AN AEROSTRUCTURAL 3D WING OPTIMIZATION USING PARALLEL GENETIC ALGORITHMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ANIL ARPACI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

AN AEROSTRUCTURAL 3D WING OPTIMIZATION USING PARALLEL GENETIC ALGORITHMS

submitted by **ANIL ARPACI** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. Göktürk Üçoluk
Supervisor, **Computer Engineering, METU** _____

Dr. Onur Tolga Şehitoğlu
Co-supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Yusuf Sahillioğlu
Computer Engineering, METU _____

Prof. Dr. Göktürk Üçoluk
Computer Engineering, METU _____

Assoc. Prof. Dr. Tansel Dökeroğlu
Computer Engineering, TED University _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Anıl Arpacı

Signature :

ABSTRACT

AN AEROSTRUCTURAL 3D WING OPTIMIZATION USING PARALLEL GENETIC ALGORITHMS

Arpacı, Anıl
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. Göktürk Üçoluk
Co-Supervisor : Dr. Onur Tolga Şehitoğlu

September 2019, 82 pages

As a multi-disciplinary optimization problem, aerostructural shape optimization of airplane wings requires both aerodynamic and structural analysis to meet an objective defined as the sum of parameters like drag to lift ratio and wing weight, subjected to penalty of structural yield stress and geometrical sizing constraints to get aerodynamically efficient and lightweight 3D wings.

In our study, genetic algorithms are utilized for optimization of 3D wing with its internal structural components. In order to optimize an airplane wing with genetic algorithms, parametric automated geometry and mesh generator is developed. Since automation and aerostructural analysis increase the complexity of the fitness calculation, utilization of parallelism for genetic algorithms becomes crucial.

This thesis proposes an aerostructural 3D wing optimization tool using different models of parallel genetic algorithms. Moreover, the results of these models are discussed in the scope of this study.

Keywords: Geometry Parameterization, Shape Optimization, Genetic Algorithms

ÖZ

PARALEL GENETİK ALGORİTMALAR KULLANARAK 3B KANADIN AERODİNAMİK VE YAPISAL OPTİMİZASYONU

Arpacı, Anıl
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Göktürk Üçoluk
Ortak Tez Yöneticisi : Dr. Onur Tolga Şehitoğlu

Eylül 2019 , 82 sayfa

Çok disiplinli bir optimizasyon problemi olan uçak kanatlarının aerodinamik ve yapısal şekil optimizasyonu; çeşitli kırılma, gerilme, ağırlık ve geometrik kısıtlamalar uygulanarak, sürtünme-kaldırma oranı ve kanat ağırlığı gibi parametrelerin toplamı olarak tanımlanan bir amacı karşılamak için hem aerodinamik hem de yapısal analiz gerektirir. İşbu analizler sonucunda aerodinamik açıdan verimli ve yapısal olarak hafif 3B kanatlar elde edilebilir.

Genetik algoritmalar kullandığımız çalışmamızda, bir uçak kanadını eniyilemek için, parametrik ve otomatik geometri ve çözümağı üreticisi geliştirilmiştir. Otomasyon süreci ve ardından gelen aerodinamik ve yapısal analizler zindelik hesaplamasının karmaşıklığını arttırdığından, genetik algoritmalar için paralelliğin kullanılması hayati önem kazanmaktadır.

Farklı paralel genetik algoritma modelleri kullanan aerodinamik ve yapısal 3B kanat optimizasyon aracı öneren bu tez kapsamında, uygulanan farklı modellerin sonuçları da tartışılmıştır.

Anahtar Kelimeler: Geometrik Parametrizasyon, Şekil Optimizasyonu, Genetik Algoritmalar

To my family and the expected (r)evolution

ACKNOWLEDGMENTS

First and foremost, I would like to express my most sincere gratitude to Dr. Onur Tolga Şehitoğlu for his constant support, guidance and patience throughout the preparation of this study. I am deeply glad to have the chance to benefit from his invaluable experiences and insights.

I would like to give my special thanks to Dr. Erdal Oktay and EDA Engineering Design & Analysis Ltd. Co. for supporting and helping me, and allowing me to use their proprietary software modules of CAEeda^{TM}. I must also express my gratitude to Prof. Dr. Hasan U. Akay for his suggestions and comments during this study.

Studies performed in this thesis comprise a part of The Scientific and Technological Research Council of Turkey(TUBITAK) TEYDEB project (No: 3150095) named "The Code Development for Aero-structural Shape optimization and Design Automation", granted to EDA Engineering Design & Analysis Ltd. Co.

It is a pleasure to thank my past and present colleagues and friends Baler, Ramin, Fateh, Kutay, Ertuğrul, Kourosh and Berke for their help and encouragement.

I would like to thank my high school friends for their support and invaluable friendship. I always feel so lucky to have such great friends in my life. They supported(!) me so much throughout this study.

I owe my deepest and warmest thanks to my family who give me strength and encouragement throughout my studies. I am so grateful to my mother Nesibe, my father Cavit and my brother Erdem for their love and support through my whole life. And I wish to express my sincere gratitude to my love İnci for her everlasting support, encouragement, understanding and patience. This study would not have been possible without them.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AOA	Angle of Attack
PARSEC	Parametric Section
LE	Leading Edge
TE	Trailing Edge
CST	Class-Shape-Transformation
TR	Taper Ratio
GA	Genetic Algorithm
PGA	Parallel Genetic Algorithm
GPGA	Global Parallel Genetic Algorithm
CPGA	Coarse-grained Parallel Genetic Algorithm
FPGA	Fine-grained Parallel Genetic Algorithm
NURBS	Non-Uniform Rational B-Spline
CAE	Computer Aided Engineering
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
EO	Evolving Objects
FEM	Finite Element Method
BC	Boundary Condition
ADT	Alternating Digital Tree
MPI	Message Passing Interface
SIN	Search and Interpolate
SAP	Structural Analysis Program
W	Weight

CHAPTER 1

INTRODUCTION

The aerostructural shape optimization is a crucial and challenging multi-disciplinary problem that requires solutions of both aerodynamic and structural physics of the object by meeting constraints and objectives like efficiency, strength and weight. In this thesis we develop an aerostructural shape optimization tool that automatically generate and evolve wing geometries in parallel using genetic algorithms. The tool includes multiple methods of engineering and computer science like automated geometry generation, aerodynamics solution, structural solution, force transfer between meshes, genetic algorithms and parallel computation.

1.1 Motivation

Over the years, numerous wing designs have been developed and it led industry to concern more about efficiency of design and its process. Dealing with efficiency and quality of a wing has many aspects, like less fuel consumption, lightness, strength, longer lifetime, production cost, etc.

Researches on efficient wing design requires two main criteria which are aerodynamics and structural qualities of the wing. Aerodynamic design concerns the physics of the wing with air (or fluid) around its outer shape where the air flows. Structural design deals with the internal shape of wing, which implies the stiffness of it. Each of these aspects has different physical characteristics, thus two different solvers are used in different stages of the design evaluation.

The conventional method in wing design is the trial-and-error method. Human ex-

perts use common design patterns and their experience to design an initial wing, then run aerodynamic solver, analyze the result, if not satisfactory update it, when aerodynamic design is complete, try to design interiors and make sure wing is structurally correct by running structural solvers. This method has kind a heuristic approach, but it is considerably slow and inefficient process. Because it starts from a sub-optimal initial point and number of iterations a human can do is limited. Also in human design, aerodynamics and structure properties are designed as different stages. Genetic algorithm (GA) is also a heuristic search algorithm, but since it is inspired by evolution theory, it can find more efficient results than the human design. Although GA starts from a random initial state, utilizes its search history to reach to an optimum target according to its objective function. Our target in this thesis is the wing design with good aerodynamic and structural scores given by solvers. In short, aerodynamic and structural solvers are coupled with each other for using them in optimization process with genetic algorithm to reach the optimal wing design.

In order to automatically generate a candidate wing design for optimization process, we need to encode and represent it as a set of parameters. To put it simply, we can say that parameterization is a conversion from engineering problem into mathematical problem. Therefore, our wing design process turns into mathematical equations. In recent years, several parameterization methods have developed for wing design. For this study, we enumerate these methods and choose the suitable parameterization method depending of its flexibility, representational effectiveness, computational cost, etc. As the choice of parameterization technique affects optimization process, it also has an impact on the result.

When the parameterization method is decided, optimization process will be ready to be started. As long as the parameters are in reasonable intervals, parameterization and thereby optimization process will work correctly. Encoding that can generate feasible designs, there will be a better design; however, more parameter means more complexity. Also aerodynamic and structural solvers, geometry and mesh generations need the computational power. Due to the high computational cost, a parallelized optimization becomes almost a must. So, this thesis proposes a parallel aerostructural optimization with an easy and fast parameterization technique, which will be discussed comprehensively in the following sections.

1.2 Objective

In order to design an efficient wing automatically, aerostructural shape optimization has become inevitable and it includes crucial part, parametric geometry generation. Since wings are different in many aspects, and optimization process needs to be fast and accurate, parameterization technique becomes more important. There are lots of ways to parameterize a wing. In this thesis, we try to discuss different parameterization techniques and what advantages/disadvantages they have. Then, by introducing and implementing a new hybrid parameterization technique, we try to develop multidisciplinary wing design optimization tool according to aerodynamic and structural aspects with genetic algorithm. Moreover, as the fitness calculation including aerodynamic and structural analysis is a time costly process, we try to utilize parallel genetic algorithm with various parallelization models to shorten the total optimization time.

1.3 Problem Definition

As a whole, wing optimization process has many steps until the optimal wing design and it starts with parametric geometry generation. As mentioned before, to design wing easily and fast, easy and fast parameterization technique is inevitable for shape optimization. Because it is the main and the base part of the shape optimization. In the very first step of the parametric wing generation, we should parameterize and create the section of wing, namely airfoil. Then, these airfoils are used to generate a whole 3D wing geometry. So parametric generation of airfoil means that we can create a whole wing. Furthermore, we also need to generate a mesh on wing geometry, which is created by parameterized airfoil, to be used by the aerodynamic solver to calculate airflows on it. After the external shape of wing is known, we have to generate a mesh on the interior part of the wing for computing stiffness by the structural solver. Moreover, to compute stiffness correctly, calculated external aerodynamic pressures must be given to the mesh of structural solver. This mesh to mesh data transfer operation have to be done by us, so that the two solvers are combined. Furthermore, it tries to combine aerodynamic and structural approaches to optimize wing geometry with the comparison of different parallel genetic algorithm models.

1.4 Scope of the Thesis

Chapter 2 starts with the terminology on aircraft wings. Then, background information about parametric geometry, shape optimization and data transfer methods are given and the previous studies on these topics are summarized. At the end of this chapter, both aerodynamic and structural analyzes tools are presented.

Chapter 3 gives the details of developed hybrid parameterization method. Then, it explains the steps of parametric geometry generation, mesh automation processes for both analyzes and the data transfer algorithm.

Chapter 4 firstly illustrates the environment setup for the experiments conducted in this thesis. Then, it explains the optimization process of 3D wing using parallel genetic algorithms with the given details about objective function and genetic operators. Lastly, results of several experiments are shown and discussed.

Chapter 5 concludes this thesis and discusses the results of the experiments. Lastly, suggestions for the future work are given.

CHAPTER 2

LITERATURE REVIEW

In this chapter, historical background about parametric geometry generation, shape optimization and mesh to mesh data transfer methods are explained. We briefly describe how an airfoil geometry can be parameterized and generated, also what principles the shape optimization process has. Furthermore, mesh to mesh data transfer operation is explained as a computational geometry problem, then some algorithms that can solve this problem is presented. Latest researches on topics of each section are also discussed.

2.1 Parametric Geometry Generation

As mentioned before, it is necessary to represent the wing cross-sectional profile, namely airfoil, with the mathematical equation that has finite parameters. The term of airfoil is used to describe the cross-sectional shape of an object that, when move through a fluid such as air, creates an aerodynamic force. Airfoils are used on aircrafts as the cross-section of a wing to produce lift, which is needed for climbing and flying. Camber of an airfoil directly affects the lift force, since camber is a measure of the curvature of an airfoil that lies from leading edge to trailing edge, which is shown in Figure 2.1. Since camber allows to keep the air below of the airfoil, it has direct effect in the lift force of the wing.

The lift force is generated by the wing which acts perpendicular to the incoming flow and it allows an aircraft climb. The drag force is a consequence of the production of lift and acts parallel to the airflow. An example airfoil and mentioned forces on it can be seen in Figure 2.2. The straight line drawn from leading edge (front of the airfoil)

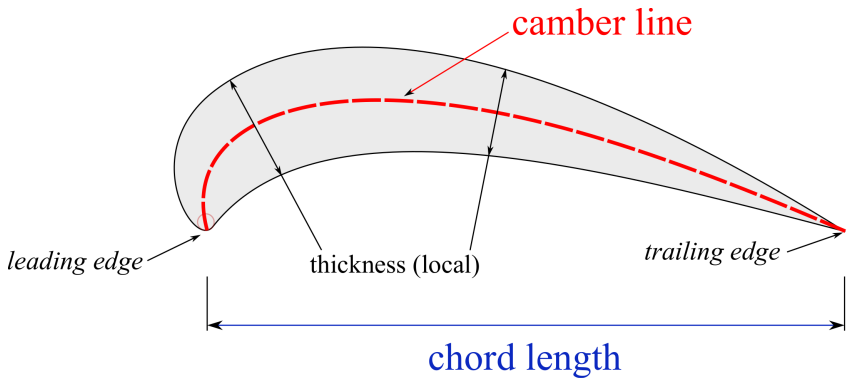


Figure 2.1: Cross section of an airfoil.

to trailing edge (back of the airfoil) of airfoil is called chord line. Angle of attack (AOA) is the angle between the oncoming air (flow direction) and a chord line of the airfoil.

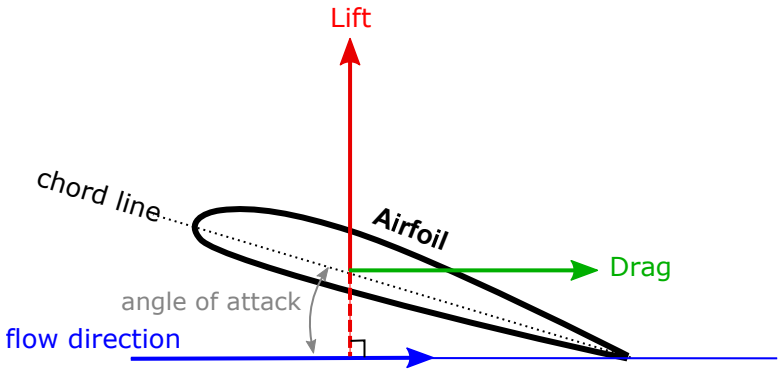


Figure 2.2: Airfoil and aerodynamic forces acting on it.

Nowadays, an airfoil can be encoded with many different parameterization methods, some of which have advantages over others. There are set of rules that have to be satisfied by the method. Some of rules are:

- Mathematical equation should represent wide range of airfoils to ensure that it covers all in the optimization process.
- It is better to have less parameters, since computational complexity and time depend on parameter count.

- In order to minimize the search space and associated complexity, output geometry should not generate invalid and/or unacceptable airfoil geometries.

