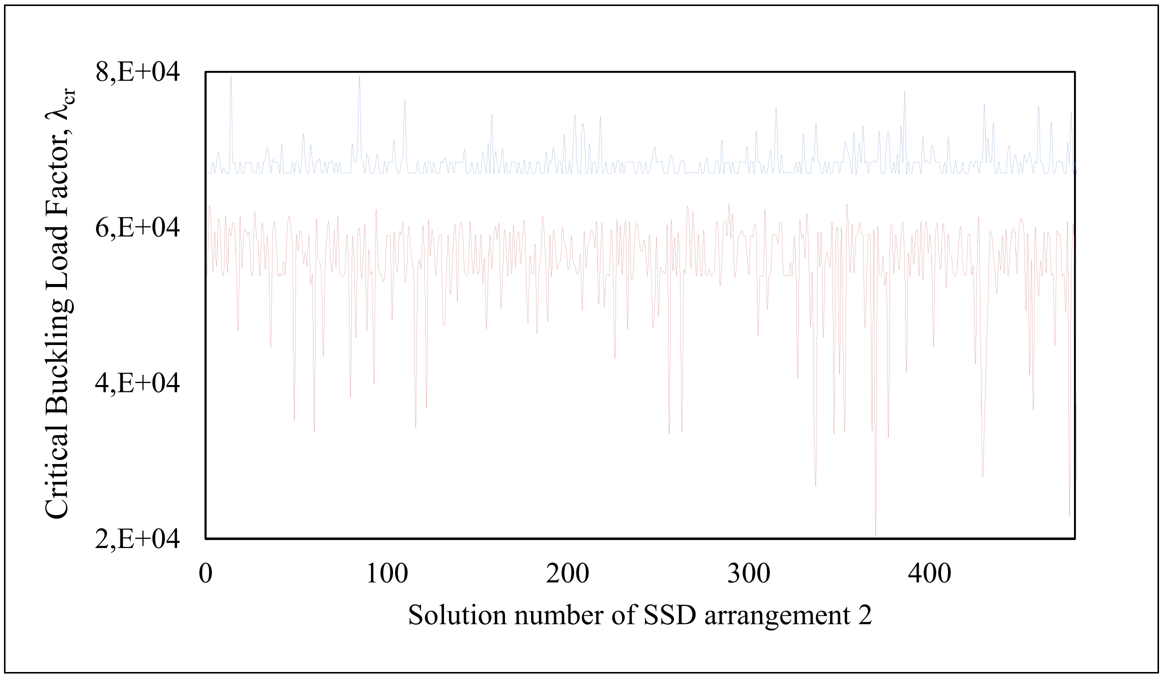


Figure 6.21 The anti-optimization procedure iteration vs. critical buckling load factor for SSD2

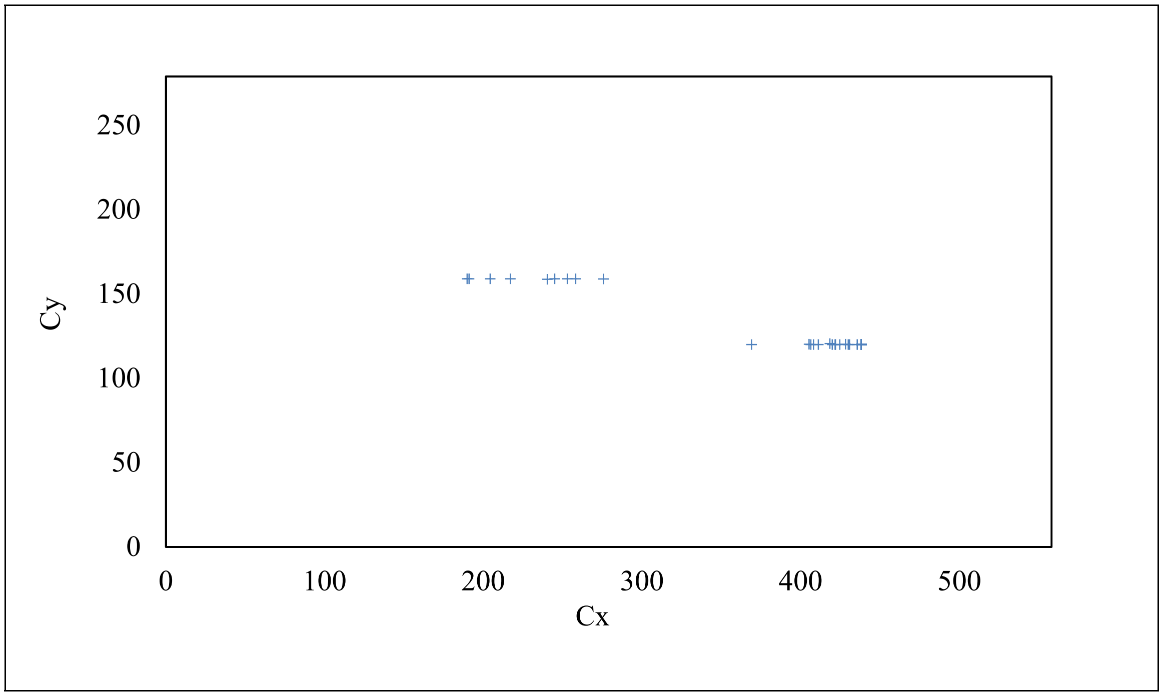


Figure 6.22 Cut-out center points of SSD3

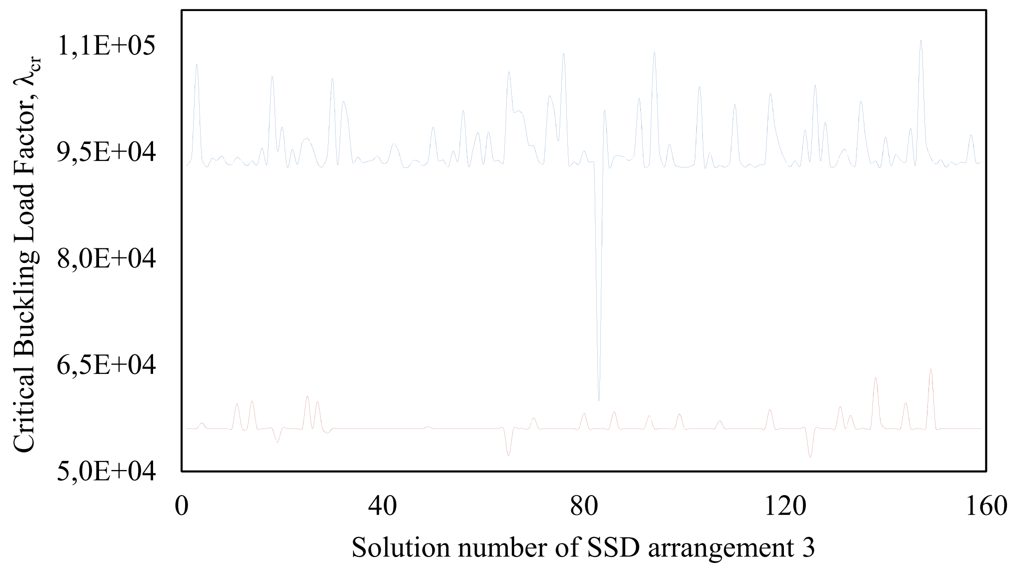


Figure 6.23 The anti-optimization procedure iteration vs. critical buckling load factor for SSD3

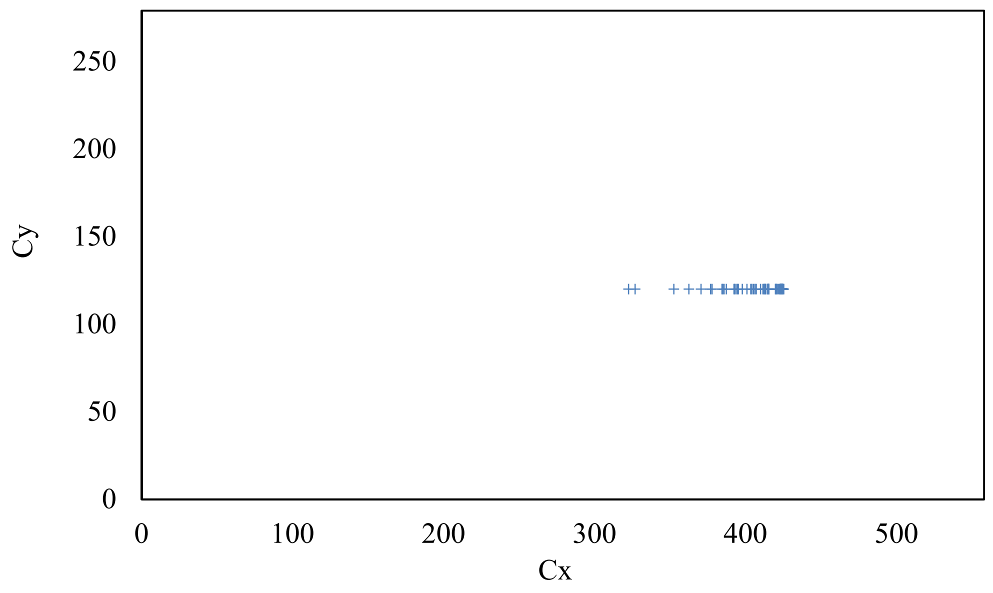


Figure 6.24 Cut-out center points of SSD5

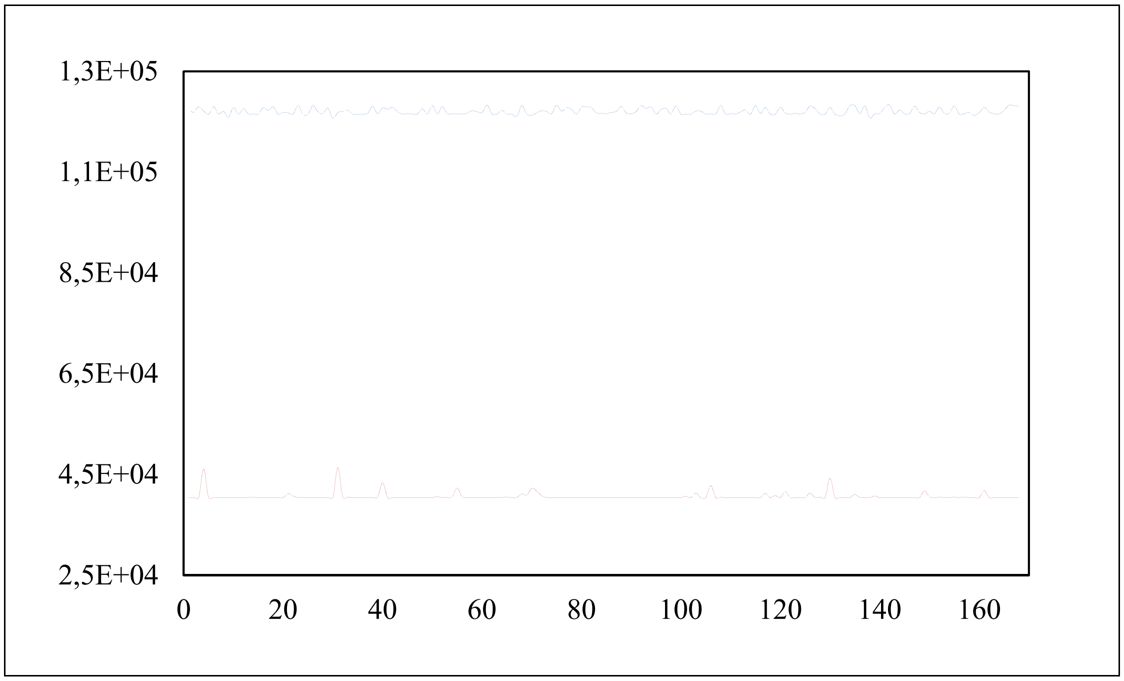


Figure 6.25 The anti-optimization procedure iteration vs. critical buckling load factor for SSD5

The anti-optimization results of SDD2 exhibited a stable minimum value of λ_{cr}, around 67000 on average, while the minimum values of the anti-optimization level appeared to have extremely fluctuated. Furthermore, the gap between the minimum optimization and maximum anti-optimization is smaller compared to SSD3 and SSD5.

However, the difference between the optimization and anti-optimization objective functions never been constant for the current case study of perforated composite laminate; thus, the optimal solution considered for the solution with simultaneously minimum of optimal and anti-optimal objective function values. Accordingly, the SSD3 is the robust optimal solution for uncertain loading within an interval of 0 to 1 (N/m) for any center point of the cut-out around P2 and a diameter ratio of 0.8. Eventually, the appointed optimal SSD is implicitly able to receive nominal critical buckling loads varied from 20 to 67 KN/m.

6.7 Conclusion

The current study introduced a cost-effective, robust optimization framework to optimize the SSD that maximize the critical buckling load factor of a perforated composite laminated plate subjected to uncertainty influences of buckling loading and cut-out center point location. The framework adopted the anti-optimization approach to consider uncertainty influences. The main idea behind the anti-optimization is searching the design space of the problem to find the worst-case scenario (anti-optimization) for the current optimal SSD. Then it is followed by new searching for another optimal SSD (optimization). This process is iterative, and it needs a large number of objective function evaluations. Thus, it becomes a very expensive choice compared to deterministic or probabilistic design frameworks. However, the emerging utilization of the ANN technique in solving different engineering problems can give the anti-optimization approach a significant push that makes it a feasible alternative as a cost-effective design framework. Here, the ANN technique is used to develop an accurate prediction of critical buckling load factor values of a perforated composite laminate plate subjected to uncertainty conditions. The ANN trained based on a set of input data, x, sample generated by Latin Hypercube plan and their design responses, y, were determined by commercial FEA software of ANSYS workbench. The validation results demonstrated a significant agreement between the ANN predictions and Black-box function (ANSYS) results.

Though the proposed framework is created through two steps, first, a preliminary investigation has been done to determine the diameter and location of the cut-out that minimizes the critical buckling load factor under certain loading conditions. The purpose of this step is to find the initial worst-case to start the anti-optimization procedure. The Adapted Discrete Cuckoo Search Algorithm (ADCSA) is used to optimize the SSD of the perforated composite laminated plate for maximum critical buckling load factor while a Simulated Annealing algorithm is used to anti-optimize the obtained optimal SSD.

The results of the numerical experiments revealed that there are 17 SSD options appeared during the anti-optimization tour of 1000 iteration. Moreover, the 90°_{2} fiber orientation has been found to be the major orientation selected for all optimal SSD solutions. Also, it noticed that most anti-optimal solutions occurred with bidirectional loading conditions when the Ny loading is high (or even the maximum load) while a smaller load is applied in the x-direction. Lastly, the selected optimal SSD is implicitly able to receive nominal critical buckling loads varied from 20 to 67 KN/m for the considered uncertainty conditions.

CONCLUSION

The thesis's preceding chapters, having demonstrated the importance of the optimization technique selection based on a comprehensive performance assessment criterion and how this could affect the structural design optimization process in terms of solution quality and cost. Furthermore, the thesis raises the gap between the practical design problems and those academically presented.

The literature review's chapter shows the mass development of new MHs to solve different benchmarking SDO problems. Even though the importance of using fair assessment criterion to evaluate the MHs performance, regrettably, it has received a little attention, reflecting a wider failure of implementing MHs to solve practical design problems. However, this research opportunity and others presented in the literature review chapter, motivated the current thesis to determine answers for the following research questions:

- How does the performance of different MHs could be fairly assessed to determine the suitable MH that solves a real-life SDO problem efficiently?
- Does the designated MH qualify for further performance improvement such that it becomes more competitive in terms of solution quality and computational cost?
- How does a robust optimization framework could be developed so that it can handle a structural optimization problem associated with an expensive objective function, and concurrently it considers the influences of different design uncertainties?

After that, CHAPTER 2 introduces the methodology framework pursued here to develop the required knowledge to answer the research questions. The different components of the framework are explained in more detail; therefore, the methodology of achieving the predefined objectives is clarified. Eventually, the summary of the research achievements and novelty are presented in the last section.

A real-life application of customized I-beam overhead gantry crane is introduced in Chapter 3 as a novel SDO problem. The crane is composed of three rectangular plates, with the same length and different thicknesses and widths, welded together by full penetration welds over the span length to form an I-Beam profile. The thicknesses and widths of plates must be optimized to have the minimum cross-section area while respecting yield, buckling, deflection and fatigue criteria. A mathematical procedure based on the Timoshenko beam theory and Crane Manufacturers Association of America (CMAA) combined with the Genetic Algorithm (GA) is presented, and a Mathcad code is implemented to find the optimal I-Beam cross-section dimensions.

In response to the first two research questions, a comprehensive assessment criterion has been initialized in CHAPTER 4 and then extended in CHAPTER 5. The basic version of the assessment criterion has been used to measure the performance of a novel variant of Cuckoo Search (CS) MH in CHAPTER 4. The proposed variant, called the Adaptive Discrete CS Algorithm (ADCSA), uses the rank-value approach to turn real values of random Lèvy walks (steps/jumps) into their equivalent discrete values. Moreover, the ADCSA intensification effort was enhanced by adding four different local search movements of permutation, swap, insertion and bit flip. The performance of the final version of ADCSA validated across a well-known benchmark problem of the composite laminated plate. Then, ADCSA employed to optimize a discrete version of the customized I-beam crane introduced in CHAPTER 3.

The MHs performance assessment criterion has been developed in CHAPTER 5, where it extended to include more efficient measures such as the practical reliability, price (computational cost), normalized price, performance rate, solution quality and Fitness-landscape analysis. Additionally, two different convergence rates were imposed on examining the MHs at slow and fast rates. As well as the reproducibility of the numerical experiments results considered within the procedure of MHs assessment. Thereafter, the proposed criterion has been employed to compare five different variants of Ant Colony Optimization (ACO). The proposed measures demonstrated a comprehensive assessment of the compared ACOs performance. The initial results of the comparison study reveal that the Hyper-Cube

Framework (HCF) ACO variant outperforms the others. Consequently, an investigation of further improvement led to introduce an enhanced version of HCFACO (or EHCFACO). The new variant has advanced intensification features that use insertion and bit-flip movements to enhance the local search effort. Eventually, the EHCFACO variant was compared with other ACO variants, and it exhibited a significant performance.

In answering the last research question, a robust design optimization framework was introduced in CHAPTER 6. The framework adopted the anti-optimization approach to consider uncertainty influences. The main idea behind the anti-optimization is searching the design space of the problem to find the worst-case scenario (anti-optimization) for the current optimal solution. Then it followed by new searching for another optimal solution (optimization). This process is iterative that needs a large number of objective function evaluations, which make it a very expensive choice compared to deterministic or probabilistic design frameworks. Thus, the cutting-edge Deep Learning algorithm of Artificial Neural Network (ANN) has been adopted in this chapter to predict the design response of a perforated composite laminated plate subjected to buckling loading. The ANN trained based on a set of input data, x, sample generated by Latin Hypercube plan and their design responses, y, were determined by commercial FEA software of ANSYS workbench. The validation results demonstrated a significant agreement between the ANN predictions and Black-box function (ANSYS) results. However, the proposed framework is created through two steps. First, a preliminary investigation that has been done to determine the diameter and location of the cut-out that minimizes the critical buckling load factor under certain loading conditions. The purpose of this step is to find the initial worst-case to start the anti-optimization procedure. The ADCSA MH, which developed in CHAPTER 4, is used to optimize the SSD of the perforated composite laminated plate for the maximum critical buckling load factor. In contrast, a Simulated Annealing (SA) algorithm is used to anti-optimize the obtained optimal SSD.

The rest of the conclusion section is devoted to presenting the thesis findings and list the main contributions. It then continues to address the thesis's overall implications and limitations. A possible perspective research opportunity has been outlined at the end of the section.

Findings

The optimal solution of customized I-beam crane presented in CHAPTER 3 reveals interesting observations. The optimized custom I-section has a configuration of narrow and thick lower flange, thinner and wider upper flange, and the web is tall and very thin, which could save about 18% of weight compared to commercial standard I-Beam. Furthermore, it is found that the constraints of general lateral buckling and local buckling of the upper flange are always reached for all examined cases. The web local buckling constraint is critical for about 66% of cases; the yield and fatigue constraints found critical for 33% of cases, and the deflection constraint is not a problem at all. Another impressive observation was that the discrete version of the crane, presented in CHAPTER 4, followed the same pattern of the cross-section configuration of the original crane design problem. Moreover, the equivalent standard I-beam that examined across the design criteria demonstrated a poor strength response and even violated one or more constraints on several occasions.

Using the elapsed time as a performance measure of MH does not reflect the actual computation cost required by MH to find the optimal solution. Based on findings of the proposed assessment scheme presented in CHAPTER 5, it noticed that MHs with high-reliability solutions need more time to find the global optima but need relatively a small number of computation iterations. This observation led to another important remark, the MHs that shown good performance are hardly exploring the design space where their Fitness-Distance Correlation, r, figures become lower than the other MHs. However, the difference in r values is noticed, but it does not negatively impact the overall performance of the designated MHs, except increasing the computational time. Furthermore, it is observed that applying the HyperCube Framework (HCF) to standard ACO has a significant influence on the overall performance of ACO. Also, imposing local search movements, as an enhancement of exploitation effort, helped HCFACO to deliver a cost-effective solution. These improvements in ACO performance are in line with suggestions made by previous studies that rewarding HCF and local search movements the dominant factor in improving standard ACO algorithm performance.

CHAPTER 4 presented a novel discrete variant of ADCSA that integrated a rank/value to turn the continuous domain of the design problem into a discrete domain. One of the significant relevant findings to extract from this chapter is that using rounding to the nearest discrete value has a negative impact on the computational cost of the optimal solution. The optimal solution using the rounding approach has cost more than twice of the rank/value-based optimal solution. Furthermore, the initialization methods experiment conducted in CHAPTER 4 illustrated a slight effect of the initial population generation on the performance of ADCSA. The LHC sampling approach improved the reliability slightly compared to ADCSA that initialized using DUD or Hybrid DUD-LHC.

A robust optimization framework has introduced in CHAPTER 6. One of the most obvious findings to emerge from this part is that employing the ANN algorithm has tremendously reduced the associated cost of the objective function evaluation. Furthermore, the enumeration of the possible SSD configurations that can be obtained for the designated laminate produced 729 configurations. Despite this number of available configurations, the robust optimal solution is limited to just 17 SSD configurations. Moreover, the 90°_{2} fiber orientation has been found to be the major orientation selected for all optimal SSD solutions. Also, it noticed that most anti-optimal solutions occurred with bidirectional loading conditions when the Ny loading is high (or even the maximum load) while a smaller load is applied in the x-direction.

Limitations

A number of important limitations need to be considered. The current study has examined the buckling loading conditions only and this could be referred to the inclination of the current thesis to raise awareness of the consequence of underestimating the critical buckling loading. Not surprisingly that minimizing the weight is common practice in the SDO domain. Thus, the buckling failure mode should get enough attention from the designer where the lighter structures are more exposed to failure due to buckling than other loading conditions.

The critical design analysis represents an important interest in the current work. Accordingly, using simply supported edges conditions for plates examined on different occasions in this thesis could be explained in this direction. Based on classical plate theory, the lowest values of critical buckling load factor are going to cases with simply supported edge conditions; thus, the attained optimal design still valid for other edge boundary conditions. However, it is not necessary, in real-life applications, to optimize against such excessive edge conditions.

Even though the MHs literature is full of studies that approve the noticeable impact of parameters tuning on the MHs performance, they considered constant here for all examined ones. This perceived ignorance of such an aspect refers to the associated computation cost of this repetitive investigation, and such an exhaustive practice would diminish the scope of the thesis significantly.

Future Work

It is recommended that further research be undertaken in the following areas:

- The proposed ADCSA and EHCFACO have been applied to view different SDO problems so far and examining them for other structural optimization problems could be prospective work. Furthermore, investigating the different initialization methods on the proposed algorithm to find better performance deserves a try.

- Another possible area of future research would be to investigate the performance of the proposed approach of a robust design optimization framework to handle multi-objective SDO problems.

- Further research is needed to examine more closely the performance of the new proposed metaphor-based MHs. The MHs performance assessment criterion which developed here be used for such task.

- A future investigating study of the performance of the developed MHs, ADCSA and EHCFACO, for SDO problems that consider the thermal buckling loading would be very interesting.

Summary of Achievements and Novelty

The current study made novel contributions to the research domain of SDO that we could summarize in the following points:

- The work in CHAPTER 3 presented a new optimization approach that could be used to solve different SDO problems. The basic purpose of the developed approach was promoting of the built-in optimization tools in ANSYS Workbench software. However, the approach has been improved through the thesis sections and it became a cornerstone of more sophisticated SDO framework that presented in the CHAPTER 6.

- Develop a new discrete variant of CS MH that uses an adaptive technique to adjust step/jumps of Lévy flights, which used to create the new solutions in CS optimization. Furthermore, four different local search movements were integrated into the proposed ADCS MH, see CHAPTER 4.

- Present an Enhanced HCF-ACO (EHCFACO) variant as a novel optimizer for SDO problems. The standard HCF-ACO variant never examined before as a solver of any SDO problems, up to the best of our knowledge, even though it used successfully to solve other NP-hard combinatorial optimization problems, see CHAPTER 5.

- Establish a comprehensive performance assessment criterion for MHs used to solve SDO problems. The new criterion introduced new performance measures such as Fitness-Distance correlation factor, performance rate and solution quality. Additionally, the MHs performance was examined at different levels of convergence rate (slow, fast). Theses measures used together with traditional performance measures of computational price, computational time and successful rate (reliability) for the first time to evaluate the MHs performance as SDO problems optimizer. Lastly, the comparison criterion examines all numerical experiments for 200 times each and for ten different seed numbers. The number of experiments devoted to overcoming the stochastic behaviour of MHs while using different seeds number aims to make the assessment results able to be reproduced, see CHAPTER 5.

- Propose a new robust design optimization framework to solve NP-hard SDO problems using the anti-optimization approach. The proposed framework uses ANN models to predict the value of the expensive objective functions of the SDO problem, see CHAPTER 6.

- Lastly, introduce two novel SDO problems of customized I-beam profile overhead gantry crane and a perforated composite laminated plate. Both new SDO problems developed based on two different materials. The new SDO problem presented in two individual published journal papers, see CHAPTER 3 and CHAPTER 4.

Moreover, some other sub contributions have been introduced in the current work, such as examining the impact of random numbers generator of the initial solution on the final solution obtained by MH. All the Matlab routines of different MHs implemented here are available in the appendices, and soon they will be uploaded online as open-source repositories for interested researchers.

Published work

Eventually, the thesis has led to five journal publications, three of them are published while two are submitted. Additionally, four conference papers (and extended abstracts) have worked out, and the details of all publications are listed here below.

Journals

- Ahmid, A., Le, V., Dao, T. (2017). An Optimization Procedure for Overhead Gantry Crane Exposed to Buckling and Yield Criteria. *IRA International Journal of Technology & Engineering* (ISSN2455-4480),8(2),28-38. DOI :http:// dx.doi.org /10.21013/jte.v8.n2.p3

- Ahmid, A., Dao, T. M., & Van Ngan, LÊ. (2019). Comparison Study of Discrete Optimization Problem Using Meta-Heuristic Approaches: A Case Study. *International Journal of Industrial Engineering*, *1*(2), 97-109.

- Ahmid, A., Dao, T. and Le, V., (2020). An Adaptive Discrete Cuckoo Search Algorithm to Solve Structural Optimization Problems. *Journal of Multidisciplinary Engineering Science and Technology (JMEST), 7(6)*.

- Ahmid, A., Le, V., and Dao, T. (2020). Enhanced HyperCube Framework ACO For Structural Combinatorial Optimization Problems (submitted to Elsevier Composite Structures journal).

- Ahmid, A., Le, V., Dao, T., Optimization of Perforated Composite Laminated Plate Subjected to Uncertain Geometrical and Loading Conditions (submitted to Computers & Structures journal).

Conferences

- Ahmid A., Le V., & Dao, T. (2017). Optimization procedure for an I-beam crane subjected to yield and buckling criteria. In 2017 World congress on advances in structural engineering and mechanics (ASEM17) (pp. 1–12). Ilsan, Seoul, Korea: ASEM.

- Ali Ahmid, T.M.Dao and V.N.Le (2019). Optimization of the Mechanical Structures Design Problem by Meta-Heuristics Approach: A Case Study. Industry, Engineering and Management Systems Conference (IEMS). California, US (extended abstract).

- Ahmid A., Le V., & Dao, T. (2019). Composite Plate Design Optimization Using Enhanced Hyper-Cube Ant Colony Optimization Algorithm. In NAFEMS World Congress (NWC). Quebec, Canada.

- Ali Ahmid, T.M.Dao and V.N.Le (2021). Enhanced Hyper Cube Framework ACO For Structural Combinatorial Optimization Problems. Proceedings of the International Conference on Industrial Engineering and Operations Management. Singapore, March 9-11, 2021 (extended abstract).

APPENDIX I

COMPARISON STUDY OF DISCRETE OPTIMIZATION PROBLEM USING META-HEURISTIC APPROACHES: A CASE STUDY

A.Ahmid [a] , T. M. Dao [b] and V. N. Lê [c]

[a,b,c] Department of Mechanical Engineering, École de Technologie Supérieure, 1100 Notre-Dame West, Montreal, Quebec, Canada H3C 1K3

APPENDIX IPaper published in International Journal of Industrial Engineering and Operations Management (IJIEOM), Volume. 1, No. 2, pp. 97-109, December 2019

Abstract

This paper presents the performance comparison of five meta-heuristic algorithms to solve a discrete optimization problem. The comparison is undertaken for a case of simply supported plate subjected to biaxial loading conditions. Furthermore, the optimization objective is to determine the optimal stacking sequence design of a laminate that maximizes the critical buckling load factor (λ_{cb}). The chosen meta-heuristics have been implemented using MATLAB with the same convergence criteria and the same maximum number of iterations to ensure a fair comparison. The implemented assessment criterion has performance measures of average CPU time, solution price, reliability, and normalized price. The results have demonstrated the outperformance of the Ant Colony Optimization Algorithm (ACOA) over other algorithms, which confirms the findings of previous studies. Moreover, the Tabu search algorithm (TS) and the Discrete Particle Swarm Optimization algorithm (DPSO) performed poorly due to their limited exploration capability. Additionally, the Genetic Algorithm (GA) and the Simulated Annealing algorithm (SA) exhibited a high level of reliability but showed an expensive solution cost. This study presents an adequate comparison approach of meta-heuristics, where it extends the comparison scope to cover the performance analysis of meta-heuristics more than that previously done in the domain of stacking sequence design optimization.

Keywords: optimization, meta-heuristic, composite laminated plate, buckling load factor.

Introduction

The high competition in production design puts pressure on the designers to introduce a good quality and low-cost product that comply with the engineering standards. With limited resource context, the designer needs to benefit from all available resources and optimization techniques can solve such an issue. In early time, the gradient optimization techniques were commonly used in different engineering design applications due to their fast and accurate solutions. But with growing complexity and variety of applications, they became costly or incapable to find the Optima. On the other hand, metaheuristics exhibited a significant performance in solving optimization problems where the gradient methods failed to do so. The optimization of composite laminated structure design is an excellent example of such type of optimization problems (Ali Kaveh, 2017).

Composite laminated structures are usually formed by laying several thin layers (plies) on top of each other and binding them with a matrix material. The combination of layers and matrix is called a laminate, which consists of microscale-oriented fibers that emerge in the matrix material. The matrix material distributes and transforms the load over the fibers. Additionally, the tensile strength of the fibers is high in their orientated direction, whereas the matrix material has a high compression strength in any direction but has low tensile strength. The designer should select the right combination of both materials to achieve the optimal design. Several design variables should be appropriately determined such as a number of layers, the thickness of each layer, ply orientation angle, and stacking sequence that ensures the highest possible performance of the structure subjected to specific loading conditions (R. M. Jones, 2014; Vasiliev, 2017).

Buckling failure mode occurs suddenly when the composite laminated structure is exposed to compressive loading that exceeds a particular critical value. This failure mode is dangerous, especially for applications such as airplanes and ships, where human lives become threatened. The designer of composite laminated structures then must try to increase the capacity of the structure to bear the buckling load through optimizing the structure parameters. When the thickness of the structure is constant, the stacking sequence of the laminate turns into the significant design variable that can maximize the critical buckling loading of the structure (Nikbakt et al., 2018).

The optimization of the laminated composite structures is subjected to the design and manufacturing constraints such as a limited number of fiber orientations. The optimization of the laminate then becomes a hard combinatorial optimization problem (Peeters & Abdalla, 2017; A. R. M. Rao, 2009; Zein et al., 2016). Ghiasi et al. (2010) reviewed different techniques used in the recent decades to optimize composite laminated designs and concluded that meta-heuristics are superior to gradient-based methods. Furthermore, (Nikbakt et al., 2018) reported the outperformance of meta-heuristics alongside gradient optimization algorithms due to their efficiency and stability. However, meta-heuristic algorithms are an ongoing optimization research domain to solve medium as well as large-scale problems that appear in different disciplines. Furthermore, trajectory-based meta-heuristics demonstrated a substantial local search capacity on its track to find the optimal solution, whereas population-based meta-heuristics exhibited a significant ability to explore the design space. Even though meta-heuristics, in general, could solve the discrete optimization problems efficiently, we still need to determine which algorithm outperforms the others for a specific problem according to the No Free Lunch theorem (NFL) by Wolpert and Macready (1997).

The literature is full of comparison studies of meta-heuristics that have been used to solve various engineering problems, e.g., TSP and scheduling, but little were devoted to the stacking sequence design problem. Furthermore, the previously published papers in the field were limited to two comparison approaches. First, the comparison of a newly developed algorithm (or enhanced version of a well-known algorithm) to previously published results of another meta-heuristic (Aymerich & Serra, 2008; Jing et al., 2015). Second, the selection of more than two algorithms and carrying out the performance comparison based on the author implementation of the meta-heuristics (M. W. Bloomfield, J. E. Herencia, & P. M. Weaver, 2010). Both approaches brought valuable information that increased the knowledge about meta-heuristics performance as an optimizer of stacking sequence design. Although these comparison approaches are interesting, they still have some drawbacks such as the diversity of convergence criteria for the compared algorithms as in the first approach or the comparison limitation to one category of meta-heuristics as in the second approach.

To avoid this shortcoming, five different meta-heuristics were selected in this study to represent both population-based and trajectory-based meta-heuristics. The chosen algorithms frequently appeared in the literature of stacking sequence design optimization (Nikbakt et al., 2018). The five meta-heuristic algorithms have been implemented using MATLAB with the same convergence criteria and the same maximum number of runs to ensure a fair comparison. An assessment criterion has been performed by considering different performance measures such as average CPU time, reliability, and normalized price. Additionally, a well-known benchmarking problem was selected as a case study to carry out the comparison(A. Kaveh, Dadras, & Malek, 2017; R. Le Riche & Haftka, 1993). Eventually, the overall objective of this work is to develop an improved knowledge of optimization techniques and the selection of the most efficient algorithm that solves the stacking sequence design optimization problem.

Meta-heuristic Algorithms

Meta-heuristics are known as stochastic approaches that are frequently used in solving complex optimization problems. There are many classifications of meta-heuristics, and we have adopted the one illustrated in Figure-A I - 1, which classifies meta-heuristics into two categories population-based and trajectory-based. Moreover, five different algorithms were selected to represent both categories of meta-heuristics. Genetic algorithm (GA), Ant Colony Optimization (ACO), and Particle Swarms Optimization (PSO) are population-based meta-heuristics, whereas Simulated Annealing (SA) and Tabu Search (TS) are trajectory-based meta-heuristics. This section provides a short review of each meta-heuristic and the implementation structure of the algorithms used in this study.

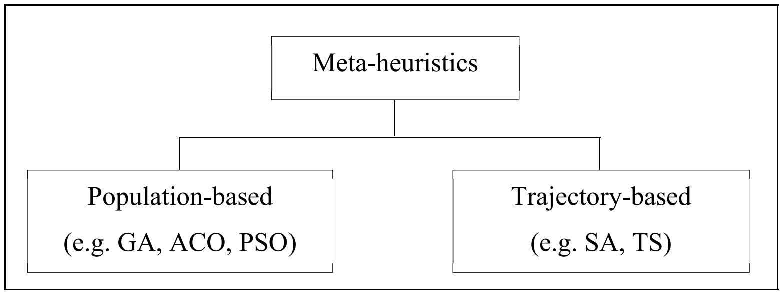


Figure-A I - 1 Metaheuristics classification

Genetic Algorithm (GA)

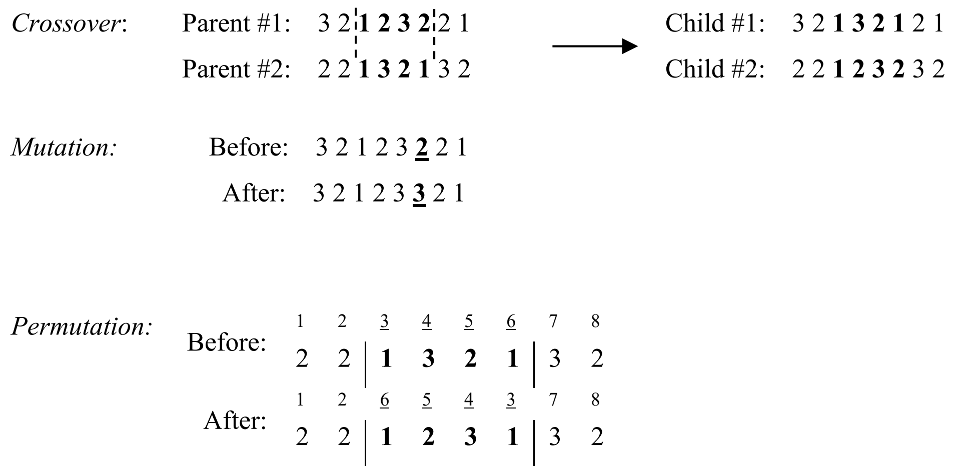
Holland suggested the original genetic algorithm in the 1960s, which was later detailed in its generally known form by Goldberg (1988). It is based on Darwin's theory of natural evolution, and it is implemented using elements of the natural genetics of reproduction, crossover, and mutation. GA shows its worthiness over classical optimization methods in solving composite laminated design optimization problems (Nikbakt et al., 2018). The significant adaptation of GA to optimize composite laminate design is credited to Le Riche (1993), as he proposed a modified GA that replaces binary coding of solution strings by integer coding. This formulation turned the binary GA algorithm into Permutation Genetic Algorithm (PGA). The results show a 2% reduction in the solution cost compared to binary GA(R. Le Riche & Haftka, 1993). The gene-rank GA introduced by B. Liu et al. (2000) is a permutation GA with gene-rank crossover operator. He compared his proposed GA with standard GA and older permutation GAs, and the gene-rank GA demonstrated better computational performance. Furthermore, Ehsani et al. (2016) used binary GA to determine the optimal stacking sequence of grid laminate by considering the different boundary conditions of the laminate edges. Moreover, GA algorithms are known for their expensive solution due to the slow convergence to the optimal solution. To overcome such drawback, Vosoughi et al. (2017) made hybrid GA with PSO algorithms as an operator to increase the convergence rate of standard GA. However, binary GA is still used as stacking sequence design optimizer. It offers a costly solution, while PGA demonstrates good performance for cheaper solutions.

In this study, the PGA structure was selected, as described by Le Riche (1993), to implement a GA program in MATLAB. Algorithm-A I - 1 illustrates the steps of PGA meta-heuristic. The algorithm is initialized by generating a random initial solution, and then it evaluates the fitness of the chromosomes (solutions). Based on their fitness value, the chromosomes are sorted from the maximum to the minimum, and the best-ranked individuals are selected for the reproduction process. To proceed with the reproduction, a pair of best individuals are randomly selected to be parents, and the crossover operator is applied to generate the children (new solution). Mutation and permutation operators are then applied to improve the new population exploration. This loop continues until the termination criteria is satisfied.

Algorithm-A I - 1 Permutation Genetic Algorithm procedure

```
Initialization:
        - Generate initial random population.
        - Evaluate the population chromosomes fitness.
While (termination criteria not satisfied) Do
        - Select best-ranked individuals to reproduction.
        - Randomly select a pair of individuals to be parents
        - Apply crossover
        - Apply mutation to children
        - Apply permutation to children
        - Evaluate chromosomes fitness.
End PGA algorithm for discrete optimization problems
```

In the original PGA, which was proposed by Le Riche (1993), the different solutions have integer representation using 1, 2, and 3 numbers; where 1,2 and 3 represent $0^\circ, \pm 45^\circ,$ and90° fiber orientations, respectively. The laminate with $[\pm 45^\circ, 0^\circ{}_2, 90^\circ{}_2, \pm 45^\circ, \pm 45^\circ, 90^\circ{}_2, 0^\circ{}_2]_s$ stacking sequence have been represented by $[2\ 1\ 3\ 2\ 2\ 3\ 1]_s$. The different PGA operators of crossover, mutation, and permutation have been illustrated as follows:



Ant Colony Optimization Algorithm (ACOA)

Dorigo (1991) developed the Ant Colony Optimization system that is inspired by the natural phenomena of the food searching strategy of the ant colony. He proposed a mathematical model that simulates this strategy of the cooperative attitude of an actual ant colony to find the optimal solution. He implemented his model to solve well-known optimization problems such as the travel salesman problem (TSP). The proposed model consists of four major steps. First, a suitable number of ants is assumed. Second, the probability of path selection is determined. Third, random numbers from 0 to 1 for each ant are generated. This step is repeated for all design variables, and it is followed by objective function evaluation and assessment. Fourth, the model checks the convergence process. ACO has been extended in different engineering areas to solve problems such as discrete structural design or composite laminated structures. Aymerich (2008) investigated the computational efficiency of ACO as an optimizer that maximizes the buckling load of a simply supported plate exposed to uniaxial loading. He compared the solution quality and robustness of ACO with GA and TS algorithms for the same reference case study, and the results show that the ACO algorithm has better performance. Furthermore, Koide et al. (2013) used the ACOA combined with finite element analysis to maximize the buckling load factor. They compared the obtained results of their proposed optimization solution with those previously obtained for GA by Le Riche (1993).

The structure of the proposed ACOA by Aymerich (2008) was mainly considered in the this study, and Algorithm-A I - 2 summarizes the steps of the implemented ACO. The procedure starts with random initial laminate stacking being selected from the feasible solution set (possible fiber orientations). This step is followed by an evaluation of the objective function which will be stored in the ant routing table and used to generate a new feasible stacking sequence. Then, the local search movements of permutation and swap are applied to all generated solutions by ants in order to find better solutions. The local movement of permutation of ACO has the same effect of the permutation operator of PGA whereas the swap (also called two points mutation) movement occurs by randomly selecting and switching positions of two bits of the solution string. Finally, the global pheromone table is updated, according to Eq.(A I- 1), where only the ants with the best solution deposited more pheromone trail on their path to the solution. This procedure continues until the termination criterion is satisfied.

Algorithm-A I - 2 Ant Colony Optimization Algorithm procedure

```
Initialization
    - Initialize the ACO parameters
While (termination criteria not satisfied) Do
    - Construct Initial Solutions Table by Ants
    - Evaluation
    - Local Search:
        ▪ Permutation
        ▪ Swap
    - New Solutions evaluation
    - Apply Pheromones Updating Rule
End ACO algorithm for combinatorial optimization problems
```

The pheromone is updated according to the following rule:

$$\tau_{ij}^{k(t)} = (1-\rho).\tau_{ij}^{k(t-1)} + \sum_{k}^{n} \Delta\tau_{ij}^{k(t-1)} \qquad \text{(A I- 1)}$$

where:

$$\Delta\tau_{ij}^{k(t-1)} = \sum_{k=1}^{m} \frac{f_k}{\sum_{k=1}^{m} f_k} \tag{A I- 2}$$

ρ denotes the evaporation rate, t is the current iteration number, m is the number of the optimal solution ants, and f_k is the fitness value of each ant.

Discrete Particle Swarm Optimization (DPSO)

Kennedy (1995) developed particle swarm optimization (PSO), and it is classified as a population-based metaheuristic. PSO is used to solve non-linear optimization problems with continues domain. PSO mimics the social behavior of a flock of birds, where each bird (called particle) moves with the flock according to two vectors of position and velocity. Each particle updates its position and velocity based on simple vector addition and subtraction until the optimal solution is found. Furthermore, PSO has various forms since it was invented, and the most popular one is known as G-best PSO. Additionally, PSO is known for its significant ability to explore a solution space with a fast convergence rate (Parsopoulos & Vrahatis, 2002). Different variants of PSO were developed to solve optimization problems of various engineering applications. Multiple versions of Discrete PSO (DPSO) were designed to solve combinatorial optimization problems such as stacking sequence optimization (Zadeh, Fakoor, & Mohagheghi, 2018). Chang et al. (2010) proposed a new variant of DPSO called Permutation Discrete Particle Swarm (PDPSO). He used PDPSO to determine the optimum stacking sequence of a laminate subjected to the buckling load criteria.

Algorithm-A I - 3 DPSO Algorithm procedure

```
Initialization:
        -  Generate initial random swarm.
        -  Evaluate the initial swarm speed and position.
While (termination criteria not satisfied) Do
        -  Update swam speed and position
        -  Evaluate the new swarm fitness
End DPSO algorithm for combinatorial optimization problems
```

DPSO algorithm, as described by Zadeh (2018), has been adopted in this comparison study. Algorithm-A I - 3 illustrates the different steps of implemented DPSO. It is initialized by selecting random swarm of particles and a set of possible solutions, and then this swarm fitness is evaluated. The best global position is devoted to the particle with maximum fitness in the initial swarm. The local particle speed and position update according to Eq.(A I- 3) and (A I- 4) to generate a new swarm. The evaluation of the new swarm is then carried out, and the global best position is updated if the fitness of best local position of the new swarm is higher than the fitness of the stored best global position. This loop continues until the termination criterion is satisfied.

The particle speed and positions are updated according to the following equations:

$$X_{k+1}^{i} = X_{k}^{i} + V_{k+1}^{i} \tag{A I- 3}$$

$$V_{k+1}^{i} = wV_{k}^{i} + c_1 r_1 \left(P_{best}^{i} - X_{k}^{i}\right) + c_2 r_2 \left(G_{best}^{g} - X_{k}^{i}\right) \tag{A I- 4}$$

where X_{k+1}^{i} and V_{k+1}^{i} represent the updated position and speed respectively.

Simulated Annealing (SA)

In 1953, Metropolis presented the concept of the simulated annealing algorithm. It is based on the mathematical analogy of the thermal annealing process of critically heated metals. When the heated metal reaches the melting temperature, the molten molecules move randomly concerning each other. Continued reduction of the temperature limits the movement of these molecules and therefore, leads them to be highly ordered until the crystal state is reached, which represents the lowest internal strain energy. The cooling rate has a direct impact on achieving the crystal state; the faster rate will not provide the molecules enough time to form a crystal, and they will attain a polycrystalline state instead, which has higher strain energy. Therefore, the crystallization of molten metals needs a controlled rate of cooling to obtain the lowest strain energy state, and this process is called annealing (Kirkpatrick et al., 1983).

Since its introduction, SA has been used to solve several engineering optimization problems, including stacking sequence design. Lombardi et al. (1992) used SA to optimize the composite laminate buckling load for a plate subjected to biaxial loading under strain limits with iso-oriented and contiguous plies. They considered the range within 0.1% of the best solution as a near or optimal solution in the design space. Erdal et al. (2005) presented an improved version of SA called Direct Simulated Annealing (DSA) to maximize the buckling load factor of the biaxially loaded laminate. DSA was developed by Ali et al. (2002) to handle continuous variable design problems based on memorizing the previous solutions and using a group of points instead of one point in its search for the optimal solution. Erdal (2005) adapted the DSA algorithm to optimize composite laminate design, which is a discrete optimization problem, and he investigated the performance of the algorithm by increasing the difficulty of the problem and increasing the design space size. He demonstrated that DSA performed well even with larger design space, and it overcame the cons of the original SA that was used by Lombardi(1992). Javidrad et al. (2017) proposed a modified SA algorithm that uses the parallelization concept, where the search is performed parallel to the multiple initial points, and the best-found solution is selected as the optimal solution. The convergence speed, for large design spaces, was a result of SA modification.