While generating airfoil, XZ plane is used in the literature. X is the chord axis, Z is the thickness axis of an airfoil. And Y axis is used for the span direction, which makes 2D airfoil into 3D wing. In chord axis (X) airfoil is placed from $X = 0$ to $X = 1$, and leading side is in $X = 0$. This 0 to 1 generalization is used for non-dimensional geometry generation for any purpose in computer aided engineering (CAE) systems.

The method of airfoil generation creates the airfoil curve that is the cross-section of the wing. In order to get through from the section of wing to the whole 3D wing, lofting operation can be used. Lofting is a modeling technique that transforms from a starting section shape and orientation to an ending shape and orientation. It connects a starting shape with an ending shape to create a 3D shape as it passes through an area in space, which can be seen in Figure 2.3.

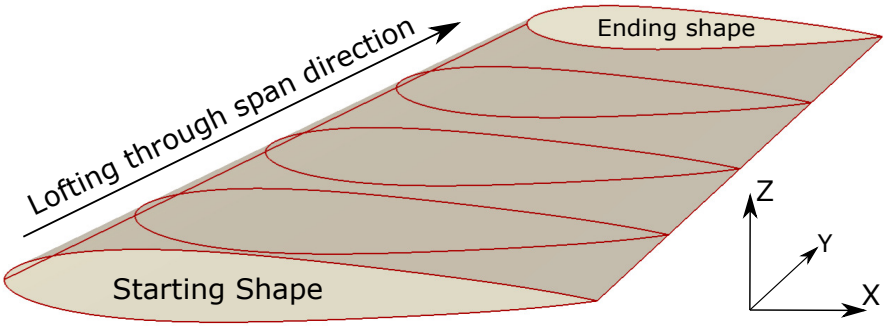


Figure 2.3: Lofting from starting shape to ending shape to create 3D wing.

There are several airfoil parameterization methods:

- Non-Uniform Rational B-Spline (NURBS) Curve,
- Bezier Curve,
- Parametric Section (PARSEC),
- Class-Shape-Transformation (CST).

2.1.1 NURBS Method

NURBS curve is the powerful extension of B-spline curve, which is used in several areas like computer aided design and manufacturing (CAD/CAM), CAE software and computer graphics [2, 3]. An airfoil can easily be represented with B-spline curve, where b-spline function is defined as a linear combination of control points and basis functions:

$$C(t) = \sum_{i=1}^{n} B_{i,k}(t) P_i \tag{2.1}$$

where P_i are the control points (see Figure 2.4) and $B_{i,k}(t)$ are scalar valued polynomials described by the order k (degree $k-1$) [4]. Then spline function, $C(t)$, forms a piecewise polynomial function of degree $k-1$ in a variable t, where t is the parameter, $a \leq t \leq b$; a and b are fixed, with $0 \leq a < b$. This non-deceasing sequence of parameters is called knot vector, $t_q|_{q=1}^{n+k}$. The pieces of B-spline function meet at the places called knots, so the parametric space is divided by the knots. Accordingly, the most significant property of B-spline function is the secured continuity of the function at the knots. Depending on whether the consecutive knots are distinct or not, derivatives of the B-spline function can also be continuous. Furthermore, if the distance between each knot is the same, then the B-spline is become uniform [5]. Rational B-splines is firstly introduced by Versprille in 1975 [6]. It is defined with the use of homogeneous coordinates. 3D control point $P_i = (x, y, z)$ becomes 4D homogeneous control point $P_i^h = (hx, hy, hz, h)$, where $h > 0$. Then, after the projection of 4D B-spline function into 3D space, reached rational B-Spline function is defined by

$$C(t) = \frac{\sum_{i=1}^{n} B_{i,k}(t) h_i P_i}{\sum_{i=1}^{n} B_{i,k}(t) h_i} \tag{2.2}$$

where h_i is the homogeneous coordinate value. Since decreasing h_i makes the curve move away from the control point P_i and increasing h_i makes the curve closer to the control point P_i, homogeneous coordinates behave like the weighting coefficients [7]. Consequently, if any of B-spline curve is non-uniform and rational, then it is called NURBS curve as a special B-spline type.

If the generated airfoil curve is in the form of basis spline (B-Spline) curve, then the whole wing can be generated easily in the form of non-uniform rational B-Spline

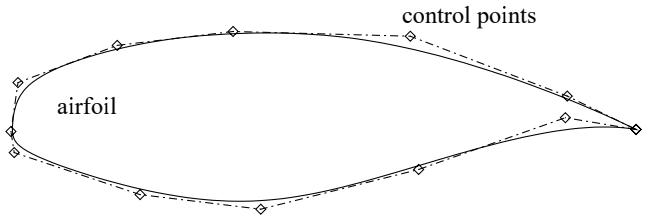


Figure 2.4: NURBS method for airfoil parameterization.

(NURBS) surface. NURBS surface function is obtained from the tensor product of two NURBS curves and is defined by

$$S(s,t) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} B_{i,k}(s)B_{j,l}(t)h_{ij}P_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} B_{i,k}(s)B_{j,l}(t)h_{ij}} \tag{2.3}$$

where P_{ij} is n times m array of control points, h_{ij} is the homogeneous coordinate value, and $B_{i,k}(s)$ and $B_{j,l}(t)$ are the basis polynomial functions with the knot vectors of $s_p|_{p=1}^{n+k}$ and $t_q|_{q=1}^{m+l}$ [8]. With the NURBS surfaces, both standard analytic and free-form surfaces can be handled efficiently. The general shape of the surface is designated by the control points, the weights of control points are determined by the homogeneous coordinates and the continuity level is set by knot vectors [9]. Since it offers fast transformations (rotation, translation), flexible and large design space, easy human interaction and less memory consumption when storing it; NURBS surfaces are commonly used by the most of modeling softwares (CAD, CAM, CAE) and became the part of many industry wide standards.

2.1.2 Bezier Curve Method

Bezier curve is a polynomial function curve, based on Bernstein polynomial. The significance and the difference of Bezier curve come from its control points, which characterize an exact continuous geometric curve [10]. Original Bezier curve with its control points P_i is defined as:

$$P(\psi) = \sum_{i=0}^{n} B_{i,n}(\psi)P_i \tag{2.4}$$

where $B_{i,n}(\psi)$ is a Bernstein polynomial with the degree of n. Bezier curve is defined as continuous curve and it is ensured by the Bernstein polynomial term. More details about Bernstein polynomial are discussed in this chapter and Section 2.1.4, where its function is given in Equation 2.8.

One of the most common airfoil parameterization technique for generating airfoil geometry is Bezier curve method [11]. In Bezier curve method, there are two functions: camber line and thickness distribution. To obtain upper and lower curves of airfoil from Bezier curve method, thickness distribution is added and subtracted to and from camber line function. Camber line function is generated by Equation 2.4. However, the main difficulty and problem in this method is to secure continuity of leading edge of airfoil. Also unknown positions of the control points may lead to irrational curves in the aerodynamic optimization process.

2.1.3 Parametric Section Method

Another common technique for parametric airfoil generation is 'PARametric SECtion' (PARSEC) method [11]. As seen in Figure 2.5, this method uses 11 main parameters to design complete airfoil geometry and these parameters linearly combined to construct a shape function, so it allows specific change in parameters. These parameters are the radius of leading edge (r_{le}), upper peak point (Z_{up} and Xup), upper peak curvature (Z_{xxup}), lower peak point (Z_{lo} and Xlo), lower peak curvature (Z_{xxlo}), trailing edge direction angle (α_{TE}), trailing edge wedge angle (β_{TE}), trailing edge thickness (ΔZ_{TE}) and trailing edge offset (Z_{TE}).

PARSEC method is highly effective for designing curvature of the curve; however this method does not have enough control to design trailing edge, which is a significant deficiency for parametric airfoil geometry generation. Since escape of air flow is on trailing edge of airfoil, it is very important to have fully control over the trailing edge for the aerodynamic calculations, and consequently the optimization process.

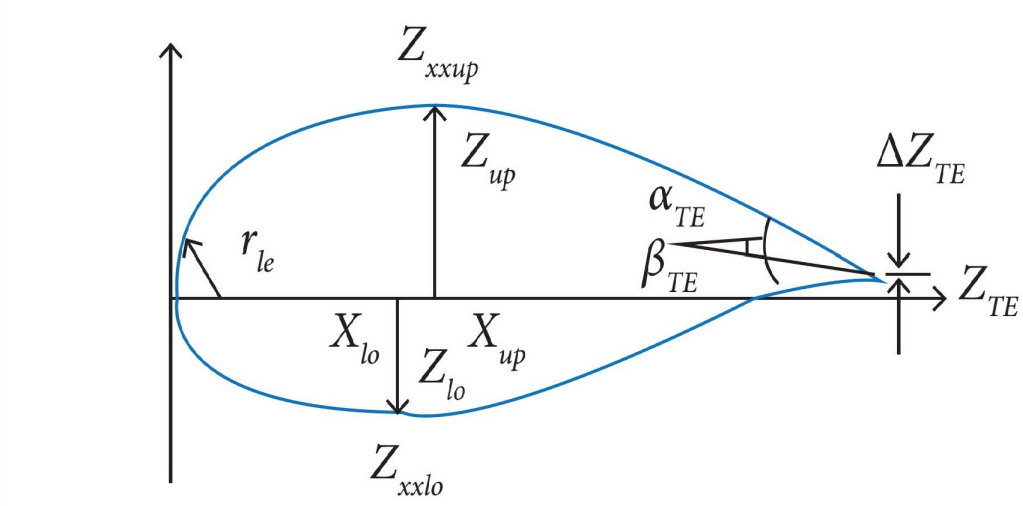


Figure 2.5: PARSEC method for airfoil parameterization. Taken from "An Implementation of Self-Organizing Maps for Airfoil Design Exploration via Multi-Objective Optimization Technique," by Jung, SungKi et al., 2016, Journal of Aerospace Technology and Management, 8(2), 193-202.

2.1.4 Class-Shape-Transformation Method

The Class-Shape-Transformation (CST) method, which was relatively newly invented by Kulfan, is allow us to design airfoil geometry with the few parameters, and it produces smooth and continuous curves [12, 13]. With this method, it is really easy to design an airfoil with leading edge continuity and to control whole structure of airfoil, including trailing edge. Actually, since this method is very general, involves 3D modeling and has a wide usage area; it becomes a milestone for designing, modeling and generating geometries. Interested readers may refer to [12, 13, 14] for different application areas of CST.

Generating airfoil geometry with CST method is the first step of this thesis. It generates airfoils with the help of polynomial functions and this advantageous feature allows us to use least square fitting method for getting CST parameters of any airfoil from literature. Since CST method guarantees the smoothness of the curve with the few equations and the continuity of the leading edge, workload of the optimization process is less than the other parameterization methods. As Lane and Marshall [14]

stated, it is similar to method of Bezier curves, but CST has added function term which classifies geometric shape into a certain category. Furthermore, Marshall [15] has also proved that a CST curve can be expressed as a Bezier curve. However, it is still hard to control the continuity of leading edge, so it means that CST method has nothing less than Bezier curve method.

General CST equation is composed of two main functions, called 'Class' and 'Shape' functions. These functions generate airfoils on XZ plane and they use x and z coordinates as design variables. To generate airfoil, it is written down as follows:

$$\zeta(\psi) = C_{N2}^{N1}(\psi) \cdot S(\psi) + \psi \cdot \Delta\zeta_T \tag{2.5}$$

where,

$$C_{N2}^{N1}(\psi) = \psi^{N1} \cdot (1-\psi)^{N2} \tag{2.6}$$

is a 'Class' function and

$$S\psi = \sum_{i=1}^{n} A_i B_{i,n}(\psi) \tag{2.7}$$

is a 'Shape' function where,

$$B_{i,n}(\psi) = K_{i,n} \cdot \psi^i \cdot (1-\psi)^{(n-i)} \tag{2.8}$$

is the Bernstein polynomial and

$$K_{i,n} = \frac{n!}{i! \cdot (n-i)!} \tag{2.9}$$

is the binomial coefficient. To use CST method in general and in any dimension,

$$\psi = \frac{x}{c} \tag{2.10}$$

is non-dimensional abscissa value and

$$\zeta = \frac{z}{c} \tag{2.11}$$

is non-dimensional ordinate value. Lastly from Equation 2.5,

$$\Delta\zeta_T = \frac{z_{TE}}{c} \tag{2.12}$$

is non-dimensional trailing edge displacement in CST method [13, 16]. As seen from Equations 2.7 and 2.8, shape function includes Bernstein polynomials to ensure that designed airfoil is smooth. Also n in Equation 2.7 is the degree of curve detail and A_i

term is the main parameter for airfoil modeling with CST method. Parametric airfoil generation with non-dimensional Equations 2.10 and 2.11 allows designer to scale airfoil anytime easily. Detailed control over the trailing edge of airfoil is handled with the Equation 2.12.

Class function generalizes Equation 2.5 to classify result into a particular category. $N1$ and $N2$ are classification parameters and to design airfoil we need to use 'airfoil' geometry class parameters. Kulfan [12, 13] stated that $N1 = 0.5$ and $N2 = 1.0$ are the parameters to generate an airfoil. The geometry for different parts of a plane, for example plane body can be generated by different class functions. Class function is the most important difference and advantage over the other geometry parameterization techniques.

Shape function of CST method allows designer to control the curve directly. Since Bernstein polynomial ensures the smoothness of the curve, geometric parameters like leading edge radius, maximum thickness, trailing edge angle can be controlled by the designer without any concerns. Consequently, designing airfoil with direct geometric parameters along the confidence on continuity and smoothness of the curve makes optimization process more robust.

All mentioned parameterization methods are summarized and compared in Table 2.1. As NURBS and Bezier methods are based on control point specifications, they become unsatisfactory when the number of geometric constraints increase. Also PARSEC method does not have fully control on trailing edge design, which is so important for the aerodynamic calculations. On the other hand, CST method does not have any design concerns like continuity, smoothness, control over curve, irrational curve generation; but it might have many parameters if its polynomial degree is higher. However, previous works [12, 17] say that 8 parameters can be enough for the representation of a relatively large set of airfoils; so we decided to use CST method with 8 parameters for airfoil parameterization.

Additionally, we decided to combine NURBS surfaces alongside the airfoil parameterization with CST, in order to get more powerful 3D wing representation. So, a hybrid CST-NURBS method was developed, because of the mentioned (in Section 2.1.1) features of NURBS on 3D modeling. Detailed information about parametric

Table 2.1: Comparison of wing geometry parameterization methods

	Leading Edge Continuity	**Parameter Count**	**Non-realistic Curve**	**Full Control over Curve**
NURBS	OK	Depends on control points	Possible	Yes
Bezier	May be fail	Depends on control points	Possible	Yes
PARSEC	OK	11	No	No
CST	OK	Depends on polynomial degree	No	Yes

3D wing generation with hybrid CST-NURBS method and its usage in this thesis are presented in Chapter 3.

2.2 Shape Optimization (Genetic Algorithm)

Shape optimization is usually a complex process that involves non-linear and non-uniform functions with hard constraints. In case of this study, shape optimization function can be evaluated as the computational result of two solver programs. As a result, it is either impossible or yet unknown to solve such problems analytically. Instead, evolutionary heuristic methods exploring the search space to reach an acceptable sub-optimum are used. Genetic algorithms is one of such evolutionary methods. GA was firstly introduced by Holland [18] as a theory, but Bagley's work [19] was the first using the Genetic Algorithm term. Computer science implementation of GA also includes mutation, crossover and survival of fittest processes like evolution by natural selection process [20].