The structure of the standard SA algorithm was used to implement the algorithm in this comparison study, and the main steps of the implemented SA are listed in Algorithm-A I - 4. SA starts by generating an initial random solution and then computing the objective function value. The initial solution is considered as a current solution, and a new solution is randomly generated about it. The energy of the new solution is determined, which is also known as accepting probability, as shown in Eq.(A I- 5). If the new solution energy is greater than the current solution energy, then the new solution becomes the current solution; otherwise, another new solution is generated. The temperature T is reduced if the SA loop iterations exceeds the certain number of iterations n. These actions are repeated until the termination criteria is satisfied (S. S. Rao, 2009) .

Algorithm-A I - 4 Simulated Annealing Algorithm procedure

```
Initialization:
        - Initialize SA parameters ($T, c, n$)
        - Generate initial random solution.
        - Evaluate the initial solution.
While (termination criteria not satisfied) Do
        - Generate a new solution from the current solution vicinity.
        - Calculate the current solution energy.
        - Calculate the new solution energy.
        - Compare both solutions energy
        - Update the current solution with the biggest.
        - If the number of iterations > $n$
        - Reduce the temperature by reduction factor $c$.
End SA algorithm for combinatorial optimization problems
```

The solution energy level is determining according to Boltzmann distribution probability as follow:

$$P(X) = e^{-\Delta f/kT} \qquad \text{(A I- 5)}$$

where

$$\Delta f = f_{new} - f_{current}$$

K is Boltzmann's constant and T is the initial temperature.

Tabu Search (TS)

Tabu Search (TS) is a local searching algorithm that explores the neighborhood of local optima. This algorithm uses a memory strategy to prevent recycling of old solutions. The original TS was presented by Glover (1991), and since then, it has improved and become widely used in solving combinatorial optimization problems such as TSP. Kaw et al. (2003) employed TS to optimize the stacking sequence of a rectangular laminate subjected to buckling loads. Three different loading cases have been investigated and compared to the previous results obtained using GA. The results illustrate a significant reduction in the solution cost by 25% and 55% for the first and second case, respectively, whereas a slight decrease of 1% was obtained in the third case. Kaw et al. (2003) concluded that TS is a competent optimization tool for stacking sequence problems, but it needs a favorable initial solution.

Additionally, Rao (2007) used TS to enhance the local searching capability of the SA algorithm to optimize the stacking sequence and the new algorithm called TSA, which demonstrated superior performance to GA. Algorithm-A I - 5 explains the structure of the TS algorithm used in the current comparison work. TS algorithm is initialized by generating a random initial solution and then evaluating it. The neighborhood search of the initial solution is carried out by applying three different movements of permutation, swap, and insertion.

Algorithm-A I - 5 Tabu Search Algorithm procedure

```
Initialization:
        - Generate initial random solution.
        - Evaluate the initial solution.
While (termination criteria not satisfied) Do
        - Search the neighborhood:
                • Permutation
                • Swap
                • Insertion
        - Evaluate the new solutions
        - Update the Tabu list.
End TS algorithm for combinatorial optimization problems
```

The first two movements are similar to what have described in GA and ACO, where the insertion movement is imposed by selecting a random bit in the solution string and inserting it between two adjacent random bits. Next, the newly generated solutions are evaluated, and then the maximum fitness value updates the optimal solution. Then, the Tabu list is updated to prevent the next neighborhood search from returning to the previously selected solution. These steps of the algorithm continue until the termination criterion is satisfied.

Optimization Case Study

The general trend in composite optimization problems is usually aiming to achieve the design optimality by considering one of the following aspects: weight minimization, strain energy minimization, or critical buckling load maximization, which intends to improve the structural strength and reduce the design cost (Nikbakt et al., 2018). Composite laminated design variables have a significant impact on the type of optimization problem. For instance, if the optimization problem considers a number of plies as a design variable to be determined, and the other design variables are imposed, the problem is classified as a discrete optimization problem. For such type of optimization problem, the solution could be carried out using some simple methods such as enumeration or branch and bound algorithms (Gürdal, Haftka, & Hajela, 1999). However, the formulation of the problem needs to consider multiple aspects such as optimization level, single or multi-objective, and constraint handling method. Additionally, the following sections illustrate the general elements of the formulation of the stacking sequence optimization to maximize the buckling load problem.

Optimization Problem Statement

This assessment study focuses on the optimization of the stacking sequences that maximize the buckling load (strength) of the laminate as a case study. The objective function of maximizing the buckling load factor for laminate subjected to buckling load conditions could be written as follows:

$$max\lambda_{cb}(p,q) \tag{A I- 6}$$

where λ_{cb} is the critical buckling load factor that buckles a simply supported plate subjected to in-plane loads of λN_X and λN_y into p and q half waves in x and, y directions. λ_{cb} could be defined, with respect to flexural stiffness, as:

$$\lambda_b(\mathrm{p,q}) = \pi^2 \frac{\left[D_{11}\left(p/_a\right)^4 + 2(D_{12}+2D_{66})\left(p/_a\right)^2 + D_{22}\left(q/_b\right)^4\right]}{\left(p/_a\right)^2 N_x + \left(q/_b\right)^2 N_y} \tag{A I- 7}$$

The smallest value of $\lambda_b(p,q)$ is considered as the critical buckling load factor. The critical values of p and q are linked to different factors such as laminate material, number of plies, loading conditions, and the plate aspect ratio. In uniaxial loading of a simply supported plate, the critical buckling load occurs when $p = 1$ whereas in biaxial critical buckling loads, it needs to be determined as the minimum value of $\lambda_{cb}\ (p,q)$ (Söyleyici, 2011).

Solution Representation and Design Space

The most commonly used fiber orientations are $0^\circ, \pm 45^\circ, \text{and } 90^\circ$. In meta-heuristic algorithms, the solution (stacking sequence) takes the form of a bit string that consists of a combination of plies with these angles. The different solutions are integrally coded with 1, 2, and 3 numbers, which respectively represent the three possible fiber orientations. For instance, the laminate with [2 1 3 2 2 3 1]$_s$ stacking sequence describes the laminate of $[\pm 45^\circ, 0^\circ{}_2, 90^\circ{}_2, \pm 45^\circ, \pm 45^\circ, 90^\circ{}_2, 0^\circ{}_2]$s fiber orientations. The simplicity of using an integer representation and the significant performance, makes it the most widely used method in meta-heuristic optimization algorithms for composite laminated design (R Le Riche & Haftka, 1995). The following formula could determine the Design Space Size (DSS) for a laminate represented by N plies:

$$DSS = K^N \tag{A I- 8}$$

where K is devoted to the number of ply orientation angles (e.g., K=3 for $0^\circ{}_2, \pm 45^\circ, 90^\circ{}_2$).

Composite Laminate Design Constraints

The design of the composite should respect certain limitations of manufacturing and specific design considerations. In literature, some rules have been proposed to improve the effectiveness of a laminate design for different applications, (Peeters & Abdalla, 2017; A. R. M. Rao, 2009; Zein et al., 2016). The most used rules are classified and listed below:

- Manufacturing limitations: the thickness of the plies and fiber orientations are limited to the available manufactured values, which are usually integer, for ply thickness or certain angles such as $\pm 45°, 0°$, and $90°$ and for ply orientations. Additionally, the symmetrical laminate makes the manufacturing process more straightforward.

- Strength and stiffness considerations: the symmetry of laminate is necessary to prevent extension-bending coupling $(i.e., B_{i,j} = 0)$. Furthermore, the balanced laminate (which has pairs of plies with the same thickness and different signs of same orientation angle θ) condition is needed to avoid shear-extension coupling $(i.e.,\ A_{16} = A_{26} = 0)$. All the plies with $\pm\theta$ will be grouped to minimize the effect of bending and twisting coupling. Moreover, the congestion of the same orientation plies should be limited to 4 plies for each group to develop a homogeneous laminate and reduce inter-laminate stresses and matrix crack failure. Furthermore, the stiffness degradation can be reduced by devoting 10% of the total number of plies for each orientation angle of 0°, $\pm$45°, and 90°.

Generally, the constraints in stacking sequence optimization with constant laminate thickness t could be treated as:

- Symmetry constraint is enforced by optimizing half of the laminate.
- Balancing constraint is enforced by selecting θ_2 for the standard fiber orientation set of 0°, $\pm$ 45°, and 90°.
- Only $N/4$ ply orientations are needed to describe laminate because of balancing constraints.
- Contiguity constraint is handled by using the penalty parameter (ρ).

Objective Function Transformation

The handling of the constraints is the most critical aspect of the optimization problem formulation. The methods used with the algorithms reviewed here fall under one of the following categories:

- Feasibility-based rule.
- Discrete penalty functions.
- Hybrid approach.

A feasibility-based rule lets the algorithms generate the feasible candidate solutions only and then find the optimum one from them. Furthermore, the penalty functions are widely used in handling the constraints due to their simplicity with consistent results. Hybridization of both the previous methods could lead to an improvement in the performance of the algorithm to find the global optima (R. Le Riche & Haftka, 1993). More details about the constraints handling topic could be found in (Barroso, Parente, & de Melo, 2017; Jiao, Luo, Shang, & Liu, 2014).

$$\lambda = (1-\rho).\max\lambda_{cb}(p,q) \tag{A I- 9}$$

Comparison and Assessment Criteria

In addition to the elapsed time (average CPU time), literature has shown that other measures can be used to measure the computational effort of an algorithm. The first measure is price (P_S), which is defined as the number of objective function evaluations within a search run, and it reflects the computational cost of the search process. The second measure is practical reliability (PR) , and it is defined as the percentage of runs that achieve Practical Optima (PO) at a specific run. The last measure is the normalized price (nP_S) which is defined as the ratio of price and practical reliability. Practical optima PO is defined as the solution within 0.1% error value of the best possible solution (Kogiso et al., 1994; R. Le Riche & Haftka, 1993; Malan & Engelbrecht, 2014).

Benchmarking Numerical Example

MATLAB programs were written for each algorithm reviewed here, which are described in pseudo codes as illustrated in Algorithm-A I - 1 to Algorithm-A I - 5. The literature has shown persistent development of new composite optimization solutions. To verify these new solutions, there is a crucial demand to select the well-known benchmarking problems. The widespread benchmarking problem used in the literature of stacking sequence optimization is accredited to Le Riche (1993), and it is indeed widely used in the reviewed studies of the current work. The original problem describes a simply supported plate subjected to an in-plane biaxial loading, as shown in.Figure-A I - 2.

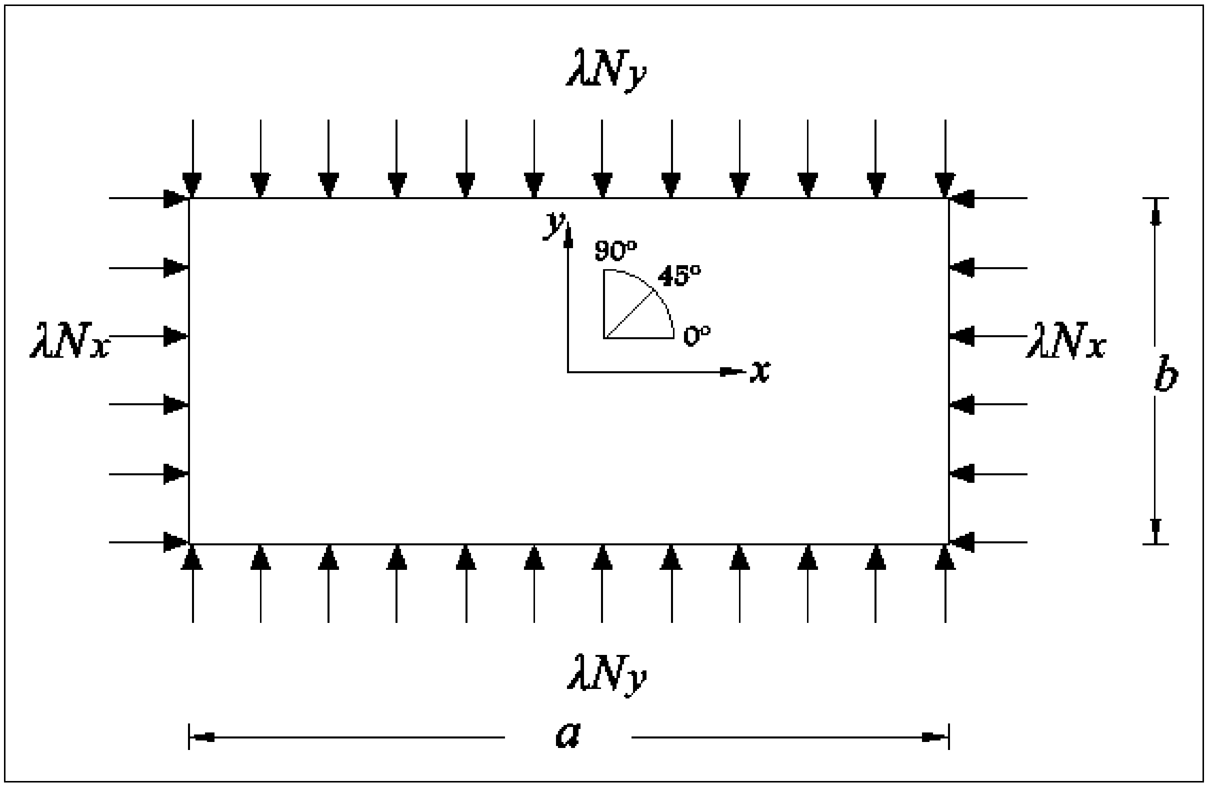


Figure-A I - 2 Simply supported plate subjected to biaxial loading.

The thickness of each ply t is assumed constant, and the plies' orientations are limited to $0°, \pm 45°$, and $90°$ sets of angles. The number of plies N is constant. The required properties, dimensions, and loading conditions are listed in Table-A I - 1 and Table-A I - 2. Furthermore, the objective function is maximizing the critical buckling load. The constraints are integrated to the solution (e.g., balanced laminate, symmetrical, etc.).

Table-A I - 1 Graphite-Epoxy lamina's properties
(R. M. Koide, de Franca, & Luersen, 2013)

Elastic Properties				**Strength Properties**				
E1 (GPa)	E2 (GPa)	G12 (GPa)	ν_{12}	X_T (MPa)	Y_T (MPa)	X_C (MPa)	Y_C (MPa)	S12 (MPa)
127.59	13.03	6.41	0.3	1500	40	1500	246	68

Table-A I - 2 Dimensions and loading conditions of composite laminated plate
(R. M. Koide et al., 2013)

Dimensions				**Loading**	
# Plies NL	**Thickness t (mm)**	**Length a(mm)**	**Width b (mm)**	**Nx (N/m)**	**Nx/Ny**
64	0.127	508	25.4	175	1

The five implemented algorithms were tested on the same machine, as shown in Figure-A I - 3, for the same number of experiments; $Nexp = 200$. This number was used in the original reference case by Le Riche (1993); he used it to tune the PGA parameters and at the same time to determine the performance of his proposed PGA.

```
==========================================================
User: Ali Ahmid                              15-May-2019 00:24:25
==========================================================
    Machine Information:
              CPU Processor: Intel(R) Core (TM) i7-4790 CPU @ 3.60GHz
                CPU clock speed: 3601 MHz
              CPU Cache size (L2): 1024 KB
      Number of physical CPU cores: 4
    Installed physical memory (RAM): 16 GB
             Operating System Type: Windows
           Operating System Version: Microsoft Windows 7 Enterprise
==========================================================
```

Figure-A I - 3 The specifications of PC-machine used in the current comparison study.

An initial random solution was initialized for PGA, SA, and DPSO while TS started with the best solution of 10 random solutions. ACOA began with the initial pheromone of value $.004$. The number of practical optima was determined by considering the near-optimal solutions. In this research, the range of the practical optimal solutions was set to just 0.1% of the global optima.

Results and Discussion

The obtained results for the biaxially loaded laminate are listed in Table-A I - 3. Additionally, the maximum critical buckling load λ_{cb} values were plotted versus their experiment number, as illustrated in Figure-A I - 4. Additionally, the convergence of each algorithm has been graphically illustrated in Figure-A I - 5. According to the introduced comparison and assessment criteria in section 4, different comparison measures of average CPU time, average price, reliability and normalized price were determined and have been illustrated in Figure-A I - 6 to Figure-A I - 8.

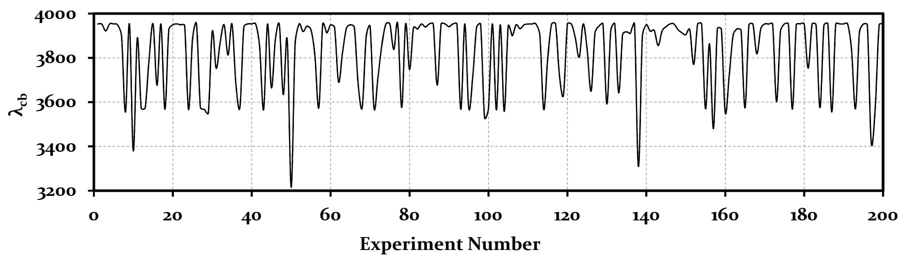
The first four algorithms reached the same global optimal solution, whereas the Tabu Search algorithm missed it slightly, as presented in Table-A I - 3. The optimal stacking sequence followed the same pattern of switching between two groups of 90_2 and ± 45 fiber orientations, which confirms the results of Erdal (2005), Aymerich (2008), and a more recent study by Kaveh (2017); however, 0_2 angle orientations did not exist in the global optimal solution. In Figure-A I - 5, the convergency of trajectory-based meta-heuristics form a series of steps line graph on its way to the optimal solution zone in the design space. On the other hand, the population-based meta-heuristics form a progressive curve graph to converge to the optimal solution. The numerical experiments confirm the random performance fluctuation of the meta-heuristic algorithms due to their stochasticity, as illustrated in Figure-A I - 4.

In terms of computational effort, SA consumed the less CPU time, with just 2.75 sec to complete one run of the algorithm in average, and ACO became the second with 4 sec, while DPSO needed around 32 sec, as shown in Figure-A I - 6. The reliability values are shown in Figure-A I - 7(b) that demonstrates the outperformance of PGA and SA over ACOA, DPSO, and TS. However, in terms of the solution cost, ACOA ranks above all others, as it only costs 87.73 runs on average to reach the global optima with 76.5% reliability. Even though PGA and SA exhibit high reliability, they produced expensive solutions compared to ACOA, as shown in Figure-A I - 8.

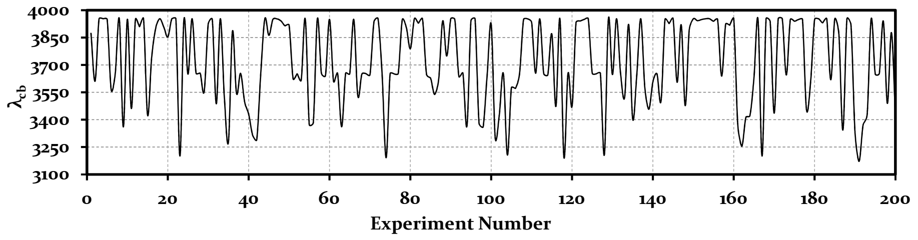
Table-A I - 3 The optimal stacking sequence for 64 ply laminates subjected to biaxial loading without contiguity constraint ($Ny/Nx = 1$ and $a/b = 2$).

Algorithm	**Optimal Stacking Sequence**	**Critical Buckling Load Factor**
PGA	[90_2 90_2 90_2 90_2 90_2 $\pm45_2$ $\pm45_2$ 90_2 $\pm45_2$ $\pm45_2$ $\pm45_2$ 90_2 $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$]	3973.01
ACOA	[90_2 90_2 90_2 90_2 $\pm45_2$ 90_2 $\pm45_2$ 90_2 $\pm45_2$ 90_2 $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$]	3973.01
SA	[90_2 90_2 90_2 90_2 90_2 $\pm45_2$ 90_2 $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$ $\pm45_2$ 90_2 $\pm45_2$]	3973.01
DPSO	[90_2 90_2 $\pm45_2$ $\pm45_2$ 90_2 90_2 90_2 90_2 90_2 90_2 90_2 90_2 $\pm45_2$ 90_2 90_2 90_2]	3973.01
TS	[90_2 90_2 90_2 90_2 90_2 $\pm45_2$ $\pm45_2$ $\pm45_2$ 90_2 $\pm45_2$ 90_2 $\pm45_2$ $\pm45_2$ 90_2 $\pm45_2$ $\pm45_2$]	3972.50

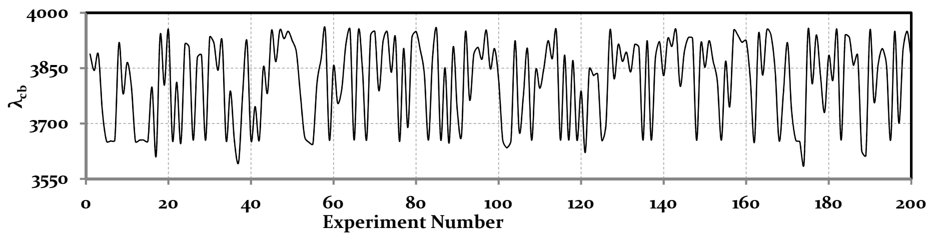
However, the current work is devoted to providing a general overview of the performance of meta-heuristic algorithms. The critical strength factor that has been considered here is the buckling load factor only. Adding the strain failure factor to the optimization criteria could affect the final optimal stacking sequence design. Furthermore, the size of the design space is another factor that could have a significant impact as well (Todoroki & Haftka, 1998).



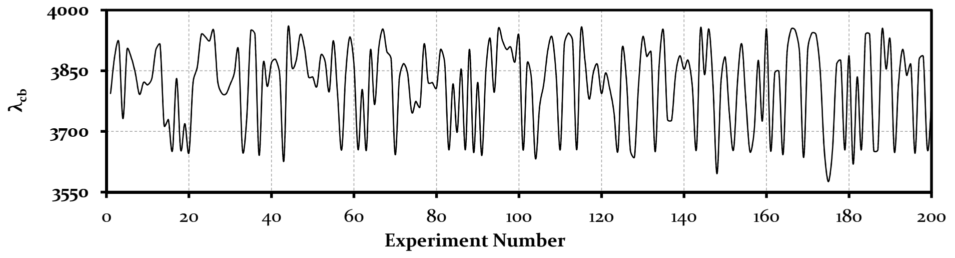
(a) PGA



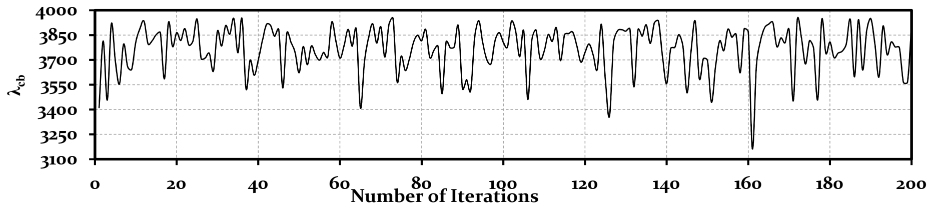
(b) ACOA



(c) DPSO



(d) SA



(e) TS

Figure-A I - 4 : Maximum critical buckling load factor vs. experiment number for the five MHs

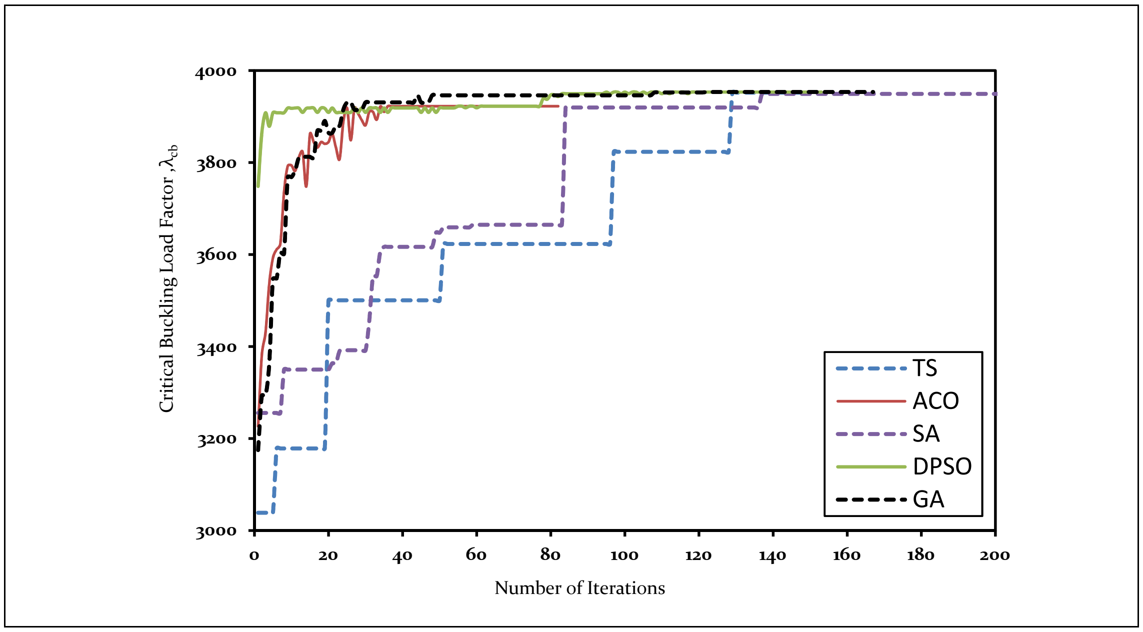


Figure-A I - 5 Meta-heuristics Convergence for Maximizing Critical Buckling Load Factor.

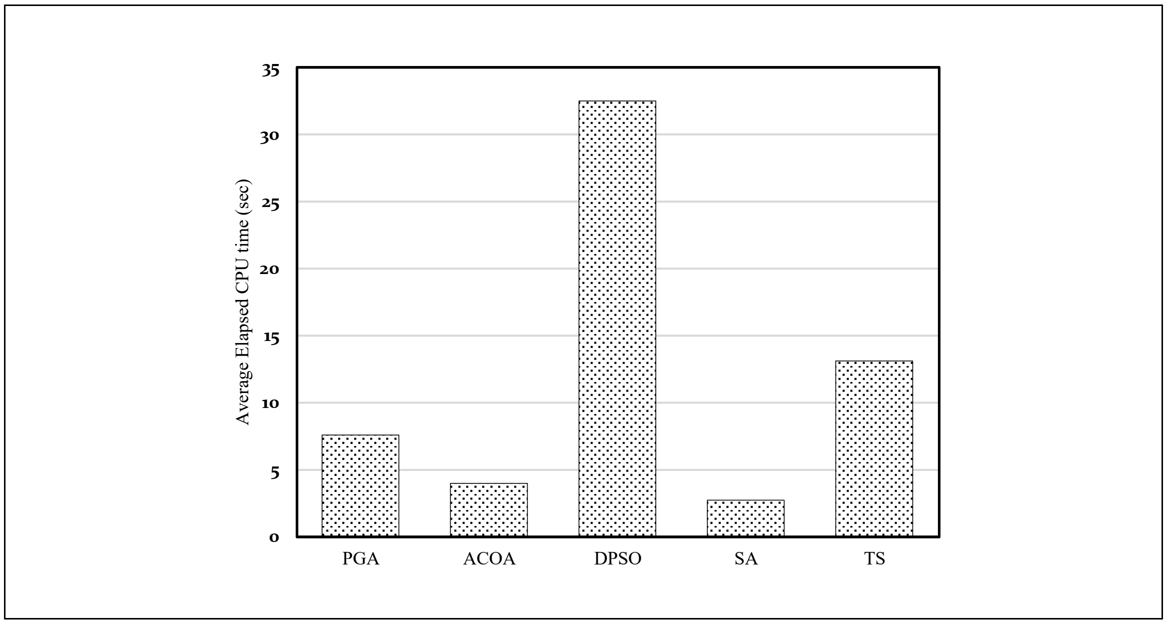


Figure-A I - 6 Average elapsed CPU time for the implemented meta-heuristic Algorithms

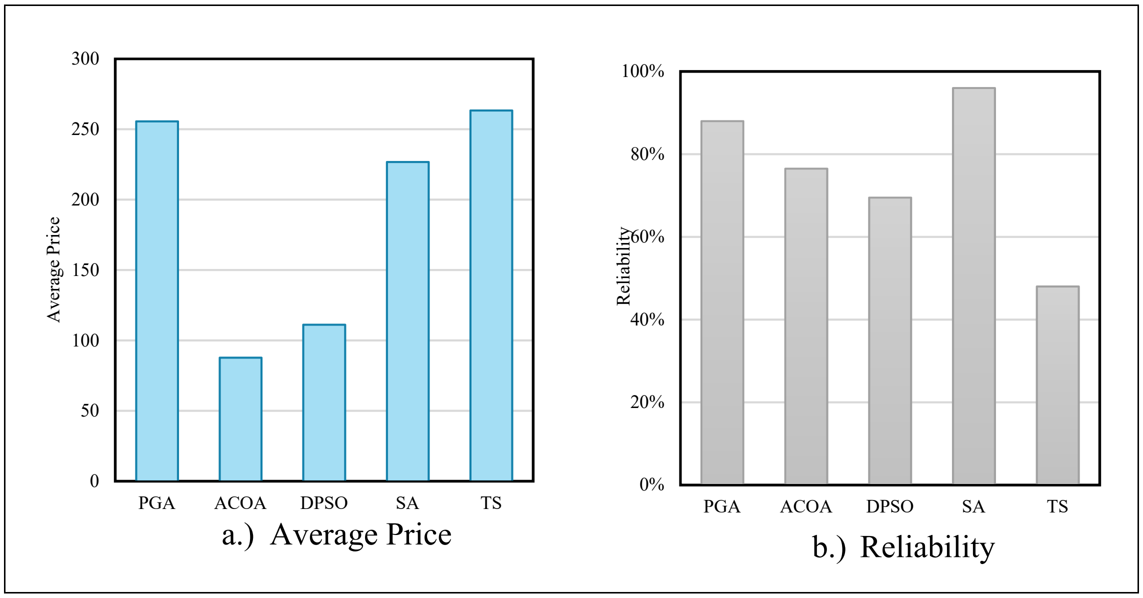


Figure-A I - 7 Average price and reliability for implemented meta-heuristic Algorithms

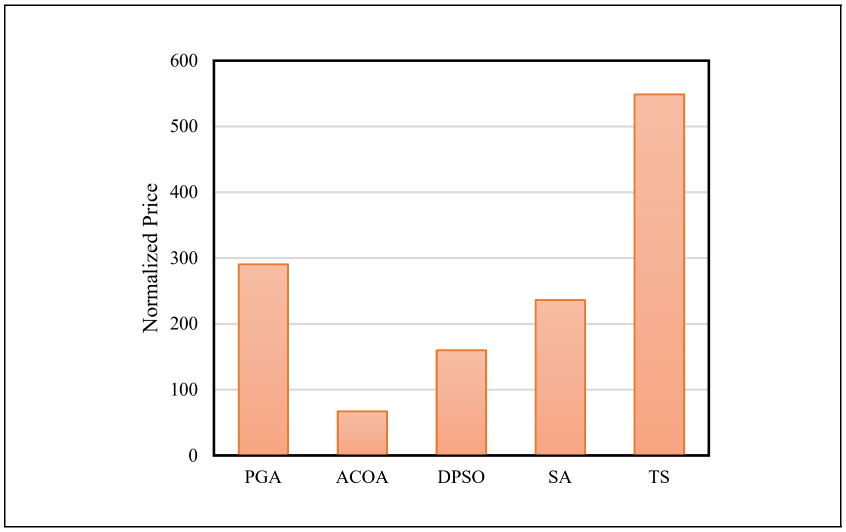


Figure-A I - 8 Normalized price comparison for Meta-heuristic

Conclusion

A comparison of meta-heuristic optimization techniques has been conducted in this article, and the basic knowledge of stacking sequence optimization fundamentals has been introduced.

Five different well-known optimization algorithms have been implemented and examined. This research tried to bridge the gap of previous investigations such as comparing the performance of algorithms from the same category of meta-heuristics. In addition, it applied the same convergence criteria of the investigated meta-heuristics to ensure a fair performance assessment. It may be useful to note that the different meta-heuristics parameters that have been used here were taken from previous studies, and any parameters refinement is out of the current study scope.

The reliability analysis results reveal that GA and SA offer a more reliable solution than ACOA. In terms of solution cost, ACOA ranks above all others, as it only costs 87.73 runs on average to reach the global optima with 76.5% reliability, whereas DPSO, the nearest other meta-heuristic, costs 120 runs on average with 70.15 % reliability. Based on the current case study results, we can conclude that ACOA is a promising algorithm, and this agrees with previous studies of Bloomfield (2010) and Aymerich (2008). The significant performance of ACOA is expected, where it is basically designed to solve discrete optimization problems (M. W. Bloomfield et al., 2010).

ACOA could be improved by integrating other local search movements rather than only relying on permutation and swap movements (Marco Dorigo & Stützle, 2019). GA has low local search performance, which could be improved by combining it with other efficient local search algorithms such as PSO (M. W. Bloomfield et al., 2010). Eventually, further investigations are needed to verify this significant performance of ACOA, such as extend the comparison to include bigger design space or more design constraints with respect to stacking sequence design.

APPENDIX II

PERMUTATION GENTIC ALGORITHM FOR OPTIMIZATION OF COMPOSITE LAMINATED PLATE SUBJECTED TO BUCKLING LOADING

PermutationGA.m

```
function
[xopt,fopt,ige]=PermutationGA(objfunc,nvar,xd,ii,nPop,Ps,Pc,Pm,Imax)
%%% Permutation GA to maximize or minimze f(x)..the permutation
%%% probability is 100%
%   [xopt,fopt]= PermutationGA(objfunc,nvar,xd,sPop,Ps,Pc,Pm,Imax)
%         xopt = Optimal Solution
%         fopt = Optimal solution
%           f  = Objective Function
%        nVar  = no. of design variables
%          xd  = Discrete design variables values vector
%         nPop = Size of the population
%           Ps = Probability of selection
%           Pc = Probability of Crossover
%           Pm = Probability of mutation
%         Imax = max number of generations
%%%-----------------------------------------------------------------------
%
 if nargin < 9, Imax=1000;end   % Number of generations
 if nargin < 8,   Pm=.08 ;end   % Mutation probability
 if nargin < 7,   Pc=1.0 ;end   % Crossover probability
 if nargin < 6,   Ps=0.5 ;end   % Selection fraction
 if nargin < 5, nPop=8   ;end   % Population size

%-------------------------------------------------------------------------
%                             1. Initialization
%-------------------------------------------------------------------------
%
ige=0;
for n=1:nPop,x(n,:)= randi([1 length(xd)],1,nvar); end % generate random
population
for n=1:nPop,funcx(n,:)=feval(objfunc,x(n,:)); end % evaluate all
                                                    % population members.
%-------------------------------------------------------------------------
%                              2. Selection
%-------------------------------------------------------------------------
%
[funcx,ind]=sort(funcx,'descend');             % max fitness is first.
 x=x(ind,:);                                   % sorting population from..
                                               % ..lowest to largest.
maxfx(1)=max(funcx);                           % maximum population
fitness.
meanfx(1)=mean(funcx);                         % mean population fitness.
```

```
keep=floor(Ps*nPop);                               % no. of survivors.

nM=ceil(Pc*(nPop-keep))/2;                         % no. of mating parents.
odds=1;
for i=2:keep,odds=[odds i*ones(1,i)];end
nodds=length(odds);
pik1=ceil(nodds*rand(1,nM)); % parent#1
pik2=ceil(nodds*rand(1,nM)); % parent#2
ma=odds(pik1);% index of mother
pa=odds(pik2);% index of father
% -----------------
% fid_ali=fopen('PGA_data(ii).txt','w');
% ----------------
% fid_ali = fopen(sprintf( 'PGA_Conv\PGAConv%i.txt',ii),'w' );  % where j
is your loop variable
% fid_ali = fopen(strcat([ 'PGA_Conv\PGAConv',num2str(ii),'.txt']),'wt');
while ige<Imax
ige=ige+1;
%--------------------------------------------------------------------------
%                              3. Crossover
%--------------------------------------------------------------------------
%
  for i=1:nM
    pa1=x(ma(i),:);
    pa2=x(pa(i),:);
    indx=2*(i-1)+1;                                % skipping index
    chroms2=[pa1;pa2];
    x(keep+indx:keep+indx+1,:)=crossover(chroms2,nvar);
  end
%--------------------------------------------------------------------------
%                              4. Mutation
%--------------------------------------------------------------------------
%
numt=ceil(Pm*nPop);
 for j=1:numt
    xn=randi([1 nPop]);
    x(xn,:)=mutuation(x(j,:),nvar);
 end
%--------------------------------------------------------------------------
%                              5. Permutation
%--------------------------------------------------------------------------
%
for np=1:nPop
    x(np,:)=permutuation(x(np,:),nvar);
end

for nk=1:nPop,funcx(nk,:)=feval(objfunc,x(nk,:)); end
[funcx,ind]=sort(funcx,'descend');
x=x(ind,:);
% disp('     ind            x            funcx')
% disp('------------------------------------------')
% disp([ind x funcx])
maxfx(ige)=max(funcx);
minfx(ige)=min(funcx);
meanfx(ige)=mean(funcx);
```

```
fbest=maxfx(ige);
if maxfx(ige)-minfx(ige)<eps,break;end
end
% -----------Declare the optimum solution of Xopt & fopt-----------------%
[fopt,ixb]=sort(fbest,'descend');
 xopt=x(ixb,:);
%-----------------------------------------------------------------------
%                           Displays the output
%-----------------------------------------------------------------------
%
day=clock;
disp(datestr(datenum(day(1),day(2),day(3),day(4),day(5),day(6)),0))
disp(['Optimized Function is ' 'CBLF'])
format short g
disp(['Popsize = ' num2str(nPop) ' mutrate = '...
num2str(Pm) ' Number of variables = ' num2str(nvar)])
disp(['      Best cost(fopt)= ' num2str(fopt)])
disp([' Best solution (xopt):'])
disp(['                       ' num2str(xopt)])
```

crossover.m

```
function chroms2=crossover(chroms2,nvar)
    rp1=randi([1 nvar]);    % 1st random point
    rp2=randi([1 nvar]);    % 2nd random point
    cr1=min(rp1,rp2);       % 1st crossover point
    cr2=max(rp1,rp2);       % 2nd crossover point
    tmp=chroms2(1,cr1:cr2); % switcher
    chroms2(1,cr1:cr2)=chroms2(2,cr1:cr2);
    chroms2(2,cr1:cr2)=tmp;
```

mutation.m

```
function chrom1= mutation(chrom1,nvar)
    mp1=randi([1 nvar]);
    mp2=randi([1 nvar]);
    tmp=chrom1(mp1);
    chrom1(mp1)=chrom1(mp2);
    chrom1(mp2)=tmp;
```

permutation.m

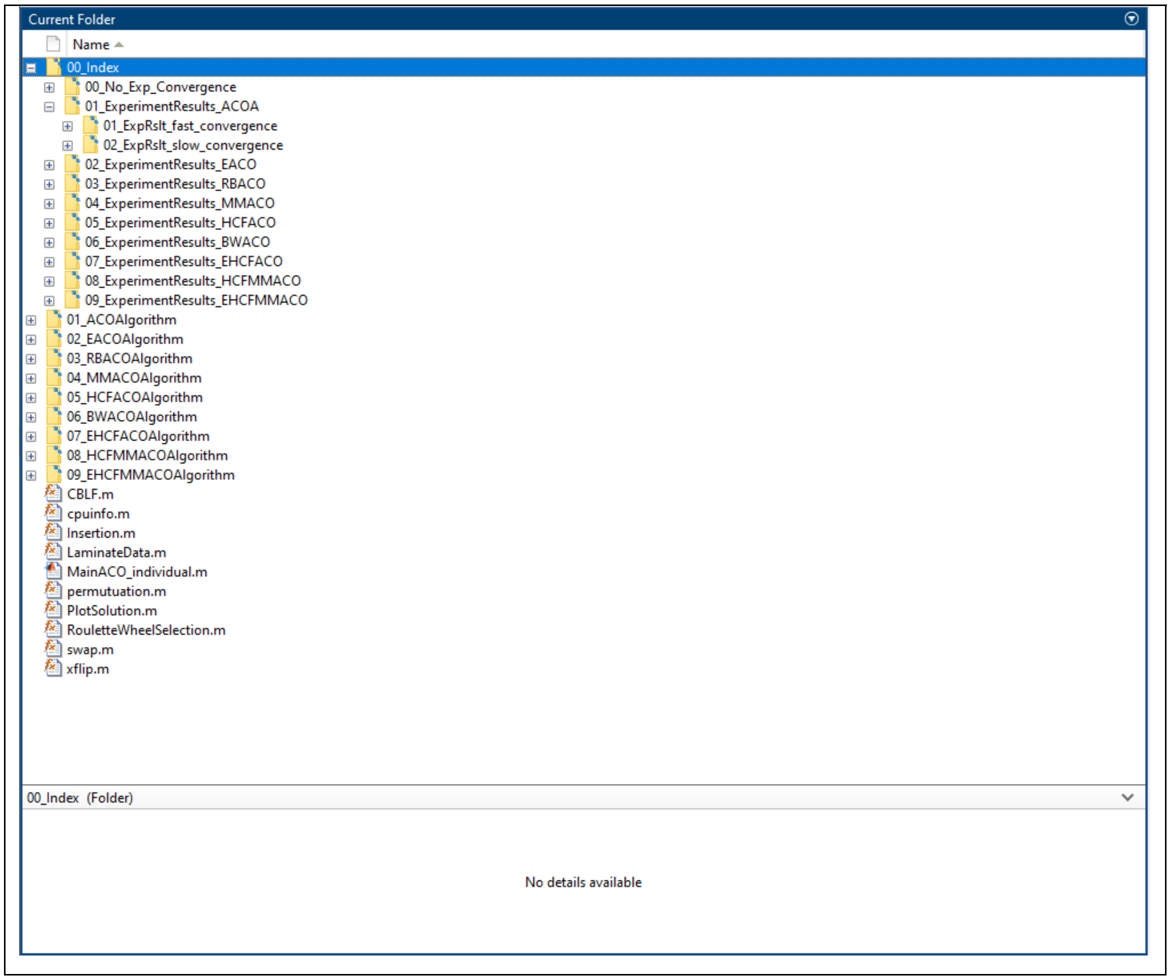
```
function pchrom1=permutation(pchrom1,nvar)
rp1=randi([1 nvar]);
rp2=randi([1 nvar]);
```

```
prp1=min(rp1,rp2);
prp2=max(rp1:rp2);
pchrom1(prp1:prp2)=pchrom1(flip(prp1:prp2));
```

APPENDIX III

MATLAB CODES FOR ANT COLONY OPTIMIZATION ALGORITHMS

Ant Colony Optimization Matlab Folder Structure



```
Current Folder
Name
00_Index
    00_No_Exp_Convergence
    01_ExperimentResults_ACOA
        01_ExpRslt_fast_convergence
        02_ExpRslt_slow_convergence
    02_ExperimentResults_EACO
    03_ExperimentResults_RBACO
    04_ExperimentResults_MMACO
    05_ExperimentResults_HCFACO
    06_ExperimentResults_BWACO
    07_ExperimentResults_EHCFACO
    08_ExperimentResults_HCFMMACO
    09_ExperimentResults_EHCFMMACO
01_ACOAlgorithm
02_EACOAlgorithm
03_RBACOAlgorithm
04_MMACOAlgorithm
05_HCFACOAlgorithm
06_BWACOAlgorithm
07_EHCFACOAlgorithm
08_HCFMMACOAlgorithm
09_EHCFMMACOAlgorithm
CBLF.m
cpuinfo.m
Insertion.m
LaminateData.m
MainACO_individual.m
permutuation.m
PlotSolution.m
RouletteWheelSelection.m
swap.m
xflip.m

00_Index  (Folder)

No details available
```