Evolution is achieved by finding the best parameters for individuals with genetic operators. GA determines the best individual with the calculation of the fitness score for each individual according to the encoded solution. An individual having a higher fitness score means it is better and also, better individual has more chance to transfer its genes to the following generations. The priority of better individuals increases the

chance to reproduce high fit members in the following generations. Identically, low fit members will become rare in the next generations, since they have less chance to transfer their parameters.

A parameterization method converting real life object to a set of parameters is needed to encode a solution. Set of parameters for representing that solution is called its "genotype" and its real life counterpart is called "phenotype". Each parameter from genotype set is called a gene and the set of parameters is called a chromosome, a group of genes. In GA, traditionally the chromosome of individual is a vector that has finite number of parameters. For example, a solution can be encoded by 8 integer values as parameters. From these parameters, a function, called fitness function, returns a single value for evaluating the success of the individual. The selection process of GA is carried out according to fitness function; therefore the survival of fittest rule of the natural selection process is applied to GA.

As the initialization, GA starts with the population that has random individuals. In the successive generations, latter population should be reproduced from former with the help of the genetic operators: selection, variation (mutation and crossover) and replacement. Initially, individuals of starting generation are randomly created by using given parameter intervals; then, fitness values of that generation is calculated. Better individuals are chosen from the population based on their fitness values by selection operator and fed to the variation operators. Without any alteration and variation, evolution throughout next generations continues with the same genetic heritage, which causes to stuck on the local optimum where all individuals are same with the best individual of the initial population. In order to explore the search space while keeping genetic heritage of the better individuals, we need to use variation operations crossover and mutation. These operators are applied in the reproduction step with the preassigned probability for individuals. The reproduction step is done by the replacement operator of GA, which is the borderline between the current and the next generations. When all individuals are replaced with the new ones, transition between the generations is completed. Then GA cycle continues with the calculation of the fitness values of the new generation. The pseudo-code of the GA algorithm is stated at 1 and the general flow of the GA process can be seen in Figure 2.6.

Algorithm 1 Pseudo-code of GA algorithm

procedure GENETICALGORITHM(Pop)
    $t = 0$ ▷ Generation number
    InitializePopulation($Pop(t)$) ▷ Sets initial random population
    EvaluatePopulation($Pop(t)$) ▷ Evaluates initial population
    while !StopCriteria($t, Pop(t)$) **do** ▷ Reproduction loop
        $Parents$ = Selection($Pop(t)$) ▷ Selects individuals for reproduction
        $Children$ = Crossover($Parents$)
        $Children$ = Mutation($Children$)
        EvaluatePopulation($Children$)
        $Pop(t+1)$ = Replacement($Children, Pop(t)$) ▷ Moving to the next generation
        $t = t + 1$
    end while
end procedure

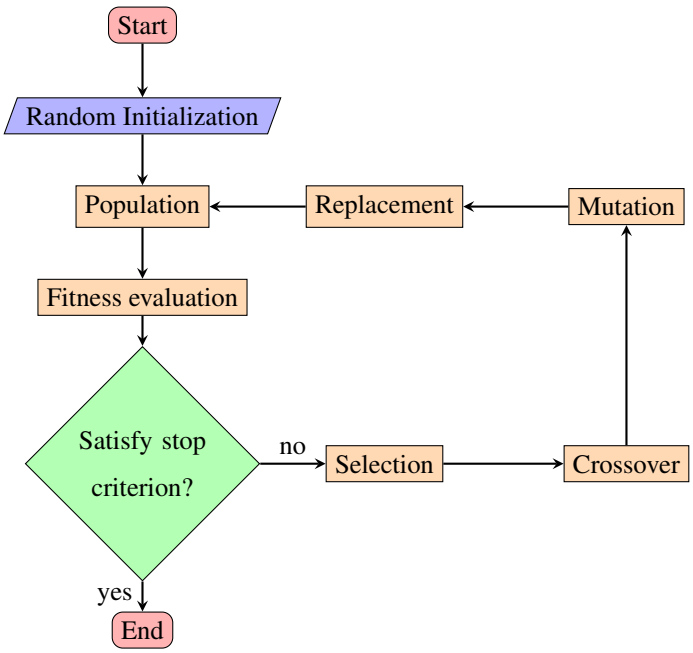


Figure 2.6: The flow chart of the heuristic search with the GA.

The search process of GA has typical generation cycle:

- Individuals are selected according to their fitness value,

- Crossover is applied to chromosomes of the selected individuals,
- Mutation is applied to genes after crossover with some probability,
- Fitness values of the new population are calculated according to fitness function,
- New population takes the place of the old population by using replacement,
- Generation cycle starts from the beginning.

In every generation cycle, it is expected that newly generated populations get closer to the optimum solution. If genetic operators are applied appropriately, better gene characteristics are transferred to the next generations, so fitness values of individuals get close to the optimum value. Generation cycle continues until the convergence is reached. The convergence is decided with the given stopping criteria, like setting the maximum generation number or defining a certain tolerance change between the most optimum consecutive solutions throughout the generations. In most cases, maximum generation number is used for the stopping criteria, due to the difficulty in the prediction of the tolerance change. Genetic algorithms do not guarantee the optimum solution but converges to one of the acceptable sub-optimum.

2.2.1 Selection

As mentioned before, individuals are selected from the population to be the parents of the next generation. To create better generations, GA needs to select better individuals, so that the best chromosomes can pass their lineage to their successors. Selection operator forms a mating pool with the selected good individuals that take place in the variation phase to create better individuals. There are different selection methods in genetic algorithms literature like fitness proportionate selection, ranking selection, steady state selection, tournament selection, etc.

In fitness proportionate selection (also known as roulette wheel selection) [21], calculated fitness of an individual is directly used in the probability value of the selection

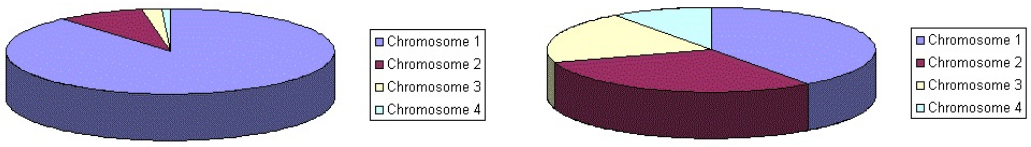
operator. The selection probability of an individual i is calculated as follows:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2.13}$$

where, f stands for the fitness values and N is the number of individuals in the population.

Probability of the selection gets relatively higher proportional to the fitness value of an individual, so an individual with the higher fitness value is less likely to be eliminated. However, if its fitness is extremely higher than the others, there is a possibility of selection almost 100%. This selection operator gives individuals with lower fitness far less chance to be selected and it may cause premature convergence, because the fittest individual is highly dominant over the others [22]. This problem is called selection pressure and can be controlled by adding ranking coefficient to the fitness, which is called rank selection [23]. In ranking selection the selection probability will depend on the rank instead of the pure fitness value. As a result, probabilities will be more reasonable as depicted in Figure 2.7.

Even though all the individuals have a chance to be selected compared to the fitness proportionate selection, rank selection may cause a slower convergence, due to the minimal difference between the best and the worst individuals.



a) b)

Figure 2.7: Selection probabilities: a) in fitness proportionate selection. b) in ranking selection

In another method, only nest n individuals are selected and it is called truncation selection [24]. Population in descending order of fitness values is truncated with the predefined proportion and retained individuals are duplicated to maintain the population size. Truncation selection is a very simple selection method, but it may lead premature convergence, because the diversity from weak individuals will decrease

over the generations. So this selection method is not frequently used in practice, except some group of problems [25].

As the name suggests, tournament selection chooses individuals by running tournaments among several random individuals. It was firstly studied in Brindle's dissertation [26] and there are many recent works using elimination design like tournament selection [27, 28, 29]. Among randomly selected individuals, the winner of the tournaments is decided by their fitness values. In this selection method, tournament size is one of the main design concern for controlling fitness pressure. If the tournament size is chosen too large, less fit individuals are most likely eliminated and it causes loss of diversity over generations. However, if the tournament size is chosen too small, generations preserve the diversity with a decrease in convergence speed. So tournament size is really an important design choice and it is usually set depending on the optimization problem. This selection method has many advantages over the other selection methods: it can be parallelized for a certain architectures, tournament size can be changed easily in the middle of optimization process, it has faster run-time [30] and it is efficient to code.

2.2.2 Variation: Crossover & Mutation

Crossover is simply a recombination operator used for finding a better individual. It gets different parts of two chromosomes from the previous population and recombines them to variate species throughout the generations. It generates new individuals by combining its ancestors' chromosomes, so crossover makes individuals exchange their genes among themselves [31]. The ancestor individuals are called parents and the reproduced individuals are known as children.

It is clear that the crossover operator is used mainly for the search of the new individuals by using two individuals. This may cause worst gene recombination, still it may increase the fitness of chromosome. Because in the next step, reproduced better individuals transfer their genes more frequently to the next generation. Nevertheless, it is predictable that crossover operator might be as detrimental as can be beneficial. For this reason, it is not applied to all individuals and there is a probability factor (p_c) to decide whether to apply crossover or not [32].

As previously described, a mating pool is formed in the selection phase and it has better individuals which are inserted by selection operator. However, there is no newly created individuals in the selection phase and in the pool. As mentioned before, crossover operator makes use of that mating pool to create new individuals and variation across the generations. It is done by exchanging chromosomes among the individuals from the mating pool with the probability p_c.

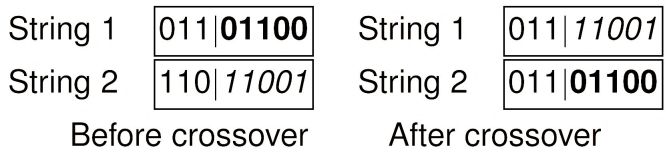


Figure 2.8: One-point crossover operation

Crossover can be applied in several ways. In one-point (single-point) crossover, two different individuals are randomly taken. Then, their chromosomes are divided from arbitrary one point and they are cut at that point. As seen in Figure 2.8, there are head and tail parts of each chromosome and they exchange their tail parts to produce two new individuals of new generation. In two-point crossover, two different individuals are taken again. But their chromosomes are divided from arbitrary two points. As represented in Figure 2.9, the parts between chosen two point are exchanged between two chromosomes. Chromosomes can be cut more than two points, which is called n-point crossover that might improve the efficiency of operator in some optimization problems having too many parameters [33]. The crossover methods behave differently according to both population size and parameter size [34].

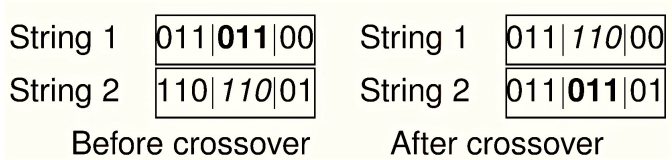


Figure 2.9: Two-point crossover operation

If the chromosomes contain non-binary genes, an arithmetic crossover operator can be applied [35]. It linearly combines two parent chromosome vectors to produce two new individuals that are weighted arithmetic mean of two parents, where the operator is applied according to the following equations:

$$Individual_1 = a \cdot Parent_1 + (1 - a) \cdot Parent_2 \tag{2.14}$$

$$Individual_2 = (1 - a) \cdot Parent_1 + a \cdot Parent_2 \tag{2.15}$$

where a is a random coefficient between [0, 1] and it is chosen before each crossover operation.

As an another genetic variation operator, mutation provides randomness to search process of GA in the search space. While crossover generates individuals directly from its parents, mutation operates at gene level, as depicted in Figure 2.10. It can be said that crossover tries to find better ones by looking on the current solution, where mutation explores the whole search space. If population sticks on local optima, mutation helps GA to reach the global optima [31]. This effect may not be seen in the next generation directly, but reaching beneficial genetic codes in the search process supports following generations.

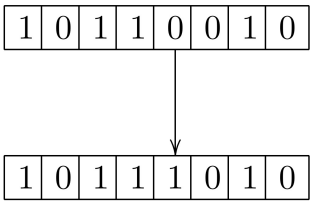


Figure 2.10: Mutation in the fifth gene of chromosome

Similar to crossover operator, mutation operator also has probability to decide whether gene is mutated or not. It changes (mutates) an arbitrary gene with a preassigned probability p_m. In general, mutation probability is assigned quite small values, since higher mutation probability may cause divergence in the search process [32]. A high value of mutation probability turns heuristic search into random search; nonetheless,

the minor and random diversity (low mutation probability) can end up with better solution, which is the main purpose of using mutation.

Since mutation operator allows GA to jump to another region on multidimensional search space by avoiding concentration on local optima, evolution with GA can continue without any stuck or slowdown. For example, if individuals in the population do not have any 0s in the left most bit positions, then without mutation it is impossible to create 0 in that position. Also global optima solution may require 0 there, but neither crossover nor selection operator can create 0 in the left most bit position for that example. Only mutation operator is capable of turning 1 into 0, and of course with a probability p_m.

In the literature, several mutation techniques are employed for the different representations of parameters [36]. If parameters are binary, then bit flipping is one of the most used mutation operator. As it is shown in Figure 2.10, mutation operator flips the fifth gene 0 to 1. If the parameters of chromosomes are float or integer, then uniform mutation can be given as an example to mutate the non-binary parameters. It changes the value of random gene with uniform random value between the given upper and lower bounds for that gene. The bounds can be specified for each gene separately.

Both genetic operators, crossover and mutation, have different missions in the search process as a variation operators of genetic optimization process. Crossover just looks on the local search space and mutation explores the other regions. Most of the time, mutation is applied just after the application of crossover. Thus, combination of these two genetic operators increases efficiency of reproduction of population while advancing through the generations.

2.2.3 Replacement

GA uses selection operator to choose individuals as an input for crossover operator and then mutation is applied to enlarge the search space. After the selection and variation operators, GA needs to evolve population continuously, so genetic inheritance should be passed between the sequential generations. Therefore, replacement

operator stands for the transmission of the genetic material from one generation to the next. When new generation is ready, replacement has to decide which of newly generated individuals would be replaced with which individual of current generation [31]. "Survival of the Fittest" principle of Darwin is the main idea behind the replacement operator. According to this principle, more chances are given to better individuals to survive and carry their better chromosomes to the next generations. There are several types of replacement operators, such as generational replacement, steady state replacement, elitism, etc.

Generational replacement replaces all population at each generation. Newly created individuals continue to the next generation and they are replaced with all the current individuals [37]. In this type of replacement operator, there is no overlapping individuals between two consecutive generations.

In steady state replacement, replacement is done with predefined very small proportion of the population. Newly created individuals are moved directly to the next generation [38]. As a benefit of this type of replacement, better individuals can participate in genetic search process, since creation of the whole generation would not be waited by that better individual. So the participation of better individual also means that the worst individual is removed on behalf of better one. Although there can be a quick advancements in the fitness of the population, steady state replacement may cause premature convergence due to the centralization of the fittest individual [22].