MainACO.m

```
%==========================================================================
%
%    File: MainACO.m
%  Author: Ali Ahmid
% Version: V2.0 March 2020
%       ?: This is the main ACO Algorithms file where the user can enter
%          his model information and select the different ACO algorithms
```

```
%=========================================================================
%
%% Adding working directories
addpath(genpath('00_Index'))
addpath(genpath('01_ACOAlgorithm'))
addpath(genpath('02_EACOAlgorithm'))
addpath(genpath('03_RBACOAlgorithm'))
addpath(genpath('04_MMACOAlgorithm'))
addpath(genpath('05_HCFACOAlgorithm'))
addpath(genpath('06_BWACOAlgorithm'))
addpath(genpath('07_EHCFACOAlgorithm'))
addpath(genpath('08_HCFMMACOAlgorithm'))
addpath(genpath('09_EHCFMMACOAlgorithm'))
%% Resetting the working space and declare Global variables
clear,clc
clear global
global Data ii MachineInformation iseed is maxIconv mk
%% Check operating system and CPU specification
MachineInformation=cpuinfo;
%% Intializiation of Data file
Data=LaminateData;
%% Selecting the Options file
% Options=Data.ACOptions;
Options =Data.MMACOptions;
% Options =Data.BWACOptions;
%% Converegence rate
ConvRate= struct('fast',10,'slow',56);
maxIconv=[ConvRate.fast,ConvRate.slow];
for mk=1:length(maxIconv)
%% The Expierments Execution
iseed=[301 2 50 75 111 200 167 225 11 25];
for is=1:length(iseed)
    rng(iseed(is));
Iexp=200;
% Select the ACO Variant
     for ii=1:Iexp,Solution(ii) = MMACO(Options);end
  %% Experiments Statistics
    ExperiementSolution.fopt        = [Solution.fopt];
    ExperiementSolution.xopt        = [Solution.xopt];
    ExperiementSolution.Maxfopt     = max(ExperiementSolution.fopt);
    ExperiementSolution.Minfopt     = min(ExperiementSolution.fopt);
    ExperiementSolution.Meanfopt    = mean(ExperiementSolution.fopt);
    ExperiementSolution.Medianfopt  = median(ExperiementSolution.fopt);
    ExperiementSolution.Stdfopt     = std(ExperiementSolution.fopt);
    ExperiementSolution.AvgPrice    = mean([Solution.ige]);
    ExperiementSolution.Relaibility = sum(ExperiementSolution.fopt>=
3973.01/1.001)/Iexp*100;
    ExperiementSolution.NormPrice   =
ExperiementSolution.AvgPrice/(ExperiementSolution.Relaibility(1)/100);
    ExperiementSolution.Quality     = (1-(3973.01-
ExperiementSolution.fopt)/3973.01)*100;%(1-
ExperiementSolution.Stdfopt/ExperiementSolution.Meanfopt)*(ExperiementSolu
tion.Meanfopt/ExperiementSolution.Maxfopt)*100;
    ExperiementSolution.AvgQuality  = sum(ExperiementSolution.Quality
)/Iexp;
```

```
    ExperiementSolution.Robustness  = ExperiementSolution.Meanfopt/
ExperiementSolution.Stdfopt;
    ExperiementSolution.PerRate     = sum( ExperiementSolution.fopt>=
ExperiementSolution.Maxfopt/1.001)/(ExperiementSolution.AvgPrice*Iexp);
    ExperiementSolution.AvgFDC      = sum([Solution.cFD])/Iexp;
    ExperiementSolution.AvgR        = sum([Solution.r])/Iexp;
    ExperiementSolution.ElapsedTime = sum([Solution.ElapsedTime])/Iexp;
    ExperiementSolution.maxTau      = [Solution.maxTauExp];
    ExperiementSolution.minTau      = [Solution.minTauExp];
     close Figure 1
%% Saving results into ExperimentResultsXXXACO.txt
if mk==1
%
fid=fopen(sprintf('00_Index/01_ExperimentResults_ACOA/01_ExpRslt_fast_conv
ergence/01_ExperimentResultsACOA%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/02_ExperimentResults_EACO/01_ExpRslt_fast_conv
ergence/02_ExperimentResultsEACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/03_ExperimentResults_RBACO/01_ExpRslt_fast_con
vergence/03_ExperimentResultsRBACO%i.txt',is),'w');

fid=fopen(sprintf('00_Index/04_ExperimentResults_MMACO/01_ExpRslt_fast_con
vergence/04_ExperimentResultsMMACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/05_ExperimentResults_HCFACO/01_ExpRslt_fast_co
nvergence/05_ExperimentResultsHCFACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/06_ExperimentResults_BWACO/01_ExpRslt_Fast_con
vergence/02_ExperimentResultsBWACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/07_ExperimentResults_EHCFACO/01_ExpRslt_fast_c
onvergence/07_ExperimentResultsEHCFACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/08_ExperimentResults_HCFMMACO/01_ExpRslt_fast_
convergence/08_ExperimentResultsHCFMMACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/09_ExperimentResults_EHCFMMACO/01_ExpRslt_fast
_convergence/09_ExperimentResultsEHCFMMACO%i.txt',is),'w');
else
%
fid=fopen(sprintf('00_Index/01_ExperimentResults_ACOA/02_ExpRslt_slow_conv
ergence/01_ExperimentResultsACOA%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/02_ExperimentResults_EACO/02_ExpRslt_slow_conv
ergence/02_ExperimentResultsEACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/03_ExperimentResults_RBACO/02_ExpRslt_slow_con
vergence/03_ExperimentResultsRBACO%i.txt',is),'w');

fid=fopen(sprintf('00_Index/04_ExperimentResults_MMACO/02_ExpRslt_slow_con
vergence/04_ExperimentResultsMMACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/05_ExperimentResults_HCFACO/02_ExpRslt_slow_co
nvergence/05_ExperimentResultsHCFACO%i.txt',is),'w');
```

```
%
fid=fopen(sprintf('00_Index/06_ExperimentResults_BWACO/02_ExpRslt_slow_con
vergence/06_ExperimentResultsBWACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/07_ExperimentResults_EHCFACO/02_ExpRslt_slow_c
onvergence/07_ExperimentResultsEHCFACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/08_ExperimentResults_HCFMMACO/02_ExpRslt_slow_
convergence/08_ExperimentResultsHCFMMACO%i.txt',is),'w');
%
fid=fopen(sprintf('00_Index/09_ExperimentResults_EHCFMMACO/02_ExpRslt_slow
_convergence/09_ExperimentResultsEHCFMMACO%i.txt',is),'w');
end


fprintf(fid,'%s\r\n','====================================================
=====================');
    fprintf(fid,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid,'%s\r\n','====================================================
=====================');
    fprintf(fid,'%s\r\n','Machine Information: ');
    fprintf(fid,'%s\n',' ');
    fprintf(fid,'%s\r\n',['               CPU Processor:'
num2str(MachineInformation.Name)] );
    fprintf(fid,'%s\r\n',['             CPU clock speed:'
MachineInformation.Clock] );
    fprintf(fid,'%s\r\n',['         CPU Cache size (L2):'
MachineInformation.Cache] );
    fprintf(fid,'%s\r\n',['Number of physical CPU cores:'
num2str(MachineInformation.NumProcessors)] );
    fprintf(fid,'%s\r\n',['       Operating System Type:'
MachineInformation.OSType] );
    fprintf(fid,'%s\r\n',['    Operating System Version:'
MachineInformation.OSVersion] );

fprintf(fid,'%s\r\n','====================================================
=====================');

%       fprintf(fid,'                          %s\n','ACO Varient: ACO');
%     fprintf(fid,'                          %s\n','ACO Varient: EACO');
%     fprintf(fid,'                          %s\n','ACO Varient: RBACO');
    fprintf(fid,'                          %s\n','ACO Varient: MMACO');
%     fprintf(fid,'                          %s\n','ACO Varient: HCFACO');
%     fprintf(fid,'                          %s\n','ACO Varient: BWACO');
%     fprintf(fid,'                          %s\n','ACO Varient: EHCFACO');
%     fprintf(fid,'                          %s\n','ACO Varient:
HCFMMACO');
%     fprintf(fid,'                          %s\n','ACO Varient:
EHCFMMACO');

    fprintf(fid,'%s\r\n',   ['    Global Optimal Solution = '
num2str(ExperiementSolution.Maxfopt)]);
```

```
    fprintf(fid,'%s\r\n',   ['   Minimum Optimal Solution = '
num2str(ExperiementSolution.Minfopt)]);
    fprintf(fid,'%s\r\n',   ['   Mean of Optimal Solution = '
num2str(ExperiementSolution.Meanfopt)]);
    fprintf(fid,'%s\r\n',   [' Median of Optimal Solution = '
num2str(ExperiementSolution.Meanfopt)]);
    fprintf(fid,'%s\r\n',   ['Solution Standrad Deviation = '
num2str(ExperiementSolution.Stdfopt)]);
    fprintf(fid,'%s\r\n',   ['              Average Price = '
num2str(ExperiementSolution.AvgPrice)]);
    fprintf(fid,'%s %s\r\n',['                Relability = '
num2str(ExperiementSolution.Relaibility)],'%');
    fprintf(fid,'%s\r\n',   ['           Normalized Price = '
num2str(ExperiementSolution.NormPrice)]);
    fprintf(fid,'%s %s\r\n',['   Average Solution Quality = '
num2str(ExperiementSolution.AvgQuality)],'%');
    fprintf(fid,'%s\r\n',   ['                Average FDC = '
num2str(ExperiementSolution.AvgFDC)]);
    fprintf(fid,'%s\r\n',   ['     Correlation Coefficent = '
num2str(ExperiementSolution.AvgR)]);
    fprintf(fid,'%s\r\n',   ['        Solution Robustness = '
num2str(ExperiementSolution.Robustness)]);
    fprintf(fid,'%s\r\n',   ['           Performance Rate = '
num2str(ExperiementSolution.PerRate)]);
    fprintf(fid,'%s %s\r\n',['       Average Elapsed Time = '
num2str(ExperiementSolution.ElapsedTime)],'sec');

fprintf(fid,'%s\r\n','====================================================
=====================');
    fprintf(fid,'%s %12s %12s %12s %12s\r\n','Expeirment
#','Foptimal','SolQuality','MaxExpTau ','MinExpTau');
    fprintf(fid,'%s %12s %12s%12s
%12s\r\n','============','========','==========','=========','=========');
    fprintf(fid,'     %-8.0f    %-8.3f      %-8.2f   %-8.5f      %-
8.5f\r\n',[(1:Iexp);ExperiementSolution.fopt;ExperiementSolution.Quality;E
xperiementSolution.maxTau;ExperiementSolution.minTau]);

fprintf(fid,'%s\r\n','====================================================
=====================');
fclose(fid);

rng('default')
end
end
```

ACOA.m

```
function Solution=ACOA(Options)
global Data ii iseed is maxIconv mk
tic
%% Ant Colony Optimization Algorithm (ACO) to maximize f(x)
%        Solution = ACO(Options)
%   Solution.xopt = Optimal Solution
%   Solution.fopt = Optimal solution
%          Data.f  = Objective Function
%       Data.nVar  = no. of design variables
%         Data.xd  = Discrete design variables values vector e.g[0 45 90]
%        Data.nAnt = no. of Ants
%     Options.Tau0 = Initial phormone
%     Options.zeta = Global updating scale factor
%     Options.rho  = Evaporation Rate
%     Options.Imax = Max no. of iterations
%%%------------------------------------------------------------------------
%
%% Check ACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)

DefaultOpt=struct('nAnt',25,'Imax',1000,'rho',0.07,'Tau0',0.004,'zeta',.03
);
  Options = DefaultOpt;
end
%% Assiging ACO Parameters value
 Imax= Options.Imax;
  rho= Options.rho;
 zeta= Options.zeta;
 Tau0=Options.Tau0;
 nAnts=Options.nAnt;
 %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create ACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'01_ACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/ACOAutput%i.txt',is,ise
ed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'01_ACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/ACOAutput%i.txt',is,isee
d(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'01_ACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/ACOAutput%i.txt',is,ise
ed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'01_ACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/ACOAutput%i.txt',is,isee
d(is),ii),'w' );
 end
```

```
    end
%%

fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
 fprintf(fid_ali1,'                         %s\n','ACO Varient: ACOA');
 fprintf(fid_ali1,'%s\n',['         Expeirement no.(Iexp) = ',
num2str(ii)]);
 fprintf(fid_ali1,'%s\n',['                         Seeds = ',
num2str(iseed(is))]);
 %% Main ACOA Loop
 while ige<Imax
    ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
     Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daimon actions
%       NA
%% Pheromone Updating
       % Phormen Evaporation (for all paths )
       Tau=(1-rho)*Tau;
       % Pheromone Depositing (for the best path )
```

```
            for i=1:Data.nVar
        Tau(i,Solution.xbest(1,i,ige))=
Tau(i,Solution.xbest(1,i,ige))+zeta*Solution.Segma(ige)*Solution.fworst(ig
e)/Solution.fbest(ige);
            end
              tempMaxTau(ige,:) = max(Tau);
              tempMinTau(ige,:) = min(Tau);
           Solution.maxTau(ige) = max(tempMaxTau(ige,:));
           Solution.minTau(ige) = min(tempMaxTau(ige,:));
        %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
            if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % ACOA Solution results
          [Solution.fopt, jj] = max(Solution.fbest(1:ige));
                Solution.xopt = Solution.xbest(1,:,jj);
           Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                 Solution.ige = ige;
                       [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
            Solution.FrstGopt = mm(1);
            Solution.Minfbest = min(Solution.fbest(:));
           Solution.Meanfbest = mean(Solution.fbest(:));
            Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                 Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                   Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
             Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i)  = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.maxTau(1:ige));
        Solution.minTauExp=min(Solution.minTau(1:ige));
%% Write ACOA results Output File
fprintf(fid_ali1,'%s\r\n',['           Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                   no. of Ants = '
num2str(nAnts)]);
```

```
fprintf(fid_ali1,'%s\r\n',['             Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['         Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['         no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['         Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['         Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['           SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['         Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                   Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','============================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','============================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.maxTau(:)';Solution.minTau(:)']);
fprintf(fid_ali1,'%s\r\n','============================================
=================================');
fclose(fid_ali1);
% end
```

EACO.m

```
function Solution=EACO(Options)
global Data ii iseed is maxIconv mk
tic
%% Elitisem Ant Colony Optimization Algorithm (EACO) to maximize f(x)
%         Solution = EACO(Options)
%    Solution.xopt = Optimal Solution
%    Solution.fopt = Optimal solution
%          Data.f  = Objective Function
%       Data.nVar  = no. of design variables
%         Data.xd  = Discrete design variables values vector e.g[0 45 90]
%        Data.nAnt = no. of Ants
%    Options.Tau0 = Initial phormone
%    Options.zeta = Global updating scale factor
%    Options.rho  = Evaporation Rate
%    Options.Imax = Max no. of iterations
%%%-----------------------------------------------------------------------
%
% Check ACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)

DefaultOpt=struct('nAnt',25,'Imax',1000,'rho',0.07,'Tau0',0.004,'zeta',.03
);
  Options = DefaultOpt;
end
%% Assiging ACO Parameters value
 Imax= Options.Imax;
  rho= Options.rho;
 zeta= Options.zeta;
 Tau0=Options.Tau0;
 nAnts=Options.nAnt;
 %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create ACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'02_EACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/EACOutput%i.txt',is,is
eed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'02_EACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/EACOutput%i.txt',is,ise
ed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'02_EACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/EACOutput%i.txt',is,is
eed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'02_EACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/EACOutput%i.txt',is,ise
ed(is),ii),'w' );
 end
    end
%%
```

```
fprintf(fid_ali1,'%s\r\n','==============================================
=================================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','==============================================
=================================');
 fprintf(fid_ali1,'                         %s\n','ACO Varient: EACO');
 fprintf(fid_ali1,'%s\n',['         Expeirement no.(Iexp) = ',
num2str(ii)]);
 fprintf(fid_ali1,'%s\n',['                         Seeds = ',
num2str(iseed(is))]);
 %% Main EACO Loop
 while ige<Imax
    ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
     Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daimon actions
%       NA
%% Pheromone Updating
       % Phormen Evaporation (for all paths )
          Tau=(1-rho)*Tau;
       % Pheromone Depositing (for the best path )
           for i=1:Data.nVar
       Tau(i,Solution.xbest(1,i,ige))=
Tau(i,Solution.xbest(1,i,ige))+zeta*Solution.Segma(ige)*Solution.fworst(ig
e)/Solution.fbest(ige)+Solution.Segma(ige)*Solution.fbest(ige)/sum(Solutio
n.fx);
           end
             tempMaxTau(ige,:) = max(Tau);
             tempMinTau(ige,:) = min(Tau);
          Solution.maxTau(ige) = max(tempMaxTau(ige,:));
          Solution.minTau(ige) = min(tempMaxTau(ige,:));

%% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % EACO Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
               Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                Solution.ige = ige;
                      [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                  Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.maxTau(1:ige));
        Solution.minTauExp=min(Solution.minTau(1:ige));
%% Write EACO results Output File
fprintf(fid_ali1,'%s\r\n',['           Number of variables = '
num2str(Data.nVar)]);
```

```
fprintf(fid_ali1,'%s\r\n',['                       no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['             Maximum CBLF(fopt)  = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['         Optimal Solution(xopt)  = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima   = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['         no. of Iterations(ige)  = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['         Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['         Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['           SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['         Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                   Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','==================================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','==================================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.maxTau(:)';Solution.minTau(:)']);
fprintf(fid_ali1,'%s\r\n','==================================================
=================================');
fclose(fid_ali1);
```

RBACO.m

```
function Solution=RBACO(Options)
global Data ii iseed is maxIconv mk
tic

%% Rank Based Ant Colony Optimization Algorithm (RBACO) to maximize f(x)
%         Solution = RBACO(Options)
%   Solution.xopt = Optimal Solution
%   Solution.fopt = Optimal solution
%         Data.f  = Objective Function
%      Data.nVar  = no. of design variables
%        Data.xd  = Discrete design variables values vector e.g[0 45 90]
%       Data.nAnt = no. of Ants
%    Options.Tau0 = Initial phormone
%    Options.rho  = Evaporation Rate
%    Options.Imax = Max no. of iterations
%%%-----------------------------------------------------------------------
%
%% Check ACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)

DefaultOpt=struct('nAnt',25,'Imax',1000,'rho',0.07,'Tau0',0.004,'zeta',.03
);
  Options = DefaultOpt;
end
%% Assiging ACO Parameters value
 Imax= Options.Imax;
  rho= Options.rho;
 zeta= Options.zeta;
 Tau0=Options.Tau0;
 nAnts=Options.nAnt;
 %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create ACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'03_RBACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/RBACOutput%i.txt',is,
iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'03_RBACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/RBACOutput%i.txt',is,i
seed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'03_RBACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/RBACOutput%i.txt',is,
iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'03_RBACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/RBACOutput%i.txt',is,i
seed(is),ii),'w' );
```

```
 end
    end
%%

fprintf(fid_ali1,'%s\r\n','==================================================
==================================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','==================================================
==================================');
 fprintf(fid_ali1,'                         %s\n','ACO Varient: RBACO');
 fprintf(fid_ali1,'%s\n',['         Expeirement no.(Iexp) = ',
num2str(ii)]);
 fprintf(fid_ali1,'%s\n',['                         Seeds = ',
num2str(iseed(is))]);
 %% Main ACOA Loop
 while ige<Imax
    ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
     Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
                         wRank = Solution.Segma(ige)-1;
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daimon actions
%       NA
%% Pheromone Updating
       % Pheromone Evaporation (for all paths )
```

```
          Tau=(1-rho)*Tau;
          if wRank==0
              Solution.rank=Solution.fbest(ige)/sum(Solution.fx);
          else
       % Pheromone Depositing (for the best path )
       for mu=1:wRank
            Solution.rank(mu)=(Solution.Segma(ige)-
mu)*Solution.fx(mu)/sum(Solution.fx);
       end
          end
          d=sum(Solution.rank)
       for i=1:Data.nVar
         Tau(i,Solution.xbest(1,i,ige))= Tau(i,Solution.xbest(1,i,ige))+
sum(Solution.rank)+zeta*Solution.Segma(ige)*Solution.fworst(ige)/Solution.
fbest(ige);
       end

             tempMaxTau(ige,:) = max(Tau);
             tempMinTau(ige,:) = min(Tau);
          Solution.maxTau(ige) = max(tempMaxTau(ige,:));
          Solution.minTau(ige) = min(tempMaxTau(ige,:));
       %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % RBACO Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
              Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
               Solution.ige = ige;
                     [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
               Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                 Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
```

```
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.maxTau(1:ige));
        Solution.minTauExp=min(Solution.minTau(1:ige));
%% Write RBACO results Output File
fprintf(fid_ali1,'%s\r\n',['           Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                   no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['            Maximum CBLF(fopt)  = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['        Optimal Solution(xopt)  = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima   = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['        no. of Iterations(ige)  = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['        Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['        Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['          SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['        Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                  Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.maxTau(:)';Solution.minTau(:)']);
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fclose(fid_ali1);
```

MMACO.m

```
function Solution=MMACO(Options)
```

```
global Data ii iseed is maxIconv mk
tic
%% Max-Min Ant Colony Optimization Algorithm (MMACO) to maximize f(x)
%         Solution = MMACO(Options)
%   Solution.xopt = Optimal Solution
%   Solution.fopt = Optimal solution
%          Data.f   = Objective Function
%       Data.nVar  = no. of design variables
%         Data.xd  = Discrete design variables values vector e.g[0 45 90]
%        Data.nAnt = no. of Ants
%    Options.Tau0 = Initial phormone
%   Options.pbest = probabilitiy of the best solution
%    Options.rho  = Evaporation Rate
%    Options.Imax = Max no. of iterations
%%%----------------------------------------------------------------------
%
%% Check MMACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)
  DefaultOpt
=struct('nAnt',25,'Imax',1000,'rho',.98,'Tau0',1,'pbest',.05,'zeta',0.03);
     Options = DefaultOpt;
end
%% Assiging MMACO Parameters value
 Imax=Options.Imax;
 rho=Options.rho;
 Tau0=Options.Tau0;
 nAnts=Options.nAnt;
 pbest=Options.pbest;
 zeta=Options.zeta;
  %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create MMACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'04_MMACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/MMACOutput%i.txt',is,
iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'04_MMACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/MMACOutput%i.txt',is,i
seed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'04_MMACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/MMACOutput%i.txt',is,
iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'04_MMACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/MMACOutput%i.txt',is,i
seed(is),ii),'w' );
 end
   end
 %% Create MMACOutput.txt File

fprintf(fid_ali1,'%s\r\n','===============================================
==========================');
 fprintf(fid_ali1,'%s
%s\n','User:Ali Ahmid',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','===============================================
==========================');
 fprintf(fid_ali1,'                         %s\n','ACO Varient: MMACO');
 fprintf(fid_ali1,'%s\n',[' Expeirement no.(Iexp)= ', num2str(ii)]);

%% % Main ACOA Loop
while ige<Imax
ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
% Solution Evaluation
Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daemon Actions
                    %NA
%% Pheromone Updating

% Pheromone Evaporation (for all paths )
         Tau=(1-rho)*Tau;
       % Determine Tmax and Tmin
         Solution.TauMax(ige)=(1-rho)^-
1*Solution.fworst(ige)/Solution.fbest(ige);
         Solution.TauMin(ige)=Solution.TauMax(ige)*(1-
nthroot(pbest,nAnts))/((nAnts/2-1)*nthroot(pbest,nAnts));
       % Pheromone Depositing (for the best path )
         for i=1:Data.nVar

Tau(i,Solution.xbest(:,i,ige))=Tau(i,Solution.xbest(1,i,ige))+zeta*Solutio
n.Segma(ige)*Solution.fworst(ige)/Solution.fbest(ige);
            if Tau(i,Solution.xbest(1,i,ige))> Solution.TauMax(ige)
            Tau(i,Solution.xbest(1,i,ige))=Solution.TauMax(ige);
            elseif Tau(i,Solution.xbest(1,i,ige))< Solution.TauMin(ige)
             Tau(i,Solution.xbest(1,i,ige))=Solution.TauMin(ige);
            else
            end
         end
       %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % ACOA Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
               Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                Solution.ige = ige;
                      [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                  Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
```

```
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.TauMax(1:ige));
        Solution.minTauExp=min(Solution.TauMin(1:ige));

%% Write MMACO results Output File
fprintf(fid_ali1,'%s\r\n',['          Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                  no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['           Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['       Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['       no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['       Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['       Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['         SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['       Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                 Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','================================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','================================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.TauMax(:)';Solution.TauMin(:)']);
fprintf(fid_ali1,'%s\r\n','================================================
=================================');
fclose(fid_ali1);
```

HCFACO.m

```
function Solution=HCFACO(Options)
global Data ii iseed is maxIconv mk
tic
%% Hyper-Cube Ant Colony Optimization (EHCFACO) to maximize f(x)
%        Solution = HCFACO(Options)
%   Solution.xopt = Optimal Solution
%   Solution.fopt = Optimal solution
%          Data.f  = Objective Function
%       Data.nVar  = no. of design variables
%         Data.xd  = Discrete design variables values vector e.g[0 45 90]
%        Data.nAnt = no. of Ants
%    Options.Tau0 = Initial phormone
%    Options.rho  = Evaporation Rate
%    Options.Imax = Max no. of iterations
%%%-----------------------------------------------------------------------
%
%% Check HCFACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)

DefaultOpt=struct('nAnt',25,'Imax',1000,'rho',0.07,'Tau0',0.004,'zeta',.03
);
  Options = DefaultOpt;
end
%% Assiging HCFACO Parameters value
 Imax= Options.Imax;
  rho= Options.rho;
  Tau0=Options.Tau0;
 nAnts=Options.nAnt;
 %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create HCFACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'05_HCFACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/HCFACOutput%i.txt',i
s,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'05_HCFACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/HCFACOutput%i.txt',is
,iseed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'05_HCFACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/HCFACOutput%i.txt',i
s,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'05_HCFACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/HCFACOutput%i.txt',is
,iseed(is),ii),'w' );
 end
    end
%%

fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
 fprintf(fid_ali1,'                         %s\n','ACO Varient: HCFACO');
 fprintf(fid_ali1,'%s\n',['         Expeirement no.(Iexp) = ',
num2str(ii)]);
 fprintf(fid_ali1,'%s\n',['                         Seeds = ',
num2str(iseed(is))]);
 %% Main HCFACO Loop
 while ige<Imax
    ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
     Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
```

```
        x = Solution.xbest(1,:,ige);
        y = Data.nVar;
        PlotSolution(x,y)
    %% Daimon actions
%       NA
%% Pheromone Updating
%         gama_s=Solution.fbest(ige)/sum(Solution.fx);
%         S(1,:,ige)=ones(1,Data.nVar);
        % Pheromone Depositing (for the best path )
       for i=1:Data.nVar

Tau(i,Solution.xbest(1,i,ige))=Tau(i,Solution.xbest(1,i,ige))+rho*Solution
.Segma(ige)*Solution.fbest(ige)/sum(Solution.fx);
%rho*Solution.Segma(ige)*(gama_s*S(1,i,ige)-
Tau(i,Solution.xbest(1,i,ige)));%rho*zeta*Solution.Segma(ige)*Solution.fbe
st(ige)/sum(Solution.fx);%
       end
        tempMaxTau(ige,:) = max(Tau);
        tempMinTau(ige,:) = min(Tau);
     Solution.maxTau(ige) = max(tempMaxTau(ige,:));
     Solution.minTau(ige) = min(tempMaxTau(ige,:));
        %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % HCFACO Solution results
          [Solution.fopt, jj] = max(Solution.fbest(1:ige));
                Solution.xopt = Solution.xbest(1,:,jj);
           Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                 Solution.ige = ige;
                       [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                 Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                   Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
```

```
              else
                Solution.LinConvRate(i) = 0;
                break;
              end
          end
          Solution.maxTauExp=max(Solution.maxTau(1:ige));
          Solution.minTauExp=min(Solution.minTau(1:ige));
%% Write HCFACO results Output File
fprintf(fid_ali1,'%s\r\n',['           Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                   no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['            Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['        Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['        no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['        Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['        Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['          SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['        Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                  Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','==============================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','==============================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.maxTau(:)';Solution.minTau(:)']);
fprintf(fid_ali1,'%s\r\n','==============================================
=================================');
fclose(fid_ali1);
```