Whereas elitism might not be counted as a type of replacement operator directly, it is mostly defined within the scope of the replacement operators in the literature. It is kind of a private permission for an individual to proceed to the next generation without passing through replacement operator and without any modification. Since variation operators may ruin the chromosomes of the fittest member of the population, elitism preserves one or few best individuals from the crossover and mutation. Therefore, better genes can certainly be saved for future generations and there is no possibility to lose them due to the stochastic errors. As a result of keeping the best one, GA does not waste time to rediscover previously found partial solutions. In general, several studies [39, 40, 41] show that elitism improves the performance of the GA.

2.2.4 Island Model

In a traditional GA, population of next generation is created by previous population that has the same genetic information. In order to create divergence, mutation might not be enough. After some generations, fitness distribution of generations becomes similar and this may lead a convergence on the local minimal. To avoid this, Grosso [42] subdivided one large population into interacting many sub-populations and it improved the performance of the GA. In the island model, these sub-populations are the individuals of the different islands. Besides mutation, genetic diversity is also preserved in these sub-populations. Because each island can potentially search around a different sub-area of the search space [43]. Furthermore, different genetic operators and/or rates (mutation, crossover, selection etc.) can be used in different islands, which allows varied exploration in one genetic run.

Interaction between the islands is determined with three new genetic parameters: migration interval, migration rate and migration topology [44, 45]. Migration interval is the number of generation count for the next migration, and migration rate is the number of individuals to migrate to a selected island. And the migration, communication between islands, is done through the selected topology [46]. One of the most common and basic migration topology is the ring topology. Each island interacts with its left and right neighbors only: one for individual sending and one for individual accepting. However, in a complete network topology, each island is connected to the all other islands, where migration is done randomly. There are also different network topologies like 2D/3D mesh, cube/hyper-cube, which can be used for the migration topology.

Since one of the most important advantage of the island model is its natural structure of heterogeneity, migration parameters must be set rationally to not ruin the diversity. In the literature, studies [47, 48, 49] showed that using island model can improve the search speed and accuracy. Moreover, in GA optimization with only one population, it is showed that increasing population size causes negative consequences on search time. However, creating sub-populations (islands) from one bigger population sustains genetic diversity and these created islands exploit different search space with a different genetic material. In order to combine different explorations and solutions,

migration should occur with the right timing. If it occurs too often, it proceeds like it has single population. Nonetheless, migration interval needs to be set to migrate often enough to internalize other search trajectories.

As all the sub-populations try to find global optima from the same heuristic search, genetic information should be shared between the islands with the migration. So, GA maintains some independence in islands, while sharing information between them. For keeping the balance between diversity and information sharing, migration rate -how many individuals are migrated- is also set carefully. If the rate is too big, the genetic diversity is negatively affected. And if the rate is too small, genetic search process cannot gain any advantage from the information sharing. In the literature, it is said that the migration rate should be between 10% and 20% of sub-population, in order to improve the convergence speed and solution quality [46].

2.2.5 Parallelism

GA can be applied to many real-world optimization problems. Since most of the real-world applications of GA has time-intensive fitness function and large search space, it needs to be executed on the high-performance computer. After the rapid development of technology and science, researches and development of parallelism on GA becomes inevitable. Due to its inner search mechanism, GA is naturally suitable for parallel processing. Also together with the high-performance cores, parallel genetic algorithms (PGA) solve complicated real-world problems efficiently. Hence, PGA has received much attention and a lot of methods for parallelism are developed [45, 50, 20, 51, 52].

In global PGA (GPGA), parallelism is done with master and slaves; so it is also called master-slave model. There are two different models for GPGA: synchronous and asynchronous [52]. In synchronous model, only the fitness function is calculated in slave processors; but other genetic operators (selection, crossover, mutation, replacement) are done in the master processor. So, population is kept in master processor and computational load is distributed among slaves. It is called synchronous, since slave processors have to wait until all processing of current generation to be done to move on to the next generation [20]. However, in asynchronous model, genetic operators

are also done in the slave processors, so a clustering is needed for that model. Master is responsible for generating clusters and choosing optimum solution from end of the slave calculations. As an optional method; migration between the slave clusters can be applied. In a certain interval, a certain number of individuals from slaves can be migrated to another one [53]. Asynchronous GPGA is the centralized version of the island model.

Coarse-grained PGA (CPGA) is also called distributional model, where each processor has its own sub-population that includes more than one individual [50, 52]. In the literature, this model is also known as island model, since each sub-population can be considered as an island, of which model is described in the previous subsection. Applications and methods of CPGAs are the same as an island model, but island model can also be executed sequentially.

On the other hand, each processor has only one individual in the fine-grained PGA (FPGA) model, which is also called neighborhood model. Just as the importance of island topology for the island model, neighborhood structure is critical for the search process of FPGA. There is no guaranteed neighborhood structure for the success of FPGA, since it usually varies according to problem type [52]. In FPGA model, population is distributed over processors of a 2D mesh [53]. Each processor communicates with the other processors within the neighborhood radius, where the smaller radius is chosen for keeping the diversity and preventing the premature convergence [46].

2.3 Data Transfer (Geometric Search and Interpolation)

3D wing is solved with two different solvers and they use different meshes on the one geometry. When aerodynamic solution is done, we need to transfer aerodynamic pressures to structural solver. Meshes can be triangular/quadrilateral, fine/coarse; so these two meshes can be completely different, but their geometry is same. So we need to use 3D geometric search algorithms and then interpolate data from one to other. In our case, to transfer data from aerodynamic mesh (source) to structural mesh (target), nearest aerodynamic shell element should be found for each structural vertex. This is more than just geometric searching, it is also a geometric intersection problem. For

this reason, interpolation can be done for every target vertex only after the intersected source shell element is found for them.

In the literature, many spatial tree structures were used for similar search problems. Most of them depends on space partitioning method [54]. It recursively divides the whole scene into two or more parts until the partitioning is enough. k-d tree, Binary Space Partitioning (BSP) tree, Alternating Digital Tree (ADT), quadtree, octree can be given as examples of space partitioning trees. Some of them can only work in two dimensional space, while some of them can work in three or more dimensional space.

All mentioned tree structures propose a good solution for geometric searching and their average computational complexity is proportional to log(n). Nonetheless, if the data distribution is not balanced, time complexity gets larger and tree becomes inefficient. For our case, any tree structure that supports 3D space can be used to solve our searching problem.

2.3.1 Octree

One of the most popular data structure for 3D space geometric searching is an octree. It is used in many areas, such as image processing, computer graphics, computational geometry, geographic information systems etc. First usage of octree was put through in 1980 by Meagher [55, 56]. Octree divides its volume space into eight equal subspaces, because the space is three dimensional. Decomposition of octree is done recursively and level by level in a hierarchical way. In octree, there is only one root, but each parent has at most eight child nodes, so it is not a binary tree structure. In the same level of tree, every node, called octant, has same block volume [57, 1]. An example of octree can be seen in Figure 2.11.

Using octree for geometric searching has $O(\log_8(N))$ average time complexity. But unbalanced data distribution negatively affects the efficiency of tree. In the worst case scenario, time complexity of searching becomes O(N), which is exactly equal to linear search [58].

In searching process, while traversing from root to leaves, the search area is eliminated. But the resulting leaf is not necessarily the nearest node. It is therefore nec-

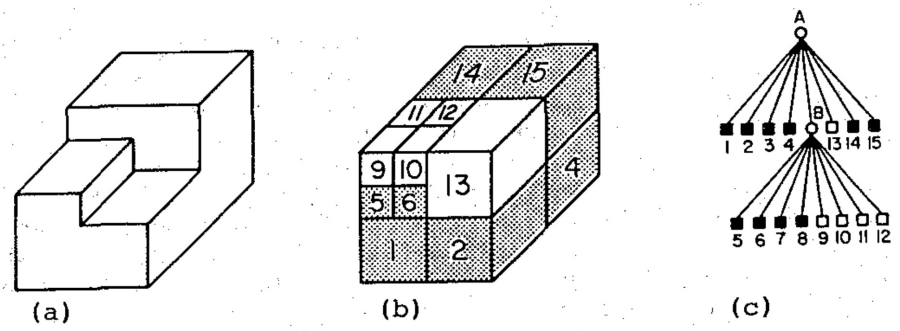


Figure 2.11: (a) Three dimensional object; (b) its octree block representation; and (c) its tree representation. Taken from [1].

essary to check the neighboring octants for any nearest node. Moreover, if there is any, checking for another is required. This process is done recursively until there is no any nearest node [59]. In literature, there are several methods to minimize the cost of checking neighbor octants: hashing of octants [60], giving an identification number to octants [61] etc. Consequently, the cost of geometric search process is still proportional to log(N) in average.

2.3.2 Alternating Digital Tree (ADT)

ADT is a general data structure that can solve geometric searching problem in any number of dimensions. It is found by Bonet in 1991 [62] to propose a solution both for geometric searching and intersection. As Bonet expressed that, it is an enhanced version of the digital tree search technique, which is applied to the one dimensional problems by Knuth [63].

While decomposing 3D search space, ADT recursively alternates the axis in each level of division, to create hierarchical tree structure. Due to the fact that ADT is a binary tree, its parent nodes can have at most two child nodes, called left and right sons. Each node in a tree divides its region into two equal subregions. First bisection is done across the first axis of spatial coordinates, where the division is made from the half of the maximum value of first axis. As a repetition, second division is done across the second axis of spatial coordinates and third is done across the third axis.

Cyclic axis selection in alternating order continues repeatedly, until there are no more nodes. After generating tree with the all given nodes, an association between the data and 3D subregions is very similar to octree [64]. Since ADT is a binary tree and always divides the space into two parts, its subregions are less, but the depth level of tree is more than the depth of octree.

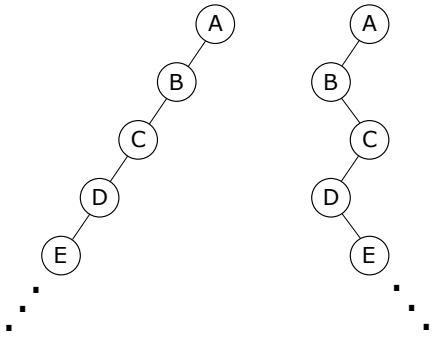


Figure 2.12: Degenerated ADT structures.

Another similarity between octree and ADT is computational complexity of searching algorithms with the trees. Since their construction approach are similar, their searching process is also analogous. So, average computational time complexity of geometric searching with ADT is O(log(n)) [65]. However, efficiency of tree mainly depends on spatial distribution of data, like octree. And if the data distribution is not balanced, the corresponding tree structure might be degenerated as in Figure 2.12. Elimination in geometric search process can not be done with degenerated trees and thus, the cost of search process becomes equal to a linear search. By looking their computational complexities and construction processes, it can be said that there is no important difference between ADT and octree.

2.3.3 k-d tree

As a multidimensional binary search tree, k-d tree was presented by Bentley for associative searching [66]. With the help of its unique characteristics, the tree structure can be used for any dimensional geometric search problems. But the implementation of k-d tree is more complicated than previously explained tree structures.

The space is split by axis of nodes in an alternating order. Division axis is changed one by one, as ADT does. Also the creation process of tree is same with the ADT. Nevertheless, the difference is that ADT divides the space into two equal regions, but k-d tree divides from nodes and it can divide from anywhere. Accordingly, balance of the k-d tree is more likely to depend on data insertion order than data distribution.

Computational complexity of geometric search operation with k-d tree is O(log(n)) in average, like octree and ADT. As the other trees, efficiency of k-d tree is also affected by whether it is balanced or not. Nonetheless, data distribution is not the only variable that can affect the balance of the tree; as mentioned before, data insertion order is an another important variable. Regardless of the effect of that variable on balance, the search process on tree is very similar with octree. Optimization applications can be done to minimize the searching cost, just like octree. To summarize, k-d tree is analogous with ADT in creation process, and is similar with octree in searching process.

2.4 Analysis Tools

In the search process of better wing design, it needs to be decided whether the design is better or not. As an analysis tool, solver calculates forces according to given model. In this thesis, two solvers are used, aerodynamic and structural. While defining fitness function of GA, results of these solvers are used. Thus, GA can choose which design is better.

2.4.1 Aerodynamic Solver

Aerodynamic analysis is simply the calculation of forces and motion of air when the solid object moves in the air. Since the solid object can be an aircraft or a wing, it is a wing in our work. There are many aerodynamic analysis tools and methods in the literature. PanAir is an aerodynamic solver released by NASA [67]. It uses panel method to analyze subsonic and supersonic flows with solving linear partial differential equations.

PanAir takes the input solid model as partitioned networks of surfaces. In the other words, the input contains quadrilateral surface meshes of networks. User input includes these networks of model and their boundary conditions. The coordinates and connectivities of mesh of networks, boundary conditions of each network and flow characteristics must be supplied by the user in the input. Mach number and angle of attack can be given as examples of flow characteristics.

From given inputs, PanAir calculates pressures, forces and moments. While doing that, numerous flow quantities are calculated as an intermediate step. Skin pressure coefficients, velocities and local mach number are some of flow quantities computed at each network points. Then, combination of these results composes lift and induced drag coefficients, axial forces and moments in the final output.

Basically, in addition to induced drag, total drag coefficient has also *friction drag* component [68]. Induced drag is an inevitable result of the lift. But the friction drag is the resistance force applied to the wing moving in an air and panel method can not calculate friction drag [69]. In order to get more realistic drag coefficient, friction drag must be calculated.

Friction drag can be calculated by solving boundary layer equations, which are the equations to compute forces on the bounding surface of the object where the viscosity is significant [70]. The aerodynamic analysis tool named XFOIL [71] can solve boundary layer equations to calculate the friction drag. XFOIL takes mach number, angle of attack, the points of airfoil with skin pressure coefficients as the input. As XFOIL is a 2D solver, sample airfoils across the span are given to XFOIL to calculate 3D friction drag coefficient. Since 3D effects are so important for a realistic solution, pressure coefficients calculated in PanAir are also given to XFOIL. By using these inputs, calculated friction drags for each sample airfoil are combined to result total friction drag for 3D wing. Then, the total friction drag is added to induced drag to calculate total lift coefficient (C_L) that include both PanAir and XFOIL drag results.

Drag and lift coefficients (C_D, C_L) are typical dimensionless values used in aircraft design. Drag coefficient is a resistance of an object in the air while lift coefficient is a lifting of an object through the air. Lower C_D with higher C_L directly means the better fuel economy and climb performance. By looking the meaning of c_D and

C_L, it can be said that C_L/C_D is very useful ratio for understanding the performance and efficiency of object. In the literature, this ratio is used very frequently to determine aerodynamic performance of an object, so in this thesis, the fitness function of aerodynamic optimization process is formed from a ratio related with both C_L and C_D.

2.4.2 Structural Solver

Internal analysis of the solid object is handled with structural analysis. The solid object can be in moving or stable (non-moving) state. Analysis calculates internal forces, stress and deflection of a physical object under applied load condition. According to computation method, there are many approaches and structural analysis tools in the literature. One of the most common method is the finite element method (FEM). SAPeda [72] is a structural solver that uses FEM. It is developed by Dr. Erdal Oktay and Prof. Dr. Hasan U. Akay and used in EDA Co. for structural problems and there are several published studies using it [73, 74].