BWACO.m

```
function Solution=BWACO(Options)
global Data ii iseed is maxIconv mk
tic
%% Best-Worst Ant Colony Optimization Algorithm (MMACO) to maximize f(x)
%         Solution = BWACO(Options)
%    Solution.xopt = Optimal Solution
%    Solution.fopt = Optimal solution
%          Data.f  = Objective Function
%       Data.nVar  = no. of design variables
%         Data.xd  = Discrete design variables values vector e.g[0 45 90]
%        Data.nAnt = no. of Ants
%     Options.Tau0 = Initial phormone
%    Options.pbest = probabilitiy of the best solution
%     Options.rho  = Evaporation Rate
%     Options.Imax = Max no. of iterations
%%%----------------------------------------------------------------------
%
%% Check ACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)
  DefaultOpt
=struct('nAnt',25,'Imax',1000,'rho',0.1,'Tau0',1,'pbest',.05,'lamda',0.6,'
zeta',.03);
     Options = DefaultOpt;
end
%% Assiging ACO Parameters value
 Imax=Options.Imax;
  rho=Options.rho;
  Tau0=Options.Tau0;
 nAnts=Options.nAnt;
 zeta=Options.zeta;
 pbest=Options.pbest;
 %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create BWACOOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'06_BWACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/BWACOutput%i.txt',is,
iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'06_BWACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/BWACOutput%i.txt',is,i
seed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'06_BWACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/BWACOutput%i.txt',is,
iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'06_BWACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/BWACOutput%i.txt',is,i
seed(is),ii),'w' );
 end
   end
 %% Create BWACOutput.txt File

fprintf(fid_ali1,'%s\r\n','==============================================
==========================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','==============================================
==========================');
 fprintf(fid_ali1,'                          %s\n','ACO Varient: BWACO');
 fprintf(fid_ali1,'%s\n',[' Expeirement no.(Iexp)= ', num2str(ii)]);

%% % Main BWACO Loop
while ige<Imax
ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
% Solution Evaluation
Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daemon Actions
                    %NA
%% Pheromone Updating
       % Phormen Evaporation (for all paths )
       llamda=.6;
        % Pheromone Evaporation (for all paths )
         Tau=(1-rho)*Tau;
       % Determine Tmax and Tmin
         Solution.TauMax(ige)=(1-rho)^-
1*zeta*Solution.Segma(ige)*Solution.fworst(ige)/Solution.fbest(ige);
         Solution.TauMin(ige)=Solution.TauMax(ige)*(1-
nthroot(pbest,nAnts))/((nAnts/2-1)*nthroot(pbest,nAnts));
       % Pheromone Depositing (for the best path )
         for i=1:Data.nVar

Tau(i,Solution.xbest(:,i,ige))=Tau(i,Solution.xbest(1,i,ige))+zeta*Solutio
n.Segma(ige)*Solution.fworst(ige)/Solution.fbest(ige);
            if Tau(i,Solution.xbest(1,i,ige))> Solution.TauMax(ige)
            Tau(i,Solution.xbest(1,i,ige))=Solution.TauMax(ige);
            elseif Tau(i,Solution.xbest(1,i,ige))< Solution.TauMin(ige)
             Tau(i,Solution.xbest(1,i,ige))=Solution.TauMin(ige);
            else
            end
         end

          for i=1:Data.nVar
          Tau(i,Solution.xbest(:,i,ige))=Tau(i,Solution.xbest(1,i,ige))-
llamda*Solution.fworst(ige)/sum(Solution.fx);
          end
          tempMaxTau(ige,:) = max(Tau);
             tempMinTau(ige,:) = min(Tau);
          Solution.maxTau(ige) = max(tempMaxTau(ige,:));
          Solution.minTau(ige) = min(tempMaxTau(ige,:));
%% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % BWACO Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
               Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                Solution.ige = ige;
                      [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
```

```
   end
                Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                  Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.TauMax(1:ige));
        Solution.minTauExp=min(Solution.TauMin(1:ige));

%% Write EHCFMMACO results Output File
fprintf(fid_ali1,'%s\r\n',['          Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                  no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['           Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['       Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['  Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['       no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['       Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['       Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['         SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['       Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                 Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','==================================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','==================================================
=================================');
```

```
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.maxTau(:)';Solution.minTau(:)']);
fprintf(fid_ali1,'%s\r\n','=============================================
=================================');
fclose(fid_ali1);
```

EHCFACO.m

```
function Solution=EHCFACO(Options)
global Data ii iseed is maxIconv mk
tic
%% Enhanced Hyper-Cube Ant Colony Optimization (EHCFACO) to maximize f(x)
%         Solution = EHCFACO(Options)
%    Solution.xopt = Optimal Solution
%    Solution.fopt = Optimal solution
%          Data.f   = Objective Function
%       Data.nVar   = no. of design variables
%         Data.xd   = Discrete design variables values vector e.g[0 45 90]
%        Data.nAnt = no. of Ants
%     Options.Tau0 = Initial phormone
%     Options.rho  = Evaporation Rate
%     Options.Imax = Max no. of iterations
%%%-----------------------------------------------------------------------
%
%% Check EHCFACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)
  DefaultOpt=struct('nAnt',25,'Imax',1000,'rho',0.1,'Tau0',0.004);
  Options = DefaultOpt;
end
%% Assiging ACO Parameters value
  Imax = Options.Imax;
   rho = Options.rho;
  Tau0 = Options.Tau0;
 nAnts = Options.nAnt;
%% % Intialization
nNodes = length(Data.xd);
   Tau = Tau0*ones(Data.nVar,nNodes);
   ige = 0;
 %% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create EHCFACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'07_EHCFACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/EHCFACOutput%i.txt'
,is,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'07_EHCFACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/EHCFACOOutput%i.txt',
is,iseed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'07_EHCFACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/EHCFACOOutput%i.txt'
,is,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'07_EHCFACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/EHCFACOOutput%i.txt',
is,iseed(is),ii),'w' );
 end
    end

fprintf(fid_ali1,'%s\r\n','==============================================
==========================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','==============================================
==========================');
 fprintf(fid_ali1,'                         %s\n','ACO Varient: EHCFACO');
 fprintf(fid_ali1,'%s\n',['         Expeirement no.(Iexp) = ',
num2str(ii)]);
 fprintf(fid_ali1,'%s\n',['                         Seeds = ',
num2str(iseed(is))]);
 %% Main EHCFACO Loop
 while ige<Imax
    ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
     Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
```

```
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daimon actions
%       Insertion

xtemp=Insertion(Solution.xbest(1,:,ige),randi(Data.nVar),randi(Data.nVar))
;
      ftemp=feval(Data.f,xtemp);
      if
ftemp>Solution.fbest(ige),Solution.fbest(ige)=ftemp;Solution.xbest(1,:,ige
)=xtemp;end

%       Node flip (equavelent to mutuation)
      xtemp=xflip(Solution.xbest(1,:,ige),randi(Data.nVar));
      ftemp=feval(Data.f,xtemp);
      if
ftemp>Solution.fbest(ige),Solution.fbest(ige)=ftemp;Solution.xbest(1,:,ige
)=xtemp;end
      %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Pheromone Updating
       % Pheromone Depositing (for the best path )
      for i=1:Data.nVar

Tau(i,Solution.xbest(1,i,ige))=Tau(i,Solution.xbest(1,i,ige))+rho*Solution
.Segma(ige)*Solution.fbest(ige)/sum(Solution.fx);%(gama_s*S(1,i,ige)-
      end
       tempMaxTau(ige,:) = max(Tau);
       tempMinTau(ige,:) = min(Tau);
    Solution.maxTau(ige) = max(tempMaxTau(ige,:));
    Solution.minTau(ige) = min(tempMaxTau(ige,:));
       %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % HCFACO Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
               Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                Solution.ige = ige;
                      [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                  Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.maxTau(1:ige));
        Solution.minTauExp=min(Solution.minTau(1:ige));
%% Write EHCFACO results Output File
fprintf(fid_ali1,'%s\r\n',['          Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                  no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['           Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['       Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['       no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['       Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['       Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['         SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['       Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                 Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','==============================================
================================');
```

```
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','==============================================
==================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.maxTau(:)';Solution.minTau(:)']);
fprintf(fid_ali1,'%s\r\n','==============================================
==================================');
fclose(fid_ali1);
```

HCFMMACO.m

```
function Solution=HCFMMACO(Options)
global Data ii iseed is maxIconv mk
tic
%% Hyper-Cube Framework Max-Min Ant Colony Optimization Algorithm
(HCFMMACO)
%  to maximize f(x)
%--------------------------------------------------------------------------
%
%        Solution = HCFMMACO(Options)
%   Solution.xopt = Optimal Solution
%   Solution.fopt = Optimal solution
%          Data.f  = Objective Function
%      Data.nVar  = no. of design variables
%        Data.xd  = Discrete design variables values vector e.g[0 45 90]
%       Data.nAnt = no. of Ants
%    Options.Tau0 = Initial phormone
%   Options.pbest = probabilitiy of the best solution
%    Options.rho  = Evaporation Rate
%    Options.Imax = Max no. of iterations
%%%-----------------------------------------------------------------------
%
%% Check HCFMMACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)
  DefaultOpt
=struct('nAnt',25,'Imax',1000,'rho',.98,'Tau0',1,'pbest',.05,'zeta',0.03);
     Options = DefaultOpt;
end
%% Assiging HVFMMACO Parameters value
 Imax=Options.Imax;
 rho=Options.rho;
 Tau0=Options.Tau0;
 nAnts=Options.nAnt;
%  pbest=Options.pbest;
 zeta=Options.zeta;
  %% % Intialization
```

```
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create ACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'08_HCFMMACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/HCFMMACOutput%i.tx
t',is,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'08_HCFMMACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/HCFMMACOutput%i.txt
',is,iseed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'08_HCFMMACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/HCFMMACOutput%i.tx
t',is,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'08_HCFMMACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/HCFMMACOutput%i.txt
',is,iseed(is),ii),'w' );
 end
   end
 %% Create HCFMMACOutput.txt File

fprintf(fid_ali1,'%s\r\n','==============================================
==========================');
 fprintf(fid_ali1,'%s
%s\n','User:__________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','==============================================
==========================');
 fprintf(fid_ali1,'                        %s\n','ACO Varient:
HCFMMACO');
 fprintf(fid_ali1,'%s\n',[' Expeirement no.(Iexp)= ', num2str(ii)]);

%% % Main HCFMMACOA Loop
while ige<Imax
ige=ige+1;
for k=1:nAnts
% Construct Solution
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
% Solution Evaluation
Solution.fx(k)=feval(Data.f,Solution.x(k,:));
```

```
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daemon Actions
                    %NA
%% Pheromone Updating

% Pheromone Evaporation (for all paths )
         Tau=(1-rho)*Tau;
       % Determine Tmax and Tmin
         Solution.TauMax(ige)=1;
         Solution.TauMin(ige)=0;
       % Pheromone Depositing (for the best path )
         for i=1:Data.nVar

Tau(i,Solution.xbest(:,i,ige))=Tau(i,Solution.xbest(1,i,ige))+zeta*Solutio
n.Segma(ige)*Solution.fworst(ige)/Solution.fbest(ige);
            if Tau(i,Solution.xbest(1,i,ige))> Solution.TauMax(ige)
            Tau(i,Solution.xbest(1,i,ige))=Solution.TauMax(ige);
            elseif Tau(i,Solution.xbest(1,i,ige))< Solution.TauMin(ige)
             Tau(i,Solution.xbest(1,i,ige))=Solution.TauMin(ige);
            else
            end
         end
       %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % HCFMMACO Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
               Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
```

```
                Solution.ige = ige;
                      [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                  Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.TauMax(1:ige));
        Solution.minTauExp=min(Solution.TauMin(1:ige));

%% Write HCFMMACO results Output File
fprintf(fid_ali1,'%s\r\n',['           Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                   no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['            Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['        Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['        no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['        Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['        Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['          SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
```

```
fprintf(fid_ali1,'%s\r\n',['        Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                   Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.TauMax(:)';Solution.TauMin(:)']);
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fclose(fid_ali1);
```

EHCFMMACO.m

```
function Solution=EHCFMMACO(Options)
global Data ii iseed is maxIconv mk
tic
%% Enhanced Hyper-Cube FramwWork Max-Min Ant Colony Optimization Algorithm
%  (EHCFMMACO) to maximize f(x)
%-----------------------------------------------------------------------
%
%        Solution = EHCFMMACO(Options)
%   Solution.xopt = Optimal Solution
%   Solution.fopt = Optimal solution
%         Data.f  = Objective Function
%      Data.nVar  = no. of design variables
%        Data.xd  = Discrete design variables values vector e.g[0 45 90]
%       Data.nAnt = no. of Ants
%    Options.Tau0 = Initial phormone
%   Options.pbest = probabilitiy of the best solution
%    Options.rho  = Evaporation Rate
%    Options.Imax = Max no. of iterations
%%%---------------------------------------------------------------------
%
%% Check ACO Options
if ~isempty(Options) && ~isa(Options,'struct')
         error('Options must be a valid structure.');
elseif isempty(Options)
  DefaultOpt
=struct('nAnt',25,'Imax',1000,'rho',.1,'Tau0',0.004,'pbest',.05);
     Options = DefaultOpt;
end
%% Assiging HVFMMACO Parameters value
```

```
 Imax=Options.Imax;
 rho=Options.rho;
 Tau0=Options.Tau0;
 nAnts=Options.nAnt;
%  pbest=Options.pbest;
 zeta=Options.zeta;
  %% % Intialization
nNodes=length(Data.xd);
Tau=Tau0*ones(Data.nVar,nNodes);
Solution.fx=[];
Solution.x=[];
ige=0;
%% Check the convergence rate ConvRate
    if mk == 1
        mxIconv=maxIconv(1);
 %% Create EHCFMMACOutput.txt File
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'09_EHCFMMACOAlgorithm/01_fast_convergence/0%i_Seeds(%i)/EHCFMMACOutput%i.
txt',is,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'09_EHCFMMACOAlgorithm/01_fast_convergence/%i_Seeds(%i)/EHCFMMACOutput%i.t
xt',is,iseed(is),ii),'w' );
 end
    elseif mk==2
        mxIconv=maxIconv(2);
 if is < length(iseed)
 fid_ali1 = fopen( sprintf(
'09_EHCFMMACOAlgorithm/02_slow_convergence/0%i_Seeds(%i)/EHCFMMACOutput%i.
txt',is,iseed(is),ii),'w' );
 else
 fid_ali1 = fopen( sprintf(
'09_EHCFMMACOAlgorithm/02_slow_convergence/%i_Seeds(%i)/EHCFMMACOutput%i.t
xt',is,iseed(is),ii),'w' );
 end
   end
 %% Create MMACOutput.txt File

fprintf(fid_ali1,'%s\r\n','===============================================
==========================');
 fprintf(fid_ali1,'%s
%s\n','User:_________',num2str(datestr(clock)));

fprintf(fid_ali1,'%s\r\n','===============================================
==========================');
 fprintf(fid_ali1,'                        %s\n','ACO Varient:
EHCFMMACO');
 fprintf(fid_ali1,'%s\n',[' Expeirement no.(Iexp)= ', num2str(ii)]);

%% % Main EHCFMMACO Loop
while ige<Imax
ige=ige+1;
for k=1:nAnts
% Construct Solution
```

```
    for j=1:Data.nVar
        P=Tau(j,:);
        P=P/sum(P);
        Solution.x(k,j)=RouletteWheelSelection(P);
    end
% Solution Evaluation
Solution.fx(k)=feval(Data.f,Solution.x(k,:));
end
%% % Tour Statistics
             [Solution.fx,ind] = sort(Solution.fx,'descend');
               Solution.x(:,:) = Solution.x(ind,:);
           Solution.fbest(ige) = Solution.fx(1);
       Solution.xbest(1,:,ige) = Solution.x(1,:);
    % Check for current fbest
               [fbestCheck,kk] = max(Solution.fbest(1:ige));
                    xbestcheck = Solution.xbest(1,:,kk);
       if Solution.fbest(ige) <= fbestCheck
           Solution.fbest(ige) = fbestCheck;
       Solution.xbest(1,:,ige) = xbestcheck;
       end
          Solution.fworst(ige) = Solution.fx(end);
      Solution.xworst(1,:,ige) = Solution.x(end,:);
       Solution.fmean(1,:,ige) = mean(Solution.fx);
           Solution.Segma(ige) = sum(Solution.fx==Solution.fx(1));
Solution.BxWxLocalDis(1,:,ige) = Data.nVar-sum(Solution.xbest(1,:,ige)...
     == Solution.xworst(1,:,ige));
  %% Plotting results
       x = Solution.xbest(1,:,ige);
       y = Data.nVar;
       PlotSolution(x,y)
%% Daemon Actions
%       Insertion

xtemp=Insertion(Solution.xbest(1,:,ige),randi(Data.nVar),randi(Data.nVar))
;
      ftemp=feval(Data.f,xtemp);
      if
ftemp>Solution.fbest(ige),Solution.fbest(ige)=ftemp;Solution.xbest(1,:,ige
)=xtemp;end

%       Node flip (equavelent to mutuation)
      xtemp=xflip(Solution.xbest(1,:,ige),randi(Data.nVar));
      ftemp=feval(Data.f,xtemp);
      if
ftemp>Solution.fbest(ige),Solution.fbest(ige)=ftemp;Solution.xbest(1,:,ige
)=xtemp;end

%% Pheromone Updating

% Pheromone Evaporation (for all paths )
         Tau=(1-rho)*Tau;
       % Determine Tmax and Tmin
         Solution.TauMax(ige)=1;
         Solution.TauMin(ige)=0;
       % Pheromone Depositing (for the best path )
```

```
         for i=1:Data.nVar

Tau(i,Solution.xbest(:,i,ige))=Tau(i,Solution.xbest(1,i,ige))+zeta*Solutio
n.Segma(ige)*Solution.fworst(ige)/Solution.fbest(ige);
            if Tau(i,Solution.xbest(1,i,ige))> Solution.TauMax(ige)
            Tau(i,Solution.xbest(1,i,ige))=Solution.TauMax(ige);
            elseif Tau(i,Solution.xbest(1,i,ige))< Solution.TauMin(ige)
             Tau(i,Solution.xbest(1,i,ige))=Solution.TauMin(ige);
            else
            end
         end
       %% Convergence check
 Iconv = sum(Solution.fbest(1:ige)==max(Solution.fbest(1:ige)));
Iwconv = sum(Solution.fbest(1:ige)==Solution.fworst(1:ige)); % Counter of
fbest=fworst
           if Iwconv >100 | Iconv>mxIconv , break;end
 end
 % ACOA Solution results
         [Solution.fopt, jj] = max(Solution.fbest(1:ige));
               Solution.xopt = Solution.xbest(1,:,jj);
          Solution.noGoptima = sum(unique(
Solution.fbest(1:ige)>=Solution.fopt/1.001));
                Solution.ige = ige;
                      [~,mm] = find(Solution.fbest(1:ige) ==
Solution.fopt);
           Solution.FrstGopt = mm(1);
           Solution.Minfbest = min(Solution.fbest(:));
          Solution.Meanfbest = mean(Solution.fbest(:));
           Solution.Stdfbest = std(Solution.fbest(:));
   for i=1:ige
       Solution.GlobalDis(i) = Data.nVar-sum(Solution.xopt ==
Solution.xbest(1,:,i));
   end
                Solution.cFD = sum((Solution.fbest-Solution.Meanfbest).*(
Solution.GlobalDis-mean( Solution.GlobalDis)))/ige;
                  Solution.r = Solution.cFD/(Solution.Stdfbest*std(
Solution.GlobalDis));
            Solution.PerRate = sum(Solution.fbest(:)>=
Solution.fopt/1.001)/(Solution.ige*Imax);
        Solution.ElapsedTime = toc;
        Solution.LinConvRate = zeros(1,ige);
        for i=1:ige
            if Solution.GlobalDis(i)~= 0
        Solution.LinConvRate(i) = (Solution.fopt-
Solution.fbest(i))/Solution.GlobalDis(i);
            else
              Solution.LinConvRate(i) = 0;
              break;
            end
        end
        Solution.maxTauExp=max(Solution.TauMax(1:ige));
        Solution.minTauExp=min(Solution.TauMin(1:ige));

%% Write EHCFMMACO results Output File
```

```
fprintf(fid_ali1,'%s\r\n',['           Number of variables = '
num2str(Data.nVar)]);
fprintf(fid_ali1,'%s\r\n',['                   no. of Ants = '
num2str(nAnts)]);
fprintf(fid_ali1,'%s\r\n',['            Maximum CBLF(fopt) = '
num2str(Solution.fopt)]);
fprintf(fid_ali1,'%s\r\n',['        Optimal Solution(xopt) = '
'[',num2str(Solution.xopt),']']);
fprintf(fid_ali1,'%s\r\n',['   Number of Practical Optima  = '
'[',num2str(Solution.noGoptima),']']);
fprintf(fid_ali1,'%s\r\n',['        no. of Iterations(ige) = '
num2str(ige)]);
fprintf(fid_ali1,'%s\r\n',[' 1st Global optima Found @ ige = '
num2str(Solution.FrstGopt)]);
fprintf(fid_ali1,'%s\r\n',['        Minimum Best  Solution = '
num2str(Solution.Minfbest)]);
fprintf(fid_ali1,'%s\r\n',['        Mean of Best  Solution = '
num2str(Solution.Meanfbest)]);
fprintf(fid_ali1,'%s\r\n',['          SD of Best  Solution = '
num2str(Solution.Stdfbest)]);
fprintf(fid_ali1,'%s\r\n',['  Fitness Distance Correlation = '
num2str(Solution.cFD)]);
fprintf(fid_ali1,'%s\r\n',['        Correlation Coefficent = '
num2str(Solution.r)]);
fprintf(fid_ali1,'%s\r\n',['                  Elapsed Time = '
num2str(Solution.ElapsedTime)]);
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fprintf(fid_ali1,'%3s %10s %11s %10s %10s %9s %5s  %6s
%6s\r\n','ige','Fbest(ige)','Fworst(ige)','Fmean(ige)','BWLocalDis','Globa
lDis','Segma','MaxTau','MinTau');
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fprintf(fid_ali1,'%-3.0f  %-10.3f %-11.3f %-10.3f   %-9.1f  %-5.1f   %-
6.1f%-6.4f  %-
6.4f\r\n',[(1:ige);Solution.fbest(:)';Solution.fworst(:)';Solution.fmean(:
)';Solution.BxWxLocalDis(:)';Solution.GlobalDis(:)';Solution.Segma(:)';Sol
ution.TauMax(:)';Solution.TauMin(:)']);
fprintf(fid_ali1,'%s\r\n','===============================================
=================================');
fclose(fid_ali1);
% end
```