The input of SAPeda contains triangular shell mesh of all of the solid including the internal parts. Whole mesh information, node coordinates and connectivities, and boundary conditions of all components of the model should be specified one by one in the user input. Since the analysis is conducted according to structural aspect, user must also define the thicknesses of all parts of the model. Furthermore, initial external conditions, such as temperature, density, altitude, velocity etc. and any external structural load have to be defined in the input by user.

Results of the SAPeda analysis includes computed displacements and von-Mises stresses, which is a value used to determine whether the design will withstand a given load or not [75]. These results are calculated for each shell element and also for the whole model. Whereas these calculations are computed using FEM by defining stiffness of each component as a matrix, the analysis includes matrix algebra that has computation of partial differential equations. At the end of FEM calculations, resulting stresses and displacements are used for optimization processes in general [76]. Accordingly, output value of both stress and displacement are used to compose the fitness function of structural optimization process in this thesis.

CHAPTER 3

CONTRIBUTIONS

3.1 Airfoil Parameterization with CST Method and Generation of 3D Wing in NURBS Surface Form

As explained in the previous chapter, there are many existing methods to generate parametric airfoil geometry. All of them satisfy the general rules and generate an airfoil. But there are minor differences and some advantageous/disadvantageous cases between them. In Bezier curve method, it is hard to assure the continuity of leading edge of airfoil. Also in PARSEC method, the control on the trailing edge design is less sufficient than the other airfoil parameterization methods. On the other hand, CST method proposes full control over the airfoil curve with the help of Bernstein polynomials and its hybrid structure that includes shape and class functions. So we decided to implement CST method to generate airfoil in our work.

CST method is used to create 2D profile of a wing, an airfoil. But to pass from an airfoil to 3D wing, an operation like *lofting* is required. The process of creation of the designed wing has three main steps:

1. CST method generates the points of the airfoil curve from a set of parameters,
2. Generated points are used to create a NURBS curve.
3. NURBS curve of airfoil is lofted through the Y axis to create 3D wing in a NURBS surface form by considering span length, taper ratio and sweep angle.

Lofting operation is simply to extend the curve with the same shape through the given path. For a 2D airfoil aligned in X-Z axis, this will give a surface with all same points

of the airfoil but Y dimension changes. In 3D wing geometry creation, generated base airfoil is lofted through spanwise direction to the tip airfoil. In the first step above, our CST implementation produces an airfoil that has one unit chord length.

After CST implementation, use of NURBS curve and surface forms is handled with the help of Open Cascade framework [77]. It allows us to create two NURBS curves for upper and lower side of airfoil from points that are generated by CST. Lastly, these curves are lofted to compose 3D wing in the NURBS surface form with the inputs of desired span length, taper ratio and sweep angle.

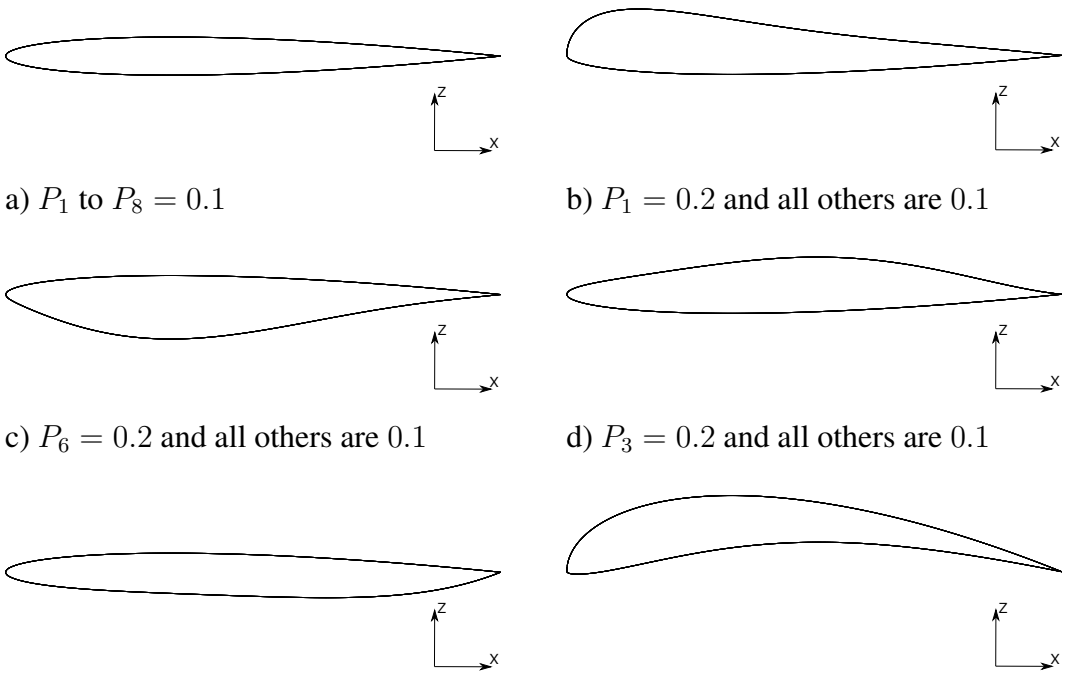
In a NURBS surface form, the generated wing will be a well composed smooth and continuous surface model. As expressed in the previous chapter, its mathematical background allows us to design free-form surfaces flexibly without any concerns about continuity of surface. It offers fast modeling with the needed precision in design and less memory consumption to keep the design. Furthermore, NURBS surface form is derived with the philosophy of engineering concepts, and facilitates the mesh generation process using the OpenCascade framework, which does not have CST implementation. Thus, it is possible to model a surface with small patches to ease modeling process and to enlarge the design space. Also the continuity of surface does not make any trouble to the designer, although the surface is patched. However, the designer has to choose the patch areas carefully, to be sure about the continuity of the whole surface.

Since airfoil model that is generated by CST method, consists of two curves, one upper and one lower, the resulting 3D wing geometry also consists of two NURBS surfaces. These surfaces can be assumed as patches of wing, so it must be certain that the continuity on the common edges of patches is maintained. Whereas two surfaces is placed in upper and lower, there are two common edges, the leading and trailing edges. In the leading edge of the wing, at least G^1 (tangent continuity) must be ensured; but since airfoil shape has a corner at the trailing part, G0 (position continuity) will be required.

Beside the two NURBS surfaces of the wing, there are also two planar surfaces: one defines base and the second defines tip of the wing. These planar surfaces are in the shape of parametrically designed airfoil. Furthermore, it needs to be ensured that

there is G^0 continuity between the NURBS surfaces and these planar surfaces. So we also define planar surfaces as the NURBS surfaces for the uniform representation of the whole wing.

Our implementation has eight parameters to generate an airfoil with CST method. Initial four parameters (P_1, P_2, P_3, P_4) are used to generate the upper curve and latter fours (P_5, P_6, P_7, P_8) are used to generate lower curve of the airfoil. As easily predicted from its equation, parameters P_1 and P_5 are responsible for the leading edge of airfoil. And also parameters P_4 and P_8 are responsible for the trailing edge. Effects of parameters on airfoil shape can be observed in Figure 3.1.



a) P_1 to $P_8 = 0.1$

b) $P_1 = 0.2$ and all others are 0.1

c) $P_6 = 0.2$ and all others are 0.1

d) $P_3 = 0.2$ and all others are 0.1

e) $P_8 = 0.2$ and all others are 0.1

f) P_1 to $P_4 = 0.4$, $P_5 = 0.05$ and all others are -0.2

Figure 3.1: CST airfoils generated for different parameter values

As this thesis tries to optimize 3D wing geometry, we need three additional parameters to advance from 2D airfoil to 3D wing: span length, taper ratio and sweep angle. Taper ratio (TR) is defined as:

$$TR = \frac{Chord_{Tip}}{Chord_{Base}} \tag{3.1}$$

where $Chord_{Tip}$ is the chord length of the tip airfoil and $Chord_{Base}$ is the chord length of the base airfoil. All mentioned parameters can be seen in Figure 3.2.

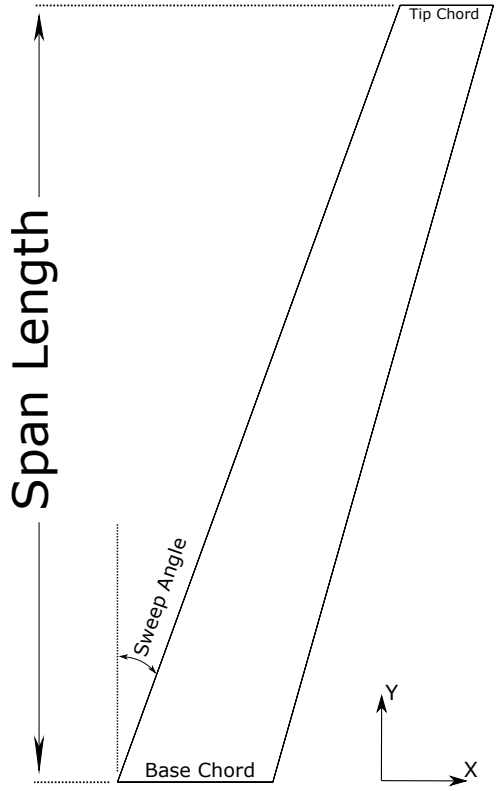


Figure 3.2: Top view of the wing.

3.2 Mesh Automation for both Aerodynamic and Structural Solver

Generated 3D wing cannot be recognized by solvers in its raw solid model. In order to conduct an aerodynamic or a structural analysis, solid model has to be described in such a way that solvers can identify and process it, so we need to create a mesh of the whole solid model. By creating a mesh, the domain is broken up into pieces and each piece represents an element. If the domain is surface, then pieces are shell elements, which can be triangle and quadrilateral. If the domain is solid, then pieces are volume elements that are formed by shell elements. In this thesis, shell elements are used only. Although 3D wing is a solid model, both of our analysis tools only

work with the shell elements. Panel solver uses exterior surface of the wing and so it only needs shell mesh of the exterior surfaces. Structural solver uses both exterior and interior parts of the wing, but it uses shell elements with thicknesses. Thus, it analyzes the wing with internally generated volumes.

As there are two different solvers (panel and structural) with different solution aspects (aerodynamic and structural) and different mesh inputs, it is not possible to use the same mesh. Since each optimization step uses both aerodynamic and structure solvers, these solvers explained in section 2.4 have to be run automatically. And since each optimization step gets a different wing model, their mesh have to be generated automatically too. So, two different mesh automation procedures are implemented: one for panel solver and the other for structural solver.

After creating the mesh, all shell meshes should be grouped properly according to their surface properties that are necessary for the analyzes. According to these properties, appropriate boundary conditions need to be assigned to each shell group. These groups are also called as zones. Boundary conditions (BC) are constraints necessary for the solution of the differential equations that are needed for aerodynamic and structural analyzes. In more clear words, BCs are the definitions for solvers to give them an information about surfaces. This information is used to treat differently to meshes that have different BCs. Since BCs are vital for analyzes, BC assignments must be done in the mesh automation process, just after the mesh generation.

3.2.1 Mesh Automation for Aerodynamic Solver

Aerodynamic solver used in this study has two parts: panel solver and boundary layer solver. PanAir [69] is used as panel solver and XFOIL [71] is used as boundary layer solver. As explained in the previous chapter, XFOIL is only used to complete drag coefficient by solving boundary layer equations.

Generation of 3D wing geometry is done by using CST method as explained in the first section of this chapter. Then, a mesh is generated for wing geometry according to requirements of panel solver, namely PanAir [69]. It uses structured (quadrilateral) shell mesh as an input for the aerodynamic solution. The generated solid of wing

consists of four surfaces on it. The two of them are the upper and lower surfaces and the other two are the base and tip surfaces. Since the flow goes through leading edge to trailing edge, we decided to place more elements along the chord axis of wing than the span axis of wing. Also the base and the tip surfaces have minor effects on total solution, so these surfaces have less elements than the upper and lower surfaces as it can be seen in Figure 3.3.

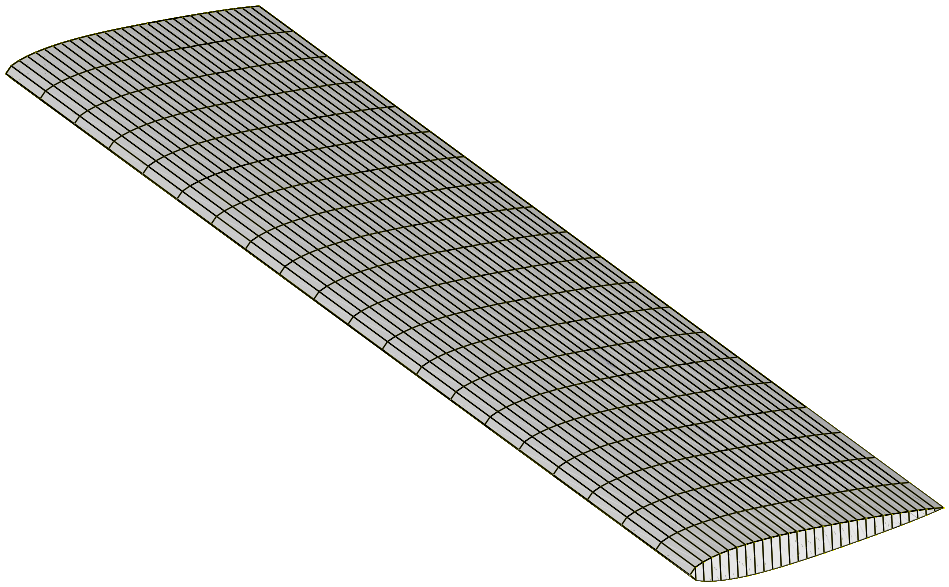


Figure 3.3: Mesh automation for panel solver

As panel solver has exponential running time with respect to number of elements [69] and it is executed for each individual in the evolution, we tend to keep the total number of shell elements minimum. After the number of points along each of the axes are decided, input file can be created according to division configuration. In our configuration, we divide the chord axis of wing into 30 equal sized lines. And the span axis of wing is divided into 20 equal sized lines. In summary, 600 shell elements (30×20) are generated for the upper and the lower surfaces, and 30 shell elements (30×1) are generated for the base and the tip surfaces.

There are different shell element zones for each faces of the 3D wing geometry. These zones are defined as a network (of vertices) in the input of the panel solver. In panel solver, we can assign different boundary conditions (BC) to each network to make

them behave differently. However, all networks have the same BCs in our case; so, we do not need to split networks. But since geometric parameterization and 3D geometry generation creates four NURBS surfaces (upper, lower, base and tip), we decided to keep them as four networks.

In order to accelerate mesh generation process, we obtain coordinates from the points on the NURBS surface of generated 3D wing. By that way, we can get points of NURBS surface easily and accurately in order to generate shell elements. OpenCascade [77] framework allows us to create, modify and play on NURBS surfaces.

Another important point in this mesh generation step is the normals of the shell elements. In panel solver, they must show the flow direction. So, we handle normals in the shell creation process. If we order shell elements with nodes respecting the right hand rule, we can control the normals of the generated shell elements. The solver finds out the normals of shell elements from its input. The input contains grid points of surfaces (nodes of shell elements), angle of attack and the velocity (mach number). Furthermore, reference positions (moment, area etc.) of input geometry must be defined in the input of the solver.

We need to put wake on the trailing edge of the wing, so that boundary curves from upper and lower sides meet on it. Wake is used in panel solver for the separation of up and down flows, but they do not physically exist. So wakes have different boundary conditions according to panel solver, since they need to treated differently. In automation process, we also put wake on trailing edge of the wing as recommended in manual of panel solver [69].

After mesh automation for PanAir, input file of XFOIL is also generated automatically to get missing component of drag coefficient, which is explained in the previous chapter. Since input file of XFOIL needs skin pressures from panel solution, its input file has to be generated after the panel solver.

From the nodes provided from NURBS surface of the generated 3D wing geometry, the connectivities of the shell elements, the flow and geometry properties, input file for both PanAir and XFOIL are automatically generated for each individual throughout the generations. For each of the individuals, combined aerodynamic

solver (PanAir+XFOIL) is run to get lift constant C_L, drag constant C_D, and skin pressures, which are explained in subsection 2.4.1. Hence, these outputs can take part in the optimization of the wing design, by using them in the objective function of the genetic algorithm.

3.2.2 Mesh Automation for Structural Solver

The 3D wing geometry generated with CST method and lofting does not have any inner structures to support exterior of the wing. In order to analyze a wing with structural solver, internal parts needs to be defined. These internal elements (spars and ribs) of the wing are demonstrated in the Figure 3.4, where spar is shown in blue and ribs are shown in red, provide particular resistance and strength to the wing so that it stays intact and does not collapse. So, before the use of the structural solver, we have to create spars and ribs inside of the parameterically generated wing geometry.

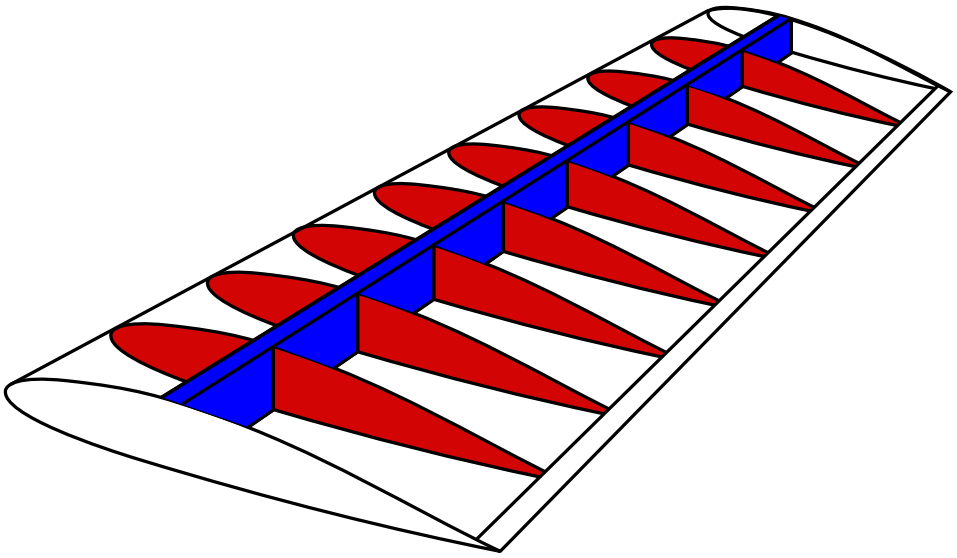


Figure 3.4: Internal structural elements of the wing. (Spar in blue, ribs in red)

Although spars and ribs are the main component for the resistance of the wing, they are main sources that increase the weight of the wing too. In structural optimization side of our work, we try to find durable design for the whole wing structure, but also minimize the weight of the wing. A special 2-level offset method is developed and

it is applied to the ribs of the airfoil geometry in order to create empty zones in the internal part of the wing. So that the 2-level offset operation is applied on ribs to minimize the weight of the wing. As it can be seen in Figure 3.5 and 3.6, the offset rib of the generated random airfoil ($0.0625454\ m^2$) has 31.3% less rib area than that is not offset ($0.0910314\ m^2$), which provides particular minimization in the weight of the wing, since ribs are duplicated along the span axis of the wing structure. Furthermore, developed 2-level offset method also helps the grouping operation, which will be explained later.

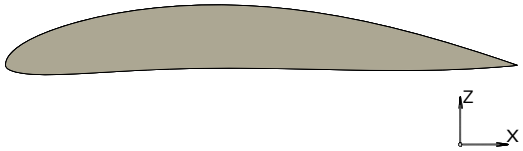


Figure 3.5: Classical rib area ($0.0910314\ m^2$) on airfoil that has 1 meter chord length

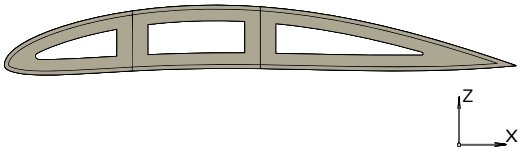


Figure 3.6: Generated offset rib area ($0.0625454\ m^2$) on airfoil that has 1 meter chord length

In contrast to the aerodynamic solver, the structural solver accepts both structured (quadrilateral) and unstructured (triangular) shell meshes as input. Since some of the faces of offset rib have different than four edges, it is really hard to generate structured mesh on these faces. Therefore, unstructured shell mesh is generated automatically on the all internal parts (spars and ribs) and the skin of the wing, which is shown in Figure 3.7. As mentioned before, skin pressures from the structured mesh of the aerodynamic analysis need to be given to the skin of the structure mesh of the wing to conduct structural analysis. However, the skin mesh of both solvers are in different types: panel solver uses structured shell meshes and structural solver uses unstructured shell meshes. So the skin pressures on the structured shell mesh of the aerodynamic solver need to be transferred to the unstructured shell mesh of the structural solver.

Unstructured shell mesh is created with the well-known triangulation algorithm called Delaunay triangulation [78]. It is applied on all faces of the wing with a few constraints. Firstly, since there is a significant curvature on the leading edge of the wing,

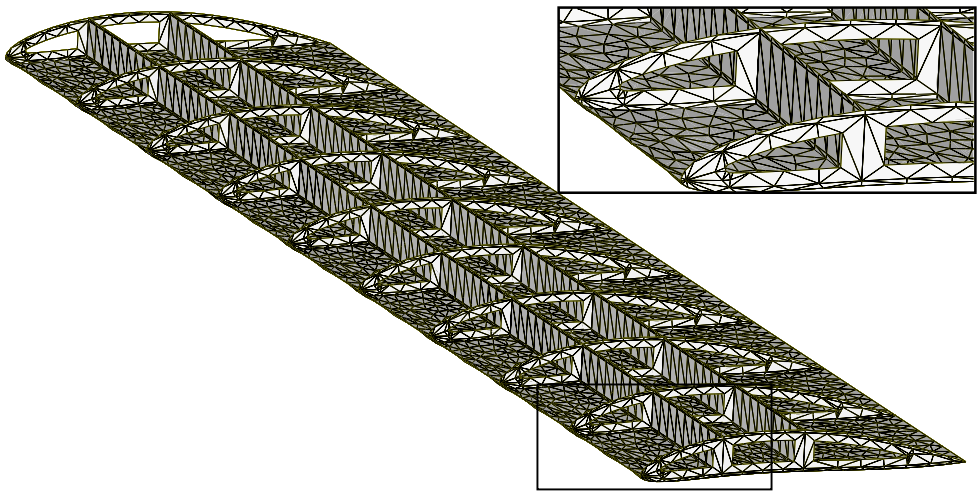


Figure 3.7: Mesh automation for structural solver

we have to create a fine mesh around it. Secondly, there must be shared nodes on the common edges of the faces to guarantee the continuity. Lastly, in order to get skin pressures without any loss in the transfer process, skin mesh should be fine enough.

Generating meshes with the above constraints is done by using MESHeda [79], mesh module of the CAEeda^{TM} software [80]. MESHeda allows user to define constraints (such as number of nodes on the edge, number of shells on the surface, etc.) on the geometry and then generates a Delaunay triangulation mesh accordingly. In our automation process, shell elements are automatically generated on all faces of the wing geometry.

Unlike the aerodynamic solver, thicknesses of all parts of the wing must be given as an input to the structural solver. So that it can calculate all forces on each part of the wing. Nevertheless, each part of the wing may have different thicknesses. In order to give different thicknesses to the different part of the wing, we need to group them in zones. Then, each element zone gets different thickness values. As it is mentioned before, the 2-level offset method is also used to group internal parts of the wing.

One of zones consists of all shell elements of the skin. Moreover, spars and ribs are divided into five different zones: upper spar, lower spar, center spar, inner rib and outer rib, which are represented in Figure 3.8. The grouping with zones allows

our optimization process work better, because whole spar or rib does not need to have the same thickness. In total, there are six different shell element zones with the different thicknesses. These six zones are automatically created by our automation process. Then, in order to solve 3D wing geometry, structural solver internally creates volumes by using given thicknesses on the mid-surfaces (given shell elements) of these imaginary volumes, which can be seen in Figure 3.9.

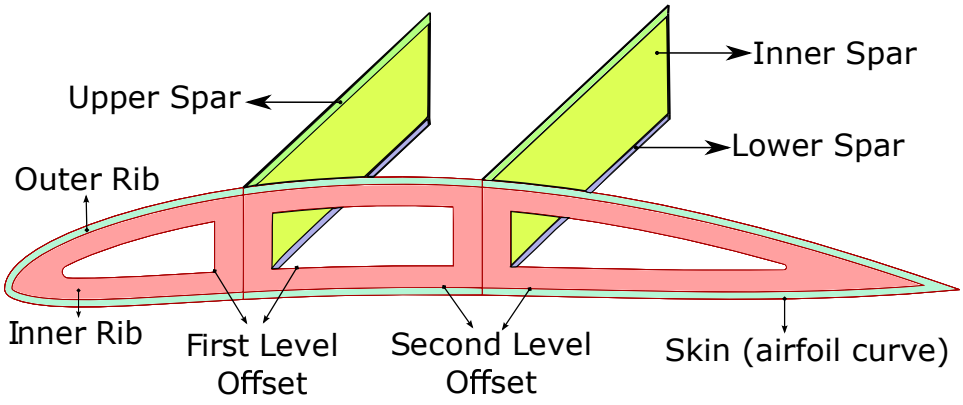


Figure 3.8: Defined zones of internal parts of the wing.

In each individual from the genetic optimization process, the mesh automation starts with the creation of spar and rib geometries inside of the wing, where ribs are offset. Then unstructured shell mesh is generated on all faces of the wing geometry. After the mesh generation, groups and their thickness assignments are set automatically. Lastly, material of the wing is set; it is also important for the optimization process, because deflection of wing is calculated with the help of material properties in the structural analysis. Furthermore, the weight of the whole wing can be calculated by using material properties.

Mesh nodes, connectivities of shell elements, zone information, material properties and transferred skin pressures compose the input file of structural solver. Similar to the input file of the aerodynamic solver, the input file for structural solver is also generated automatically for each individual in the generations of the optimization process. By using these inputs and the structural solver; deflection, Von Mises stresses [75] and the weight of the total wing volume can be computed and used to optimize the wing design.

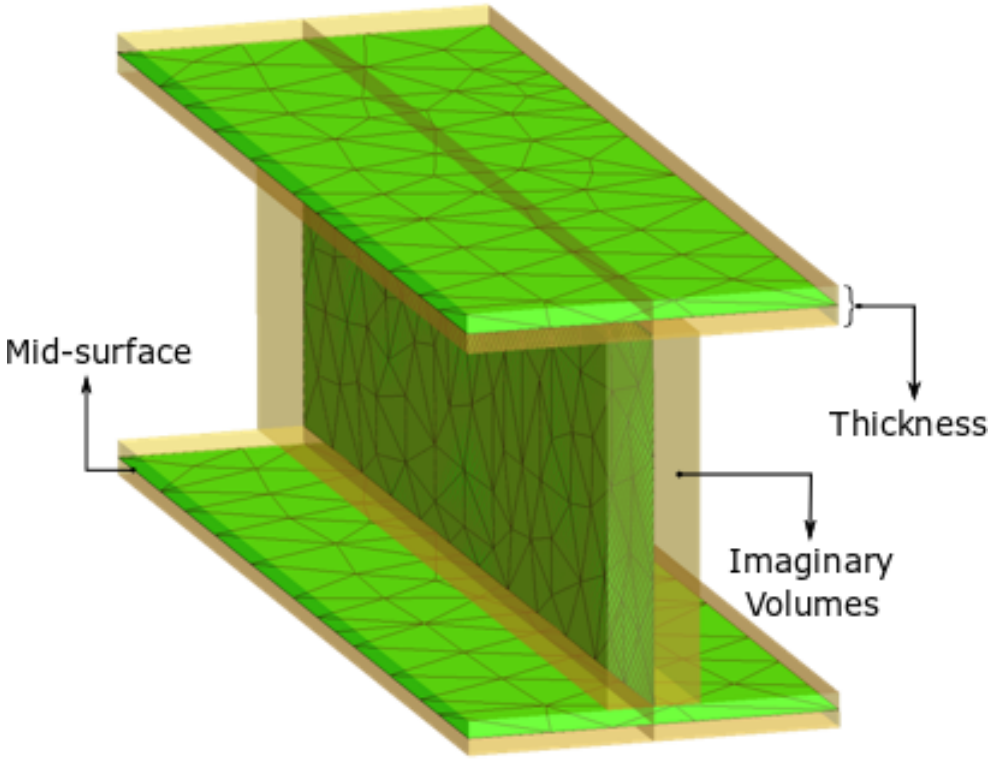


Figure 3.9: Imaginary volumes of structural analysis.

3.3 Shape Optimization via Parallel Genetic Algorithm (PGA)

A wing can be better than the others in various aspects, such as aerodynamic, structural, thermodynamic, etc. In order to design an optimized wing, an objective must be decided. Objective of this thesis is to reach better wing with respect to both aerodynamic and structural aspects. Hence, aerodynamic and structural solvers are used to compute the performance and efficiency of the aerostructural shape design of the wing.

For optimizing the wing model, ParadisEO [81, 82, 83], which is based on Evolving Objects (EO) framework [84], is adopted. It is an open source, object oriented and a template based framework which is written in ANSI-C++. EO includes many nature-inspired heuristic search paradigms like genetic algorithms, particle swarm optimization and many kind of selection, replacement and variation operators. The

framework allows us to design our optimization algorithm by using these paradigms and operators. Since it is an open source framework, algorithm designer (developer) can define and code his/her own classes that inherit from the classes of the framework. Thus, as a component based framework, EO can be used like Lego blocks according to our specific needs in the evolutionary optimization algorithm. ParadisEO is an extension of hybrid, parallel and distributed heuristic search library on EO framework. It also includes base classes for different parallel GA models like island model, so we use it for both island model and global parallelism.

This thesis uses ParadisEO framework with parallel genetic algorithm paradigm, both global parallelism and island model, to design an optimization algorithm. Its setup includes mutation and crossover operators for variation. Also tournament selection and elitism are applied in the genetic optimization cycle. Furthermore, since the fitness evaluation part is the most costly part of the genetic optimization used in this thesis, the fitness calculation process is parallelized by using the master-slave paradigm which is called *global parallelism* in GA. Each parallel processor calculates fitness value of a different individual of the same generation. Then master processor collects fitnesses to apply selection, variation (crossover and mutation) and replacement operators, in order to generate the individuals of the next generation.

In addition to global model, island model is also utilized to discuss its heuristic search performance and compare it with the performance of non-island (single population) model.