Insertion.m

```
function q=Insertion(p,i1,i2)
    if i1<i2
        q=p([1:i1-1 i1+1:i2 i1 i2+1:end]);
    elseif i1>i2
        q=p([1:i2 i1 i2+1:i1-1 i1+1:end]);
    else
        q=p;
    end
end
```

Xflip.m

```
function p=xflip(p,i)
if p(i)==3
    temp=[1 2];
    p(i)=temp(1,randi(length(temp)));
elseif p(i)==2
    temp=[1 3];
    p(i)=temp(1,randi(length(temp)));
elseif p(i)==1
    temp=[2 3];
    p(i)=temp(1,randi(length(temp)));
end
```

APPENDIX IV

MATLAB FILES OF CRITICAL BUCKLING LOAD FACTOR, INPUT DATA AND OUTPUT PLOTTING FOR COMPOSITE LAMINATED PLATE CASE STUDY

CBLF.m

```
function [CBLF1,delta]=CBLF(x)
% This function devoted to determine the critical buckling load factor,
lamda, of  composite laminate that have the information in LaminateData.m
file

global Data  % Calling Laminate plate data

%% plies Contigency check
  [~,i]=find(x==1);
 [~,ii]=find(x==2);
[~,iii]=find(x==3);
delta=0;
if length(i)>2
   for nk=1:length(i)-2
       if i(nk)==i(nk+2)-2
           delta=.08;break
       end
   end
end

if length(ii)>2
    for nk=1:length(ii)-2
       if ii(nk)==ii(nk+2)-2
           delta=.08;break
       end
    end
end
if length(iii)>2
    for nk=1:length(iii)-2
       if iii(nk)==iii(nk+2)-2
           delta=.08;break
       end
    end
end
%% Plies Orientation Matrix [teta]
   nvar=length(x);
for k=1:nvar
    if x(k)==1
        haf_teta(2*k-1)=0;
        haf_teta(2*k)=0;
    end
    if x(k)==2
        haf_teta(2*k-1)=45;
        haf_teta(2*k)=-45;
    end
    if x(k)==3
        haf_teta(2*k-1)=90;
        haf_teta(2*k)=90;
    end
end
teta=[haf_teta flip(haf_teta)];
haf_teta;
Nplies = length(teta);                  % Number of plies.
%% Stiffness Matrix  [Q]
Data.NU21 = (Data.NU12*Data.E2)/Data.E1;
Q11 = Data.E1/(1 - Data.NU12*Data.NU21);
Q12 =(Data.NU21*Data.E1)/(1 - Data.NU12*Data.NU21);
Q22 = Data.E2/(1 - Data.NU12*Data.NU21);
Q66 = Data.G12;
  Q = [ Q11 Q12 0; Q12 Q22 0; 0 0 Q66];
%% Thickness of i-th ply Group
t = Nplies * Data.h_ply ;
for i = 1:(Nplies+1)
h(i) = -(t/2-((i-1)*(t/Nplies)));
end
%% Determining [A][B][D] Matrices
A=0;B=0;D=0;
for i=1:Nplies
c=cos((teta(1,i)*pi)/180);
s=sin((teta(1,i)*pi)/180);
T = [ c^2 s^2 2*c*s; s^2 c^2 -2*c*s; -c*s c*s (c^2 - s^2)];
Qbar = inv(T) * Q * (inv(T))' ;
A = A + Qbar * (h(1,i+1) - h(1,i));
B = B + 1/2 * Qbar * (h(1,i+1)^2 - h(1,i)^2);
D = D + 1/3 * Qbar * (h(1,i+1)^3 - h(1,i)^3);
end
%% Critical Buckling Load Factor
p=Data.p;
q=Data.q;
r=Data.r;
a=Data.a;
lmda_b=ones(p,q);
for m=1:p
    for n=1:q
        lmda_b(m,n)=((pi^2)*((D(1,1)*(m^4) + 2*(D(1,2) +
2*D(3,3))*((r*m*n)^2) +...
        D(2,2)*((r*n)^4)))/(((a*m)^2)*Data.Nx + ((r*a*n)^2)*Data.Ny));
    end
end
CBLF1=(1-delta)*min(lmda_b(:));
```

LaminateData.m

```
function Data=LaminateData
%Input: None
%Output: Data - structure with optimization problem information
%------------------------------------------------------------------------
%
% Laminate properties
Data.E1 = 18.5e6;      %127.59;      % Elastic Modulus[GPa].
Data.E2 = 1.89e6;      %13.03;       % Elastic Modulus[GPa].
Data.G12 =  .93e6;     %6.41;        % Shear Modulus[GPa].
Data.NU12 = 0.3;       % Poisson ratio.
Data.Xt = 2130;        % Longituddinal Tensile Strength
Data.Xc = 1100;        % Longituddinal compression Strength
Data.Yt = 80;          % Transversal Tensile Strength
Data.Yc = 200;         % Transversal compression Strength
Data.S12 = 160;        % Shear Strength
% Loading conditions
Data.Lr=1;             % Loading ratio
Data.Nx = 1;           % Inplane Load x-direction[N/m].
Data.Ny = Data.Lr*Data.Nx;        % Inplane Load y-direction[N/m].
Data.Nxy = 0;          %[N/m]
% Plate dimensions
Data.h_ply= .005;      %0.127;       %[mm]  Ply thickness
Data.a= 20;            %.508;        % Plate length[m].
Data.b= 10;            %.254;        % Plate  width[m].
Data.r=Data.a/Data.b;             % Aspect ratio.
Data.p= 2;             % Buckling mode in x-direction.
Data.q= 2;             % Buckling mode in y-direction.
% Design Variables & Available Oreintations
Data.nVar=16;          % number of design variables
Data.xd=[0 45 90];     % Vector of possible fiber orientations
% Objective Function Definition
Data.f=@CBLF;          % Defining the objective function
% ACO Algorithm Options
Data.ACOptions =
struct('nAnt',25,'Imax',1000,'rho',0.1,'Tau0',0.004,'zeta',.03);
Data.MMACOptions =
struct('nAnt',25,'Imax',1000,'rho',0.1,'Tau0',1,'pbest',.05,'zeta',.03);
Data.BWACOptions =
struct('nAnt',25,'Imax',1000,'rho',0.1,'Tau0',1,'pbest',.05,'lamda',0.6,'z
eta',.03);
```

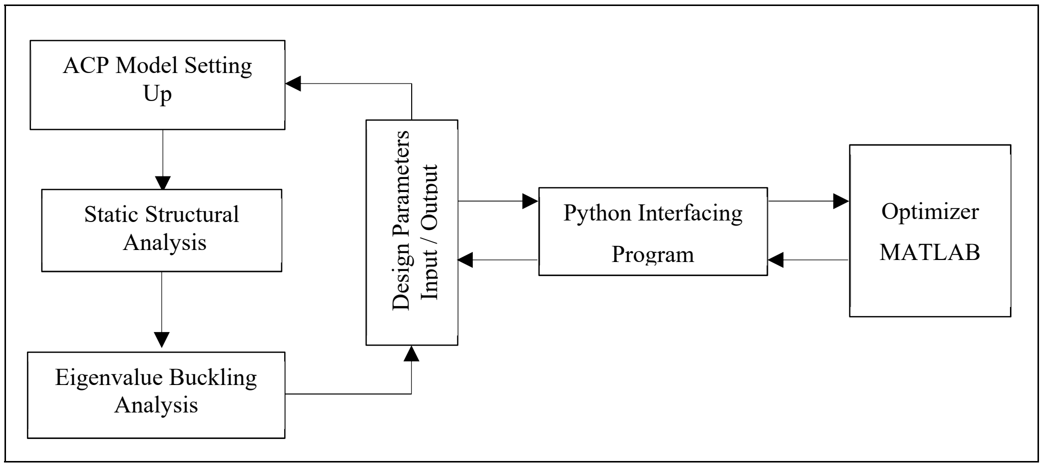
PlotSolution.m

```
function PlotSolution(x,y)
```

```
%  Determining the screen corrdinates
 set(0,'units','inches')
 %Obtains this inch information
 Inch_SS = get(0,'screensize');
 figW=3;figL=9;
tfig=figure (1);

 set(tfig,'Units','inches',...
        'Position',[Inch_SS(3)-figW-1 Inch_SS(4)-figL-1 figW figL],...
        'PaperPositionMode','auto')
 plot(x,1:y,'k-o',...
        'MarkerSize',15,...
        'MarkerFaceColor','y',...
        'LineWidth',1.5);
 axis([0 x(end) 1 y])
 set(gca,...
        'Units','normalized',...
        'YTick',1:y,...
        'XTick',1:x(end),...
        'Position',[.1 .2 .7 .7],...
        'FontUnits','points',...
        'FontWeight','normal',...
        'FontSize',12,...
        'FontName','Times')
  set(gcf, 'MenuBar', 'None')
 xlabel('Orientation angle',...
        'FontUnits','points',...
        'FontWeight','normal',...
        'FontSize',12,...
        'FontName','Times');
 ylabel('Ply number',...
        'FontUnits','points',...
        'interpreter','latex',...
        'FontSize',12,...
        'FontName','Times',...
        'Rotation',90);
        axis equal;

ax = gca;
outerpos = ax.OuterPosition;
ti = ax.TightInset;
left = outerpos(1) + ti(1);
bottom = outerpos(2) + ti(2);
ax_width = outerpos(3) - ti(1) - ti(3);
ax_height = outerpos(4) - ti(2) - ti(4);
ax.Position = [left bottom ax_width ax_height];

        grid on;
        xmin = 0;
        xmax = 4;
        xlim([xmin xmax]);
        set(gca,'XTick',xmin:1:xmax)
        set(gca,'XTickLabel',{[],1,2,3,[]});
        ymin = min(1:y)-1;
        ymax = max(1:y)+1;
        ylim([ymin ymax]);
        set(gca,'YTick',ymin:1: ymax)
        Ylables=1:y;
        set(gca,'YTickLabel',{[],Ylables,[]});

end
```

APPENDIX V

CODES OF ANSYS WORKBENCH AND MATLAB INTERFACING USING PYTHON


ACP Model Setting Up
Static Structural Analysis
Eigenvalue Buckling Analysis
Design Parameters Input / Output
Python Interfacing Program
Optimizer MATLAB

Ansys Work Bench and MATLAB interfacing flowchart

CBLF.m

```
function CBLF1=CBLF(x,B)
     format short
   %% plies Contigency check
  [~,i]=find(x==1);
 [~,ii]=find(x==2);
[~,iii]=find(x==3);
delta=0;
if length(i)>2
   for nk=1:length(i)-2
       if i(nk)==i(nk+2)-2
           delta=.08;break
       end
   end
end

if length(ii)>2
    for nk=1:length(ii)-2
       if ii(nk)==ii(nk+2)-2
           delta=.08;break
       end
    end
end
if length(iii)>2
```

```
    for nk=1:length(iii)-2
       if iii(nk)==iii(nk+2)-2
           delta=.08;break
       end
    end
end

%% ---------------------------------------------------------------------
%%
    % reading writing python filed

    fid = fopen('DV_update.wbjn', 'r');
    fic = fopen('DV_update_0.wbjn', 'w'); % intermediated file.py copy
%-------------------------------
while ~feof(fid)
        tline = fgetl(fid);
        fprintf(fic,'%s\n',tline);
        for k=1:length(B)
            str1=strcat(['    Parameter=parameter',num2str(k),',']);

            if(strcmp(tline,str1))
%             fprintf(fic,'    Expression="%4.2f [mm]"\n',v_ali(2,1));
if (1<=k)&&(k<=3)
            fprintf(fic,'    Expression="%12.4f [mm]")\n',B(1,k));
elseif (4<=k)&&(k<=27)
            fprintf(fic,'    Expression="%2.0f")\n',B(1,k));
elseif k==28 && B(1,k)==0
    fprintf(fic,'    Expression="%20.15f -1E-14 [N]")\n',B(1,k));
elseif k==28 && B(1,k)~=0
    fprintf(fic,'    Expression="%20.15f [N]")\n',B(1,k));
elseif k==29 && B(1,k)==0
    fprintf(fic,'    Expression="%20.15f -1E-14 [N]")\n',B(1,k));
elseif k==29 && B(1,k)~=0
    fprintf(fic,'    Expression="%20.15f [N]")\n',B(1,k));
end
    tline = fgetl(fid);
            end
        end
       % -----------------
   end
    fclose(fic);
    fclose(fid); % Close file.py


%-----------------------------------------------------------------------
% Replacing intermediate file with original file name
%-------------------------------
    fid_v1 = fopen('DV_update_0.wbjn', 'r'); % open intermediate file.py
to copy
    fid_v0 = fopen('DV_update.wbjn', 'w'); % print into original file.py
%-------------------------------
   while ~feof(fid_v1)
        tline = fgetl(fid_v1);
        fprintf(fid_v0,'%s\n',tline);
   end
    fclose(fid_v0);
    fclose(fid_v1); % close file.py
% -----------------------------------------------------------------------
%

dos('"C:\Program Files\ANSYS Inc\ANSYS
Student\v195\Framework\bin\Win64\RunWB2.exe" -B -R "DV_update.wbjn" ')

% Opening and reading the Ansys WB results file (Output.txt)
fidd=fopen('Output.txt','r');
m = 1;
tline = fgetl(fidd);
A{m} = tline;
while ischar(tline)
    m = m+1;
    tline = fgetl(fidd);
    A{m} = tline;
end
fclose(fidd);
lmda=A{30}; % reading the buckling load factor
% % (1-delta)*
CBLF1=sscanf(lmda,'P30,%12f [N]');
```

Python Interfacing File

```
# encoding: utf-8
# 2019 R3
SetScriptVersion(Version="19.5.112")
Open(FilePath="C:/Users/aliah/OneDrive -
ETS/03_Reserach_Activities_Spring2020/03_5th_Journal_paper/07_New_ANSYS_WB
_Models/Uncertainity_24_ply_laminate_model.wbpj")
os.remove("C:/Users/aliah/OneDrive -
ETS/03_Reserach_Activities_Spring2020/03_5th_Journal_paper/07_New_ANSYS_WB
_Models/Uncertainity_24_ply_laminate_model_files/.lock")

designPoint1 = Parameters.GetDesignPoint(Name="0")
parameter1 = Parameters.GetParameter(Name="P1")
designPoint1.SetParameterExpression(
    Parameter=parameter1,
    Expression="    139.5000 [mm]")
parameter2 = Parameters.GetParameter(Name="P2")
designPoint1.SetParameterExpression(
    Parameter=parameter2,
    Expression="    139.5000 [mm]")
parameter3 = Parameters.GetParameter(Name="P3")
designPoint1.SetParameterExpression(
    Parameter=parameter3,
    Expression="    111.6000 [mm]")
parameter4 = Parameters.GetParameter(Name="P4")
designPoint1.SetParameterExpression(
    Parameter=parameter4,
    Expression="45")
```

```
parameter5 = Parameters.GetParameter(Name="P5")
designPoint1.SetParameterExpression(
    Parameter=parameter5,
    Expression="-45")
parameter6 = Parameters.GetParameter(Name="P6")
designPoint1.SetParameterExpression(
    Parameter=parameter6,
    Expression="45")
parameter7 = Parameters.GetParameter(Name="P7")
designPoint1.SetParameterExpression(
    Parameter=parameter7,
    Expression="-45")
parameter8 = Parameters.GetParameter(Name="P8")
designPoint1.SetParameterExpression(
    Parameter=parameter8,
    Expression="45")
parameter9 = Parameters.GetParameter(Name="P9")
designPoint1.SetParameterExpression(
    Parameter=parameter9,
    Expression="-45")
parameter10 = Parameters.GetParameter(Name="P10")
designPoint1.SetParameterExpression(
    Parameter=parameter10,
    Expression="45")
parameter11 = Parameters.GetParameter(Name="P11")
designPoint1.SetParameterExpression(
    Parameter=parameter11,
    Expression="-45")
parameter12 = Parameters.GetParameter(Name="P12")
designPoint1.SetParameterExpression(
    Parameter=parameter12,
    Expression="45")
parameter13 = Parameters.GetParameter(Name="P13")
designPoint1.SetParameterExpression(
    Parameter=parameter13,
    Expression="-45")
parameter14 = Parameters.GetParameter(Name="P14")
designPoint1.SetParameterExpression(
    Parameter=parameter14,
    Expression="45")
parameter15 = Parameters.GetParameter(Name="P15")
designPoint1.SetParameterExpression(
    Parameter=parameter15,
    Expression="-45")
parameter16 = Parameters.GetParameter(Name="P16")
designPoint1.SetParameterExpression(
    Parameter=parameter16,
    Expression="-45")
parameter17 = Parameters.GetParameter(Name="P17")
designPoint1.SetParameterExpression(
    Parameter=parameter17,
    Expression="45")
parameter18 = Parameters.GetParameter(Name="P18")
designPoint1.SetParameterExpression(
    Parameter=parameter18,
    Expression="-45")
parameter19 = Parameters.GetParameter(Name="P19")
designPoint1.SetParameterExpression(
    Parameter=parameter19,
    Expression="45")
parameter20 = Parameters.GetParameter(Name="P20")
designPoint1.SetParameterExpression(
    Parameter=parameter20,
    Expression="-45")
parameter21 = Parameters.GetParameter(Name="P21")
designPoint1.SetParameterExpression(
    Parameter=parameter21,
    Expression="45")
parameter22 = Parameters.GetParameter(Name="P22")
designPoint1.SetParameterExpression(
    Parameter=parameter22,
    Expression="-45")
parameter23 = Parameters.GetParameter(Name="P23")
designPoint1.SetParameterExpression(
    Parameter=parameter23,
    Expression="45")
parameter24 = Parameters.GetParameter(Name="P24")
designPoint1.SetParameterExpression(
    Parameter=parameter24,
    Expression="-45")
parameter25 = Parameters.GetParameter(Name="P25")
designPoint1.SetParameterExpression(
    Parameter=parameter25,
    Expression="45")
parameter26 = Parameters.GetParameter(Name="P26")
designPoint1.SetParameterExpression(
    Parameter=parameter26,
    Expression="-45")
parameter27 = Parameters.GetParameter(Name="P27")
designPoint1.SetParameterExpression(
    Parameter=parameter27,
    Expression="45")
parameter28 = Parameters.GetParameter(Name="P28")
designPoint1.SetParameterExpression(
    Parameter=parameter28,
    Expression="  -0.279000000000000 [N]")
parameter29 = Parameters.GetParameter(Name="P29")
designPoint1.SetParameterExpression(
    Parameter=parameter29,
    Expression="  -0.000000000000010 [N]")


Update()

# writing the output file of Ansys WB

logFile = open("C:/Users/aliah/OneDrive -
ETS/03_Reserach_Activities_Spring2020/03_5th_Journal_paper/07_New_ANSYS_WB
_Models/output.txt","w")
```

```
for Parameter in Parameters.GetAllParameters():
        value=Parameter.Value.ToString()
        logFile.write(Parameter.Name + "," + value + "\n")
        logFile.flush()
logFile.close()

Save(Overwrite=True)
```

LIST OF BIBLIOGRAPHICAL REFERENCES

Abolghasemi, S., Eipakchi, H., & Shariati, M. (2019). An analytical solution for buckling of plates with circular cutout subjected to non-uniform in-plane loading. *Archive of Applied Mechanics, 89*(12), 2519-2543.

Adali, S., Lene, F., Duvaut, G., & Chiaruttini, V. (2003). Optimization of laminated composites subject to uncertain buckling loads. *Composite Structures, 62*(3-4), 261-269.

Agrawal, S., Panda, R., Bhuyan, S., & Panigrahi, B. K. (2013). Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm. *Swarm and Evolutionary Computation, 11*, 16-30.

Ahmid, A., Le, V. N., & Dao, T. M. (2017). An optimization procedure for overhead gantry crane exposed to buckling and yield criteria. *International Journal of Technology and Engineering, 8*(2), 11.

Ahmid, A., Thien-My, D., & Le, V. N. (2020). An Adaptive Discrete Cuckoo Search Algorithm to Solve Structural Optimization Problems. *Journal of Multidisciplinary Engineering Science and Technology, 7*(6).

Ahmid, A., Thien-My, D., & Van Ngan, L. (2019). Comparison Study of Discrete Optimization Problem Using Meta-Heuristic Approaches: A Case Study. *International Journal of Industrial Engineering, 1*(2), 97-109.

Alhorani, R. A. (2020). Mathematical models for the optimal design of I-and H-shaped crane bridge girders. *Asian Journal of Civil Engineering, 21*(4), 707-722.

Ali, M. M., Törn, A., & Viitanen, S. (2002). A direct search variant of the simulated annealing algorithm for optimization involving continuous variables. *Computers & Operations Research, 29*(1), 87-102.

Almufti, S. M. (2019). Historical survey on metaheuristics algorithms. *International Journal Of Scientific World, 7*(1), 1.

Ansys. (2015). Workbench user's guide In (Relase 15 ed.). Canonsburg PA,USA: Ansys corporation

Awad, Z. K. (2012). *Novel fibre composite civil engineering sandwich structures: behaviour, analysis, and optimum design.* University of Southern Queensland,

Aymerich, F., & Serra, M. (2008). Optimization of laminate stacking sequence for maximum buckling load using the ant colony optimization (ACO) metaheuristic. *Composites Part A: Applied Science and Manufacturing, 39*(2), 262-272.

Barroso, E. S., Parente, E., & de Melo, A. M. C. (2017). A hybrid PSO-GA algorithm for optimization of laminated composites. *Structural and Multidisciplinary Optimization, 55*(6), 2111-2130. doi:10.1007/s00158-016-1631-y

Beck, A. T., & Gomes, W. J. d. S. (2012). A comparison of deterministic, reliability-based and risk-based structural optimization under uncertainty. *Probabilistic Engineering Mechanics, 28*, 18-29. doi:https://doi.org/10.1016/j.probengmech.2011.08.007

Bloomfield, M. W., Herencia, J. E., & Weaver, P. M. (2010). Analysis and benchmarking of meta-heuristic techniques for lay-up optimization. *Computers & Structures, 88*(5-6), 272-282. doi:10.1016/j.compstruc.2009.10.007

Bloomfield, M. W., Herencia, J. E., & Weaver, P. M. (2010). Analysis and benchmarking of meta-heuristic techniques for lay-up optimization. *Computers & Structures, 88*(5), 272-282. doi:https://doi.org/10.1016/j.compstruc.2009.10.007

Blum, C., & Dorigo, M. (2004). The hyper-cube framework for ant colony optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 34*(2), 1161-1172.

Bullnheimer, B., Hartl, R. F., & Strauss, C. (1997). A new rank based version of the Ant System. A computational study.

Cheng, J. (2010). Optimum design of steel truss arch bridges using a hybrid genetic algorithm. *Journal of Constructional Steel Research, 66*(8-9), 1011-1017.

Clune, R. P. (2013). *Algorithm selection in structural optimization.* Massachusetts Institute of Technology,

CMAA. (2010). Specification 74, . In *Specifications for Top Running and Under Running Single Girder Electric Overhead Traveling Cranes Utilizing Under Running Trolley Hoist*. Charlotte, NC: MHI.

Cobos, C., Muñoz-Collazos, H., Urbano-Muñoz, R., Mendoza, M., León, E., & Herrera-Viedma, E. (2014). Clustering of web search results based on the cuckoo search algorithm and Balanced Bayesian Information Criterion. *Information Sciences, 281*, 248-264.

Cohn, M., & Dinovitzer, A. (1994). Application of structural optimization. *Journal of Structural Engineering, 120*(2), 617-650.

da Silva, G. A., Cardoso, E. L., & Beck, A. T. (2020). Comparison of robust, reliability-based and non-probabilistic topology optimization under uncertain loads and stress constraints. *Probabilistic Engineering Mechanics, 59*, 103039.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., & dos Reis Alves, S. F. (2017). Artificial neural network architectures and training processes. In *Artificial neural networks* (pp. 21-28): Springer.

de Almeida, F. S. (2016). Stacking sequence optimization for maximum buckling load of composite plates using harmony search algorithm. *Composite Structures, 143*, 287-299.

de Castro Lemonge, A. C., Duarte, G. R., & da Fonseca, L. G. (2019). An algorithm inspired by bee colonies coupled to an adaptive penalty method for truss structural optimization problems. *Journal of the Brazilian Society of Mechanical Sciences and Engineering, 41*(3), 1-19.

de Faria, A. R. (2002). Buckling optimization and antioptimization of composite plates: uncertain loading combinations. *International Journal for Numerical Methods in Engineering, 53*(3), 719-732. doi:10.1002/nme.309

de Moura Meneses, A. A., da Silva, P. V., Nast, F. N., Araujo, L. M., & Schirru, R. (2020). Application of Cuckoo Search algorithm to Loading Pattern Optimization problems. *Annals of Nuclear Energy, 139*, 107214.

Degertekin, S. (2012). Improved harmony search algorithms for sizing optimization of truss structures. *Computers & Structures, 92*, 229-241.

Degertekin, S., & Hayalioglu, M. (2013). Sizing truss structures using teaching-learning-based optimization. *Computers & Structures, 119*, 177-188.

Dennis, W. (2010). A Rigorous Analysis of the Harmony Search Algorithm: How the Research Community can be Misled by a "Novel" Methodology. *International Journal of Applied Metaheuristic Computing (IJAMC), 1*(2), 50-60. doi:10.4018/jamc.2010040104

Deveci, H. A., Aydin, L., & Seçil Artem, H. (2016). Buckling optimization of composite laminates using a hybrid algorithm under Puck failure criterion constraint. *Journal of Reinforced Plastics and Composites, 35*(16), 1233-1247. doi:10.1177/0731684416646860

Dhuban, S., Karuppanan, S., Mengal, A., & Patil, S. (2017). Effect of fiber orientation and ply stacking sequence on buckling behaviour of basalt-carbon hybrid composite laminates.

Dorigo, M. (1991). Ant Colony Optimization—new optimization techniques in engineering. *by Onwubolu, GC, and BV Babu, Springer-Verlag Berlin Heidelberg*, 101-117.

Dorigo, M. (1997). Luca Maria Gambardella: ant colony system: a cooperative learning. *IEEE Trans Evol Comput, 1*, 53-66.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE computational intelligence magazine, 1*(4), 28-39.

Dorigo, M., & Stützle, T. (2019). Ant colony optimization: overview and recent advances. In *Handbook of metaheuristics* (pp. 311-351): Springer.

Ehsani, A., & Rezaeepazhand, J. (2016). Stacking sequence optimization of laminated composite grid plates for maximum buckling load using genetic algorithm. *International Journal of Mechanical Sciences, 119*, 97-106. doi:10.1016/j.ijmecsci.2016.09.028

Elishakoff, I., Haftka, R., & Fang, J. (1994). Structural design under bounded uncertainty—optimization with anti-optimization. *Computers & Structures, 53*(6), 1401-1405.

Elishakoff, I., & Ohsaki, M. (2010). *Optimization and anti-optimization of structures under uncertainty*: World Scientific.

Erdal, O., & Sonmez, F. O. (2005). Optimum design of composite laminates for maximum buckling load capacity using simulated annealing. *Composite Structures, 71*(1), 45-52.

França, P. M., Sosa, N. M., & Pureza, V. (1999). An adaptive tabu search algorithm for the capacitated clustering problem. *International Transactions in Operational Research, 6*(6), 665-678. doi:10.1111/j.1475-3995.1999.tb00180.x

Gambardella, L. M., & Dorigo, M. (2000). An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing, 12*(3), 237-255.

Gandomi, A. H., & Yang, X.-S. (2011). Benchmark problems in structural optimization. In *Computational optimization, methods and algorithms* (pp. 259-281): Springer.

Gandomi, A. H., Yang, X.-S., & Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers, 29*(1), 17-35.

Gąska, D., Haniszewski, T., & Margielewicz, J. (2017). I-beam girders dimensioning with numerical modelling of local stresses in wheel-supporting flanges. *Mechanics, 23*(3), 347-352.

Gherboudj, A., Layeb, A., & Chikhi, S. (2012). Solving 0-1 knapsack problems by a discrete binary version of cuckoo search algorithm. *International Journal of Bio-Inspired Computation, 4*(4), 229-236.

Ghiasi, H., Fayazbakhsh, K., Pasini, D., & Lessard, L. (2010). Optimum stacking sequence design of composite materials Part II: Variable stiffness design. *Composite Structures, 93*(1), 1-13.

Gholizadeh, S., & Milany, A. (2018). An improved fireworks algorithm for discrete sizing optimization of steel skeletal structures. *Engineering Optimization, 50*(11), 1829-1849.

Gold, S., & Krishnamurty, S. (1997). *Trade-offs in robust engineering design.* Paper presented at the Proceedings of the ASME Design Engineering Technical Conferences.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning, 3*(2), 95-99.

Gurav, S., & Goosen, J. (2005). Bounded-but-unknown uncertainty optimization using design sensitivities and parallel computing: application to MEMS. *Computers & Structures, 83*(14), 1134-1149.

Gürdal, Z., Haftka, R. T., & Hajela, P. (1999). *Design and optimization of laminated composite materials*: John Wiley & Sons.

Han, X., Jiang, C., Gong, S., & Huang, Y. (2008). Transient waves in composite-laminated plates with uncertain load and material property. *International journal for numerical methods in engineering, 75*(3), 253-274.

He, L., de Weerdt, M., & Yorke-Smith, N. (2019). Time/sequence-dependent scheduling: the design and evaluation of a general purpose tabu-based adaptive large neighbourhood search algorithm. *Journal of Intelligent Manufacturing*, 1-28.

Hinton, G. E. (1986). *Learning distributed representations of concepts.* Paper presented at the Proceedings of the eighth annual conference of the cognitive science society.

Ho-Huu, V., Nguyen-Thoi, T., Vo-Duy, T., & Nguyen-Trang, T. (2016). An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. *Computers & Structures, 165*, 59-75.

Hu, H.-T., & Lin, B.-H. (1995). Buckling optimization of symmetrically laminated plates with various geometries and end conditions. *Composites Science and Technology, 55*(3), 277-285.

Iman, R. L., Helton, J. C., & Campbell, J. E. (1981). An approach to sensitivity analysis of computer models: Part I—Introduction, input variable selection and preliminary variable assessment. *Journal of quality technology, 13*(3), 174-183.

Jati, G. K., & Manurung, H. M. (2012). *Discrete cuckoo search for traveling salesman problem.* Paper presented at the 2012 7th International Conference on Computing and Convergence Technology (ICCCT).

Javidrad, F., Nazari, M., & Javidrad, H. (2017). Optimum stacking sequence design of laminates using a hybrid PSO-SA method. *Composite Structures*.

Jiang, C., Han, X., & Liu, G. (2008). Uncertain optimization of composite laminated plates using a nonlinear interval number programming method. *Computers & Structures, 86*(17-18), 1696-1703.

Jiao, L., Luo, J., Shang, R., & Liu, F. (2014). A modified objective function method with feasible-guiding strategy to solve constrained multi-objective optimization problems. *Applied Soft Computing, 14*, 363-380.

Jing, Z., Fan, X., & Sun, Q. (2015). Stacking sequence optimization of composite laminates for maximum buckling load using permutation search algorithm. *Composite Structures, 121*, 225-236. doi:10.1016/j.compstruct.2014.10.031

Jones, R. M. (2014). *Mechanics of composite materials*: CRC press.

Jones, T., & Forrest, S. (1995). *Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms.* Paper presented at the ICGA.

Kalantari, M., Dong, C., & Davies, I. J. (2017). Multi-objective robust optimization of multi-directional carbon/glass fibre-reinforced hybrid composites with manufacture related uncertainties under flexural loading. *Composite Structures, 182*, 132-142.

Katagiri, H., Hayashida, T., Nishizaki, I., & Guo, Q. (2012). A hybrid algorithm based on tabu search and ant colony optimization for k-minimum spanning tree problems. *Expert Systems with Applications, 39*(5), 5681-5686.

Kaveh, A. (2017). *Applications of metaheuristic optimization algorithms in civil engineering*: Springer.

Kaveh, A., & Bakhshpoori, T. (2013). Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights. *The Structural Design of Tall and Special Buildings, 22*(13), 1023-1036.

Kaveh, A., & Bakhshpoori, T. (2016). A new metaheuristic for continuous structural optimization: water evaporation optimization. *Structural and Multidisciplinary Optimization, 54*(1), 23-43.

Kaveh, A., Bakhshpoori, T., & Afshari, E. (2014). An efficient hybrid particle swarm and swallow swarm optimization algorithm. *Computers & Structures, 143*, 40-59.

Kaveh, A., Dadras, A., & Malek, N. G. (2017). Buckling load of laminated composite plates using three variants of the biogeography-based optimization algorithm. *Acta Mechanica, 229*(4), 1551-1566. doi:10.1007/s00707-017-2068-0

Kaveh, A., Dadras, A., & Malek, N. G. (2019). Robust design optimization of laminated plates under uncertain bounded buckling loads. *Structural and Multidisciplinary Optimization, 59*(3), 877-891.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations*.

Kaw, A. (2006). *Mechanics of composite materials*.

Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization (PSO).* Paper presented at the Proc. IEEE International Conference on Neural Networks, Perth, Australia.

Kim, J.-S., Kim, N.-P., & Han, S.-H. (2005). Optimal stiffness design of composite laminates for a train carbody by an expert system and enumeration method. *Composite Structures, 68*(2), 147-156.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science, 220*(4598), 671-680.

Kogiso, N., Watson, L. T., Gürdal, Z., & Haftka, R. T. (1994). Genetic algorithms with local improvement for composite laminate design. *Structural optimization, 7*(4), 207-218.

Koide, R. M., de Franca, G. V., & Luersen, M. A. (2013). An ant colony algorithm applied to lay-up optimization of laminated composite plates. *Latin American Journal of Solids and Structures, 10*(3), 491-504. doi:Doi 10.1590/S1679-78252013000300003

Koide, R. M., França, G. v. Z. d., & Luersen, M. A. (2013). An ant colony algorithm applied to lay-up optimization of laminated composite plates. *Latin American Journal of Solids and Structures, 10*(3), 491-504.

Koide, R. M., & Luersen, M. A. (2013). Maximization of Fundamental Frequency of Laminated Composite Cylindrical Shells by Ant Colony Algorithm. *Journal of Aerospace Technology and Management, 5*(1). doi:10.5028/jatm.v5i1.233

Krempser, E., Bernardino, H. S., Barbosa, H. J., & Lemonge, A. C. (2017). Performance evaluation of local surrogate models in differential evolution-based optimum design of truss structures. *Engineering Computations*.

Kumar, A., & Arakerimath, R. R. (2016). Numerical and experimental buckling analysis of crane girder. *International Journal of Research in Engineering and Technology, 5*(06), 192-197.

Kumar, A., & Rangavittal, H. (2019). Genetic Algorithm Parameter Effect on 3D Truss Optimization with Discrete Variable. *Advanced Journal of Graduate Research, 5*(1), 61-70.

Kumar, D., & Singh, S. B. (2012). Stability and failure of composite laminates with various shaped cutouts under combined in-plane loads. *Composites Part B: Engineering, 43*(2), 142-149. doi:https://doi.org/10.1016/j.compositesb.2011.09.005

Laguna, M., Barnes, J. W., & Glover, F. W. (1991). Tabu search methods for a single machine scheduling problem. *Journal of Intelligent Manufacturing, 2*(2), 63-73. doi:10.1007/bf01471219

Layeb, A. (2011). A novel quantum inspired cuckoo search for knapsack problems. *International Journal of Bio-Inspired Computation, 3*(5), 297-305.

Le Riche, R., & Haftka, R. (1995). Improved genetic algorithm for minimum thickness composite laminate design. *Composites Engineering, 5*(2), 143-161.

Le Riche, R., & Haftka, R. T. (1993). Optimization of laminate stacking sequence for buckling load maximization by genetic algorithm. *AIAA journal, 31*(5), 951-956.

Lê, V., & Champliaud, H. (2014). *Safety factor of welded-plate beams based on finite element linear buckling analysis*.

Lee, H. (2014). Finite Element Simulations with ANSYS Workbench 14: Theory. *Applications, Case Studies*.

Li, Q., Liu, S.-Y., & Yang, X.-S. (2020). Influence of initialization on the performance of metaheuristic optimizers. *Applied Soft Computing*, 106193.

Li, Z., Dey, N., Ashour, A. S., & Tang, Q. (2018). Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem. *Neural Computing and Applications, 30*(9), 2685-2696.

Liao, Y.-S., & Chiou, C.-Y. (2006). Robust optimum designs of fiber-reinforced composites using constraints with sensitivity. *Journal of composite materials, 40*(22), 2067-2081.

Liu, B., Haftka, R. T., Akgün, M. A., & Todoroki, A. (2000). Permutation genetic algorithm for stacking sequence design of composite laminates. *Computer methods in applied mechanics and engineering, 186*(2-4), 357-372.

Liu, P., Xing, L., Liu, Y., & Zheng, J. (2014). Strength analysis and optimal design for main girder of double-trolley overhead traveling crane using finite element method. *Journal of Failure Analysis and Prevention, 14*(1), 76-86.

Lombardi, M., HAFTKA, R., & Cinquini, C. (1992). *Optimization of composite plates for buckling by simulated annealing.* Paper presented at the 33rd Structures, Structural Dynamics and Materials Conference.

Lombardi, M., & Haftka, R. T. (1998). Anti-optimization technique for structural design under load uncertainties. *Computer methods in applied mechanics and engineering, 157*(1-2), 19-31.

Loubna, B., Mohamed, S., Abdelaziz, E., & Fatimaezzahra, M. (2017). A Novel adaptive Discrete Cuckoo Search Algorithm for parameter optimization in computer vision.

Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 20(60), 51-71.

M. Dorigo, V. M. a. A. C. (1991). Positive Feedback as a Search Strategy. *Dipartimento di Elettronica, Politecnico di Milano*, 91-016. Retrieved from http://iridia.ulb.ac.be/~mdorigo/pub_x_subj.html

Malan, K. M., & Engelbrecht, A. P. (2014). Fitness landscape analysis for metaheuristic performance prediction. In *Recent advances in the theory and application of fitness landscapes* (pp. 103-132): Springer.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software, 95*, 51-67.

Montáns, F. J., Chinesta, F., Gómez-Bombarelli, R., & Kutz, J. N. (2019). Data-driven modeling and learning in science and engineering. *Comptes Rendus Mécanique, 347*(11), 845-855.

Mortazavi, A., Toğan, V., & Moloodpoor, M. (2019). Solution of structural and mathematical optimization problems using a new hybrid swarm intelligence optimization algorithm. *Advances in engineering software, 127*, 106-123.

Narayana, A. L., Rao, K., & Kumar, R. V. (2013). FEM buckling analysis of quasi-isotropic symmetrically laminated rectangular composite plates with a square/rectangular cutout. *Journal of Mechanical Science and Technology, 27*(5), 1427-1435.

Narayana, A. L., Rao, K., & Kumar, R. V. (2014). Buckling analysis of rectangular composite plates with rectangular cutout subjected to linearly varying in-plane loading using fem. *Sadhana, 39*(3), 583-596.

Nikbakt, S., Kamarian, S., & Shakeri, M. (2018). A review on optimization of composite structures Part I: Laminated composites. *Composite Structures, 195*, 158-185. doi:10.1016/j.compstruct.2018.03.063

Ouaarab, A., Ahiod, B., & Yang, X.-S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications, 24*(7-8), 1659-1669.

Pai, N., Kaw, A., & Weng, M. (2003). Optimization of laminate stacking sequence for failure load maximization using Tabu search. *Composites Part B: Engineering, 34*(4), 405-413.

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural computing, 1*(2-3), 235-306.

Pavlovic , G., Savkovic, M., Zdravkovic, N., Bulatovic, R., & Markovic, G. (2018). Analysis and Optimization Design of Welded I-girder of the Single-beam Bridge Crane.

Peeters, D., & Abdalla, M. (2017). Design Guidelines in Nonconventional Composite Laminate Optimization. *Journal of Aircraft, 54*(4), 1454-1464. doi:10.2514/1.c034087

Piechocki, J., Ambroziak, D., Palkowski, A., & Redlarski, G. (2014). Use of Modified Cuckoo Search algorithm in the design process of integrated power systems for modern and energy self-sufficient farms. *Applied Energy, 114*, 901-908.

Popov, E. P. (1976). *Mechanics of Materials: Solutions for Problems*: Prentice-Hall.

PTC. (2011). PTC Mathcad 15 M010 User's Manual. In (15 ed.): PTC.

Qiu, Z., & Wang, X. (2010). Structural anti-optimization with interval design parameters. *Structural and Multidisciplinary Optimization, 41*(3), 397-406.

Qu, X., Xu, G., Fan, X., & Bi, X. (2015). Intelligent optimization methods for the design of an overhead travelling crane. *Chinese Journal of Mechanical Engineering, 28*(1), 187-196.

Ragsdell, K., & Phillips, D. (1976). Optimal design of a class of welded structures using geometric programming.

Rama Mohan Rao, A. (2009). Lay-up sequence design of laminate composite plates and a cylindrical skirt using ant colony optimization. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering, 223*(1), 1-18.

Rao, A. R. M. (2009). Lay-up sequence design of laminate composite plates and a cylindrical skirt using ant colony optimization. *Proceedings of the Institution of Mechanical Engineers Part G-Journal of Aerospace Engineering, 223*(G1), 1-18. doi:10.1243/09544100jaero415

Rao, A. R. M., & Arvind, N. (2007). Optimal stacking sequence design of laminate composite structures using tabu embedded simulated annealing. *Structural Engineering and Mechanics, 25*(2), 239-268.

Rao, A. R. M., & Shyju, P. P. (2008). Development of a hybrid meta-heuristic algorithm for combinatorial optimisation and its application for optimal design of laminated composite cylindrical skirt. *Computers & Structures, 86*(7), 796-815. doi:https://doi.org/10.1016/j.compstruc.2007.05.033

Rao, S. S. (2009). *Engineering optimization: theory and practice*: John Wiley & Sons.

Reddy, J. N. (2004). *Mechanics of laminated composite plates and shells: theory and analysis*: CRC press.

Sadollah, A., Eskandar, H., Bahreininejad, A., & Kim, J. H. (2015). Water cycle, mine blast and improved mine blast algorithms for discrete sizing optimization of truss structures. *Computers & Structures, 149*, 1-16.

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

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*: Cambridge university press.

Shehab, M. (2020). Introduction of Diffusion MRI and Cuckoo Search Algorithm. In *Artificial Intelligence in Diffusion MRI* (pp. 1-12): Springer.

Shehab, M., Khader, A. T., & Al-Betar, M. A. (2017). A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing, 61*, 1041-1059.

Singh, J. (2017). Introduction to Optimum design. In: Iowa: The university of Iowa, College of Engineering.

Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research, 22*(1), 3-18. doi:10.1111/itor.12001

Söyleyici, M. U. (2011). *Stacking sequences optimization of the anti-buckled laminated composites considering various failure criteria.* İzmir Institute of Technology,

Stützle, T., & Hoos, H. H. (2000). MAX–MIN ant system. *Future generation computer systems, 16*(8), 889-914.

Talatahari, S., Kheirollahi, M., Farahmandpour, C., & Gandomi, A. H. (2013). A multi-stage particle swarm for optimum design of truss structures. *Neural Computing and Applications, 23*(5), 1297-1309.

Talbi, E.-G. (2009). *Metaheuristics: from design to implementation* (Vol. 74): John Wiley & Sons.

test 1. In.

Todoroki, A., & Haftka, R. T. (1998). Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy. *Composites Part B: Engineering, 29*(3), 277-285.

Vasiliev, V. V. (2017). *Mechanics of composite structures*: CRC Press.

Venter, G., & Haftka, R. (1996). *A two species genetic algorithm for designing composite laminates subjected to uncertainty.* Paper presented at the 37th Structure, Structural Dynamics and Materials Conference.

Vosoughi, A., Darabi, A., & Forkhorji, H. D. (2017). Optimum stacking sequences of thick laminated composite plates for maximizing buckling load using FE-GAs-PSO. *Composite Structures, 159*, 361-367.

Wang, G. G. (2003). Adaptive Response Surface Method Using Inherited Latin Hypercube Design Points. *Journal of Mechanical Design, 125*(2), 210-220. doi:10.1115/1.1561044

Weiler, C., Biesinger, B., Hu, B., & Raidl, G. R. (2015). *Heuristic Approaches for the Probabilistic Traveling Salesman Problem.* Paper presented at the International Conference on Computer Aided Systems Theory.

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67-82. doi:10.1109/4235.585893

Wu, Z. (2020). 6 - Postbuckling analysis and optimization of laminated composite plates with applications in aerospace. In P. Irving & C. Soutis (Eds.), *Polymer Composites in the Aerospace Industry (Second Edition)* (pp. 123-146): Woodhead Publishing.

Xin, Z., Zhang, D., & Chen, Z. (2019). *Spectrum Allocation of Cognitive Radio Network Based on Improved Cuckoo Search Algorithm.* Paper presented at the Proceedings of the 2nd International Conference on Computer Science and Software Engineering.

Xingyu, R., Jiayi, F., & Hai, H. (2020). *Improved genetic algorithm with two-level multipoint approximation for complex frame structural optimization.* Paper presented at the Journal of Physics: Conference Series.

Yadav, A., & Arora, M. (2019). Library Services for Divyangjan in the National Institute for the Empowerment of Persons With Intellectual Disabilities. *International Information & Library Review, 51*(1), 70-74.

Yang, W. Y., Cao, W., Kim, J., Park, K. W., Park, H.-H., Joung, J., . . . Im, T. (2005). *Applied numerical methods using MATLAB*: John Wiley & Sons.

Yang, X.-S. (2013). *Cuckoo search and firefly algorithm: Theory and applications* (Vol. 516): Springer.

Yang, X.-S. (2014). *Nature-inspired optimization algorithms*: Elsevier.

Yang, X.-S., & Deb, S. (2009). *Cuckoo search via Lévy flights.* Paper presented at the 2009 World congress on nature & biologically inspired computing (NaBIC).

Yang, X.-S., & Deb, S. (2013). Multiobjective cuckoo search for design optimization. *Computers & Operations Research, 40*(6), 1616-1624.

Ye, P., & Pan, G. (2017). Global optimization method using ensemble of metamodels based on fuzzy clustering for design space reduction. *Engineering with computers, 33*(3), 573-585. doi:10.1007/s00366-016-0490-x

Yuan, Y., Lv, L., Wang, X., & Song, X. (2020). Optimization of a frame structure using the Coulomb force search strategy-based dragonfly algorithm. *Engineering Optimization, 52*(6), 915-931.

Zadeh, P. M., Fakoor, M., & Mohagheghi, M. (2018). Bi-level optimization of laminated composite structures using particle swarm optimization algorithm. *Journal of Mechanical Science and Technology, 32*(4), 1643-1652. Retrieved from <Go to ISI>://WOS:000430417300018

Zein, S., Madhavan, V., Dumas, D., Ravier, L., & Yague, I. (2016). From stacking sequences to ply layouts: An algorithm to design manufacturable composite structures. *Composite Structures, 141*, 32-38. doi:10.1016/j.compstruct.2016.01.027

Zhang, Y., Wang, H., Zhang, Y., & Chen, Y. (2011). *Best-worst ant system.* Paper presented at the 2011 3rd International Conference on Advanced Computer Control.

Zheng, F., Zecchin, A. C., Newman, J. P., Maier, H. R., & Dandy, G. C. (2017). An adaptive convergence-trajectory controlled ant colony optimization algorithm with application to water distribution system design problems. *IEEE Transactions on Evolutionary Computation, 21*(5), 773-791.

Zhou, Y., Ouyang, X., & Xie, J. (2014). A discrete cuckoo search algorithm for travelling salesman problem. *International Journal of Collaborative Intelligence, 1*(1), 68-84.

Zuberi, R. H., Kai, L., & Zhengxing, Z. (2008). Design optimization of EOT crane bridge. *Eng Opt*, 192-201.