In the tournament selection technique, random n individuals are chosen to perform a tournament between them. Best individuals (fittest) of tournaments are passed on to the next generation. Elitism technique in the replacement operator saves best individuals throughout generations from the possible degeneration caused by the variation operators (crossover and mutation). In addition to generational replacement operator, we implement a kind of an elitism method; so that we move the best individual of the generation directly to the next generation without any use of variation operators. Therefore, we can keep the best chromosomes until there is a better individual.

In order to generate external and internal shape of the wing, 15 design parameters are used: 8 of these are Kulfan's CST parameters, which correspond to the physical

airfoil shape; other 5 define sizes of spars and ribs; and last 2 are taper ratio and sweep angle. All parameters encoded as floating point in chromosomes, which have their own upper and lower limits. Genetic algorithm evolves repeatedly generated design candidates using mutation, crossover and selection throughout the whole design process. Initial population is created from random individuals and at each generation, high fit individuals are selected for crossover and mutation operations to transfer their lineage to the next generation.

In our work, the fitness value is calculated by using flow coefficients from the panel solver and the weight of the wing from the structural solver. The failure limit of the wing is added as a constraint that affects as a penalty component in the fitness value to get only strong individuals survive through generations. In our GA implementation, the penalty component of fitness value has an adaptive penalty weight factor. In order to set mentioned penalty weight factor, a variable keeping penalty adaptation period is added to our implementation of elitist replacement operator. At each period (generational interval), the penalty value of the best individual is checked and it is decided whether the penalty weight will be updated or not as it is proposed in [85]. This adaptive penalty approach prevents GA from the genetic drift and premature convergence. More details about this approach and its usage on this thesis will be discussed in Chapter 4.

3.4 Data Transfer from Quadrilateral Mesh to Triangular Mesh with ADT

In order to connect aerodynamic analysis and structural analysis, aerodynamic pressure data should be transferred from quadrilateral panel solver mesh to triangular structure mesh. The surfaces these meshes represent intersect at the skin shell of the wing but they are contain different set of vertices. After the aerodynamic analysis ends, found pressure coefficient (CP) on skin nodes should be converted to pressures. Then, these pressures should be sent to shell elements of structural analysis' skin mesh as an input load, in order to predict whether the designed wing model will break or not in the structural analysis. So, ADT is needed to search closest vertices in the target mesh and transfer pressures efficiently. The algorithm is given at 2 is implemented by using FORTRAN 90.

Algorithm 2 Alternating Digital Tree search algorithm

1: **procedure** SEARCHADT(T, p, S) ▷ T: Tree, p: Query point, S: Set of returned elements
2: **if** $root(T)$ is a leaf **then**
3: **for** $x \in root(T)$ **do**
4: Insert x in S
5: **end for**
6: **else** ▷ Recursive call is done according to next alternating axis. $left(T)$ and $right(T)$ is the left and right subtree links of T.
7: SearchADT($left(T), p, S$)
8: SearchADT($right(T), p, S$)
9: **end if**
10: **end procedure**

The ADT search algorithm was used in the work [86]. The developed interface utility is named as Search and Interpolation (SINeda) [87] and used for fluid-solid interaction problems in the scope of the work [86]. The old implementation of SINeda works with file I/O operations, which takes much more time compared to working directly in memory. In this thesis, we refine the code so that its data structures can be used directly from C++. Therefore, time consuming file I/O operations are eliminated, which makes SINeda faster.

In the aerodynamic analysis, pressure coefficients are computed. Then, giving that skin pressures as loads to the structural model of same design makes the structural optimization more accurate, since the structure of the wing will be analyzed under more realistic loads. As a result the optimized design will be more realistic.

Since meshes of two analysis are different with each other, data cannot be used directly. The difference is not just type of elements (triangular vs quadrilateral), nodes and connectivities are also completely different as shown in Figure 3.10. Thus, the difference in meshes forces us to transfer data from one to other, where the developed SIN interface algorithm takes place. The algorithm is stated at 3 is implemented by using FORTRAN 90. In our case, panel mesh (quadrilateral, structured) is the source mesh that gives the pressures and structural mesh (triangular, unstructured) is

the target mesh that gets the interpolated pressures.

Algorithm 3 Search and INterpolate (SIN) algorithm

1: **procedure** SIN(N_s, E_s, N_t, E_t) ▷ N_s: Nodes of source mesh, E_s: Elements of source mesh, N_t: Nodes of target mesh, E_t: Elements of target mesh
2: T_s = ConstructADT(N_s, E_s) ▷ T_s: ADT of source mesh
3: **for each** $n \in N_t$ **do**
4: SearchADT(T_s, n, S) ▷ S: Set of returned elements
5: Interpolate the data on $S[0]$ to n ▷ $S[0]$: Closest element
6: **end for**
7: **end procedure**

Firstly, alternating digital tree is constructed with the structured mesh of aerodynamic analysis. For each node in the unstructured mesh of structural analysis, search query is processed on ADT and closest quad element is returned. Since both meshes are created from the same solid model, each query node (from unstructured mesh) is found on the closest quad element with some tolerance. Consequently, the data on the structured mesh is interpolated to the query node of unstructured mesh.

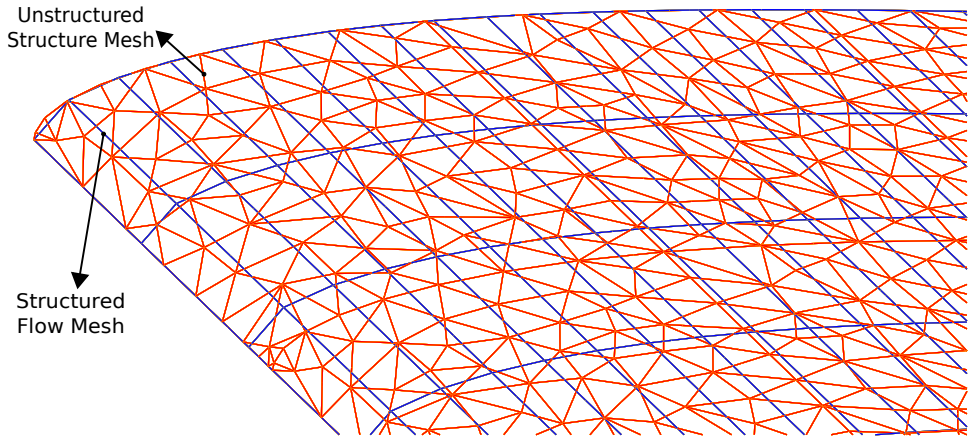


Figure 3.10: Overlapped both mesh as an ADT input

An example of data transfer between our generated wings can be seen in Figure 3.11. In this example, pressure coefficients (CP) are computed by panel flow solver and these CP values are transferred to the unstructured mesh of FEM structure solver by using ADT.

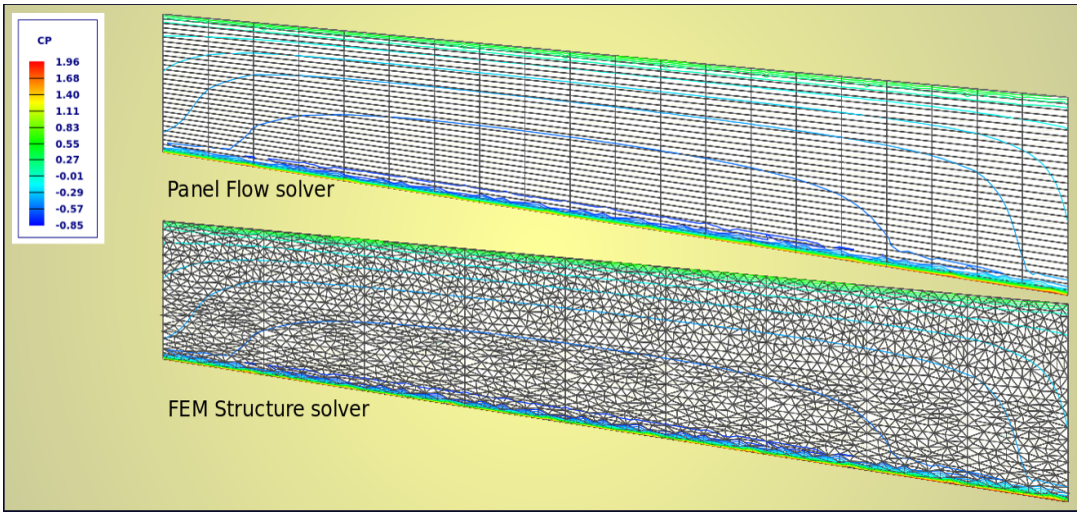


Figure 3.11: Transferred pressure coefficient (CP) values between two different mesh

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1 Environment Setup

Hardware, software and libraries used for the development of a parallel aerostructural shape optimization platform are listed below:

- Processor: Intel Core i7-4790K 4.00GHz (4 Cores) (8 Nodes = 32 cores in total)
- Network: 1 Gbps Ethernet
- Programming Languages: C++ is preferred mostly, since CAEeda (main software of EDA Ltd.) has been developing with it. Other that C++; FORTRAN 90 for search and interpolate (SIN) library and Python for scripting purposes are used.
- Parallel Genetic Algorithm Framework: ParadisEO, implements global and island parallelism.
- Parallel middleware: MPICH, which is portable implementation of MPI (Message Passing Interface).
- Geometry Framework: CADeda, which uses OpenCascade 6.9.1 framework. All geometric automation implementation is done in CADeda by using OpenCascade.
- Mesh Library: MESHeda, which uses Delaunay triangulation.

- Development Environment: CAEeda has been developed by using Qt5 and C++. Also all programming of mesh and geometry automation, SIN library calls and parallel genetic algorithm implementation are done with using C++.

4.2 Design Process

In the context of this study, we developed an integrated and automated aerostructural 3D wing shape parallel optimization platform and it includes:

- An automated and parametric geometry generator based on a hybrid CST-NURBS method.
- An automated mesh generator (both flow and structure).
- An automated input file creator for both solvers (flow and structure).
- A parallel genetic algorithm optimizer tool that uses output of solvers to optimize 3D wing shape.

Our multidisciplinary optimization cycle starts with creation of an airfoil geometry with Kulfan's CST parameterization method coupled with NURBS curve, followed by creation of a 3D wing geometry. This wing geometry can be considered as an initial shape for mesh automation system. The generated 3D wing can directly be used to generate the panel solver mesh (structured), but internal parts of the wing need to be created before the generation of the structural solver mesh (unstructured). So there are two different mesh: one for the aerodynamic panel solver (PanAir) and another for the FEM structural solver (SAPeda).

For the panel solver, a structured surface shell mesh is generated on the skin of the wing. For the structural solver, an unstructured surface shell mesh is generated both on the skin and internal parts of the wing. Subsequently, pressure data transfer between mesh of both solvers is handled by the interface utility named SINeda. The geometry of the surface and internal parts and the corresponding meshes are automatically generated during the optimization process, which is the key novelty of this thesis.

4.2.1 Genetic Encoding

Since all the geometry and mesh are generated automatically during the optimization process, inputs for the geometry and mesh generation need to be parameterized. In other words, searching for the best wing is a problem and we have to parameterize the problem, so these parameters are the encoding for a proposed solution. It is also called as chromosomes of the individuals for the GA. Each individual has its unique parameter set, namely chromosomes. As shown in Figure 4.1, chromosomes of this work are formed by airfoil shape (CST) parameters, thickness parameters for internal parts of the wing, taper ratio and sweep angle.

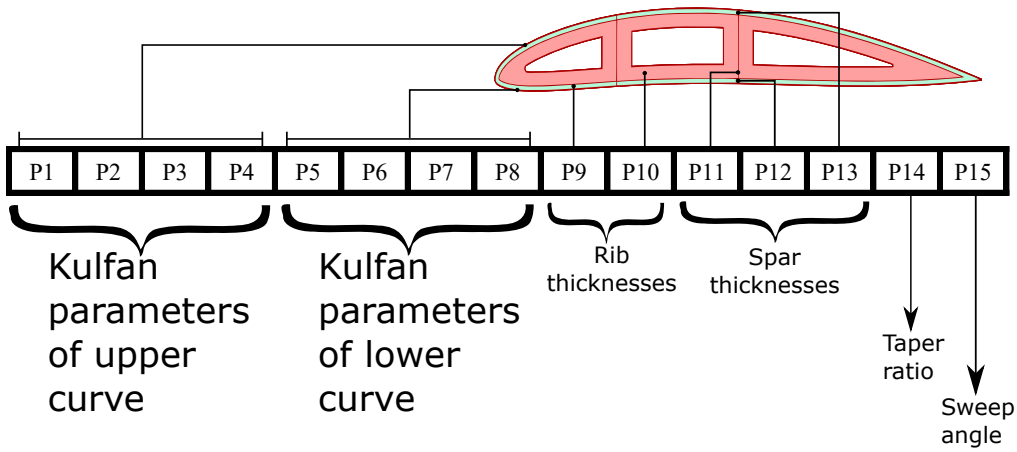


Figure 4.1: Genetic encoding

4.2.2 Objectives

Developed platform takes initial geometric data, flight, atmospheric and material data. Moreover, for a genetic inputs, objective and penalty functions must be an input for the genetic algorithm. As it can be seen in Figure 4.2, all these input data is used to calculate different fitness values for each individual. At the start, there are genetic encoding parameters but also some constant variables for a one genetic run. For example, the shape of the airfoil is constructed by CST parameters, which are the genetic chromosomes and they differ for each individual. However, material of the wing does not change throughout the generations and it is constant variable for the experiments.

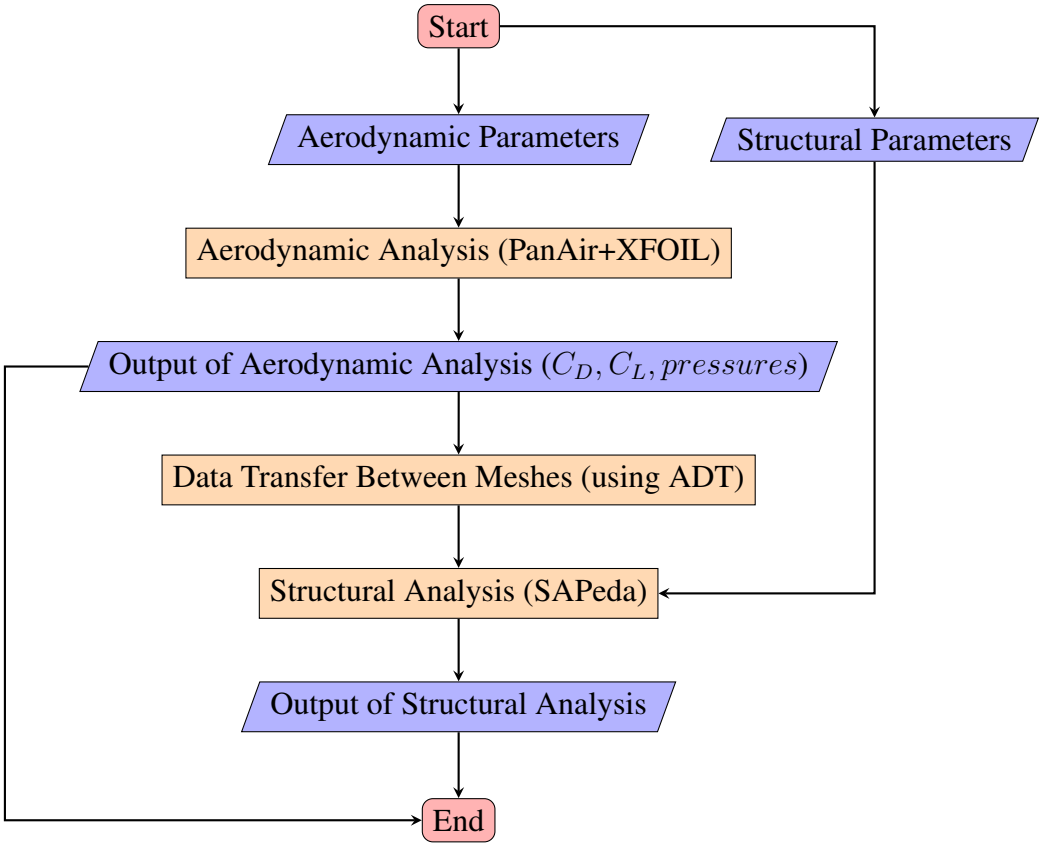


Figure 4.2: The flow chart of the fitness calculation process.

From initial geometry parameterization data, two different geometry and mesh generated. In panel geometry, the only needed part is the skin of the wing, since panel solver conducts an aerodynamic analysis. Nevertheless, structural analysis also needs internal parts of the wing; so structural geometry differs from panel geometry. In the first hand, automated panel mesh (structured mesh) is generated with panel geometry, which is explained in Chapter 3. Then, together with the aerodynamic parameters (angle of attack, mach number, etc.) and panel mesh, input file for aerodynamic analysis (PanAir+XFOIL) is generated.

Coefficients C_D, C_L and aerodynamic skin pressures (C_P) on the wing are calculated by using the aerodynamic solvers. At the end of the aerodynamic analysis, we take the aerodynamic pressures and transfer it to the structural analysis input. For the structural analysis input, automated mesh (unstructured mesh) is generated from the geometry with internal parts (ribs and spars), which is also explained in Chapter 3. Using this mesh, transferred aerodynamic pressures and other structural parameters

(material properties, thicknesses), input file for structural analysis (SAPeda) is generated. Output of the structural analysis contains deflection and von-Misses stresses (σ) of the wing. Additionally, weight of the wing (W) is calculated with the mesh area and thickness parameters of different parts of the wing. Weight calculation is done as follows:

$$W = t_{Mesh} \cdot A_{Mesh} \cdot \rho \cdot g \tag{4.1}$$

where t is thickness, A is the area, ρ is the density of material and g is the acceleration of gravity. In this thesis, fitness (objective) function of the GA is defined as:

$$f_{objective} = \frac{C_D}{C_L} \cdot W + \omega_{penalty} \cdot f_{penalty} \tag{4.2}$$

where C_D is drag coefficient, C_L is lift coefficient, W is the weight of the wing and $f_{penalty}$ is a penalty function with its weight factor, $\omega_{penalty}$. Lower C_D/C_L ratio is one of the major goals in aerodynamic shape designs; as lift is needed to fly and reaching that lift with lower drag directly leads to better climb performance and fuel consumption. Moreover, since we try to minimize total weight of the wing, it tends to be thin and fragile. So, we also define a penalty function to eliminate failed wings throughout the generations, which is defined as:

$$f_{penalty} = \begin{cases} (\sigma_{vm}^{max} - \sigma_{vm}^{yield})^2 & \text{if } \sigma_{vm}^{max} > \sigma_{vm}^{yield} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

where σ_{vm}^{max} is the calculated maximum von-Mises stress on the wing and σ_{vm}^{yield} is the material yield stress (von-Mises stress), which defines the failure limit of the wing. The material yield stress is material property that defines the beginning point of nonlinear deformation (failure of material).

Furthermore, in this thesis, we have implemented penalty weight factor $\omega_{penalty}$ as a variable throughout generations, which is called adaptive penalty in evolution of problems with constraints [85]. It allows us to make sure that the penalty weight factor is neither too small nor too large at any stage of the genetic optimization. If a single $\omega_{penalty}$ is used for entire course of the optimization, it becomes difficult to find a solution with any $\omega_{penalty}$ that is good for all stages of the optimization. A small weight tend to result in unfeasible wings whereas large weight avoids searching search space at the borders of feasibility. After the initialization, adaptation of penalty allows us

to sample adequately the search space and then gradually it pushes the optimization process to converge to the good solution. Therefore, premature convergence and genetic drift can be avoided and the possibility of finding the global optima rather than local optima increases.

Initially, we set $\omega_{penalty} = 1.0$ and we check it after every three generations. If the best individual of last three generations is always feasible (no penalty component), then we set $\omega_{penalty} = \omega_{penalty} \cdot 0.83$. But if the best individual of last three generations is never feasible, namely it has a penalty component for all three generations, then we set $\omega_{penalty} = \omega_{penalty} \cdot 2.0$. In other cases, $\omega_{penalty}$ remains same. Hence, our GA can adapt itself to search properly the regions adjacent to constraint boundaries as the global optima may lie there.

4.2.3 Parallelization

In the scope of this thesis, two different models of PGA are experimented: single population model and island model. In the single population, there is only one initial population to evolve. However, in the island model, there are many populations like they are in different islands and they evolve separately. But in fact they are not totally separated, migration is done at some generations for exchanging individuals to exchange genes among the island.

As a first model, we use GPGA paradigm, which is also called as the master-worker model, and the working scheme of this model is shown in Figure 4.3. For the single population model, master processor controls the generation loops and passes individuals to the slaves. Slave processors are responsible for only the calculation of the fitness function and they send the fitness value of the individual back to the master processor. After all fitnesses are calculated for the current generation, master processor applies genetic operators (selection, mutation, etc.) to the population to create the next generation.

CPGA paradigm is the second model that we use, which is known as an island model. In our implementation of an island model, master processors for each island are needed to apply genetic operators, such as selection, replacement etc. Furthermore,

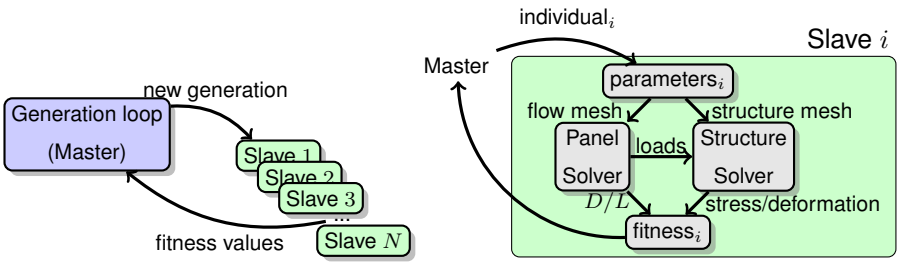
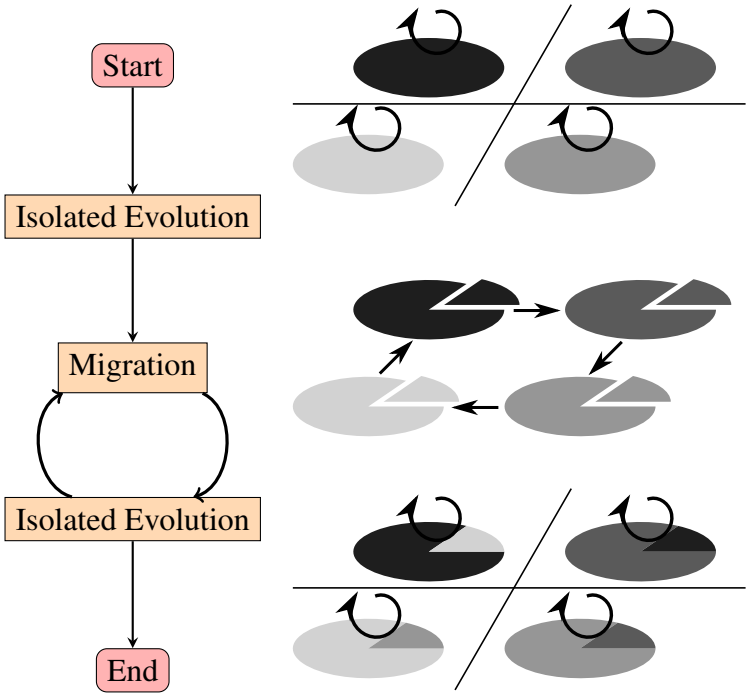


Figure 4.3: The master-worker model for parallel genetic algorithm

the communication between the islands for migration is handled directly between these master processors. As stated in Chapter 2, there are many individuals in sub-populations for each processor to calculate their fitness values. When certain number of generation passes, a few individuals from each island are migrated according to the island topology. In this thesis, we construct a ring topology with the four islands and the evolution cycle of these islands is shown in Figure 4.4. Sub-populations evolve separately, however this evolution is only interrupted by the migration at the certain periods of generations (migration interval).



a) Flow chart b) Symbolic model

Figure 4.4: Island model with four sub-populations in ring topology.

4.3 Experiments

As it is stated in the previous sections, genetic encoding of our work has 15 parameters and the bounds of these parameters are shown in Table 4.1. These bounds are determined based on common wing design practices. Due to the leading edge (LE) radius and the continuity of LE, lower bounds of $P1$ and $P5$ are not set as 0.0, but 0.05. Since upper curve of an airfoil should provide the camber of the airfoil, lower bounds of $P2$, $P3$ and $P4$ are set as 0.0. Likewise, $P6$, $P7$ and $P8$ are also responsible for the camber of the airfoil, so they can pass through the upper side with the negative lower bound, -0.2. All upper bounds of the CST parameters are set as 0.4 to cover most of the airfoils from the literature. Moreover, all bounds of thickness parameters are set by considering percentage of thickness of airfoils from the literature and offsetting proportion on it. The significant point is that the upper bound of the inner rib thickness cannot be higher than the lower bound of the outer rib thickness. Relationship between the inner spar thickness and other spar thicknesses is the same as the significant point of rib thicknesses mentioned in previous sentence. Bounds of the other parameters ($P14$ and $P15$) are set by looking frequently used wings from the literature.

Parameters of the GA is given in Table 4.2. In the island model implementation of our work, we decided to use four islands with population size 15 of each, where we use population size 60 for single population case. By doing this, we can easily compare the island model and the single population model, since total fitness calculation count will be same across generations.

Besides these genetic parameters, there are also some constant parameters, which are set as the design choices:

- Mach number: 0.5,
- Angle of attack: 2 degrees,
- Span of the wing: 5 meters,
- Chord length of root: 1 meter,
- Skin thickness: 2 millimeters,

	Lower Bound	Upper Bound	Stands for ...
P1	0.05	0.4	1st CST parameter of airfoil
P2	0.0	0.4	2nd CST parameter of airfoil
P3	0.0	0.4	3rd CST parameter of airfoil
P4	0.0	0.4	4th CST parameter of airfoil
P5	0.05	0.4	5th CST parameter of airfoil
P6	-0.2	0.4	6th CST parameter of airfoil
P7	-0.2	0.4	7th CST parameter of airfoil
P8	-0.2	0.4	8th CST parameter of airfoil
P9	0.015	0.022	Outer rib thickness (m)
P10	0.010	0.015	Inner rib thickness (m)
P11	0.015	0.022	Inner spar thickness (m)
P12	0.022	0.030	Upper spar thickness (m)
P13	0.022	0.030	Lower spar thickness (m)
P14	0.6	1.0	Taper ratio
P15	0.0	20.0	Sweep angle (deg)

Table 4.1: Upper & lower bounds of the optimization parameters and their meanings.

- Material: Polyamide.

Parallelization of GA has done by using ParadisEO framework, which allows us to utilize several cores in parallel for the fitness evaluation. Parallelism with this framework needs one scheduler core that organizes and maintains the jobs of other processors. Depending on which model we use, the framework may also need one master processor. Hence, other than scheduler and master processors, all other processors are used in fitness calculation, which is the most time and computational power consuming operation of this thesis.

Execution time of the fitness calculation takes approximately 27.5 seconds in average. General fitness calculation consists several steps with given average execution times in seconds:

	Single Population Model	Island Model (x4)
# of Generations	100	100
Population Size	60	15 (x4)
Selection Type	Tournament (size = 2)	Tournament (size = 2)
Mutation Type / Rate	Uniform / 0.2	Uniform / 0.2
Crossover Type / Rate	Arithmetic / 0.7	Arithmetic / 0.7
Replacement Type	Elitist	Elitist
Migration Interval	Not applicable	5
Migration Size	Not applicable	2
Migration Topology	Not applicable	Ring
# of Cores (Parallel)	1 to 32	2 to 32

Table 4.2: Parameters of the genetic algorithm for both models.

- Initialization of geometry & mesh data structures: 10.00
- Automated panel geometry & mesh generation: 0.50
- Panel solver analysis: 9.25
- Automated structural geometry & mesh generation: 5.25
- Data transfer with SIN: 0.50
- Structural solver analysis: 2.00

Even though fitness calculation takes nearly 27 seconds in total, it takes less execution time when the individual fails at any step of the fitness calculation. For example, if the panel solver outputs negative C_L value, fitness function directly returns a very large value without entering data transfer and structural analysis steps; since a wing with negative C_L cannot fly. Although the parameters for shape creation are in reasonable interval, there might be irrational shapes with the "unlucky" parameter sequence, resulting in geometry or mesh automation failure. If the case is that, fitness function directly returns a very large fitness value. As mentioned in previous chapters, an individual with a very large fitness value has very small probability to transfer its genes to the next generations.

4.4 Results and Discussion

It needs to be reminded that all of experiments are conducted with the population size of 60 with the experimental setup defined in subsection 4.2. In the island model population is divided into 4 equal sub-populations, which makes total number of fitness calculations in the optimization process is the same in both models. So we can compare the total optimization time and performance of two models.

Results of experimentation with single population model and island model of PGA are presented in this section. In Figure 4.5, average fitness values of 5 different seeded experiments are presented. Also, in Figure 4.6, fitness value (fitness function is defined in subsection 4.2.2) changes of the best individuals of one experiment during evolution are shown. As we can infer from the results, island model is able to show better overall performance than single population model in 3D wing shape optimization. However, it can be seen from the Figure 4.6 that single population model reached better fitness value in the early 20s generations. Since the population sizes of each island are smaller than the single population model's, the diversity slowly affects on fitness minimization in the island model. Still, one of the islands (Island #2) reached better fitness value just a few generations later, and then genetic variety of Island #2 spread across the other islands. But the best individual of the single population model sticks around the same local minimal that is found in the early 20s generations.

In order to evaluate our fitness function, components of it for the best individuals from the single population model are demonstrated in Figure 4.7. As a reminder, fitness function was composed from two components: C_D/C_L and weight (W) component. The C_D/C_L component is affected directly from the parameters of the shape of the airfoil, but the weight of the wing is affected by all parameters.

It can be observed that the C_D/C_L component is more dominant than the W component. Genetic evolution firstly tends to minimize the C_D/C_L component, which has relatively large numerical interval and so it becomes more dominant than the W component. As a result of the dominance of C_D/C_L component, W component value increased while total fitness value was decreased by the C_D/C_L component in the early generations. After the decrease of the C_D/C_L component to some value, the

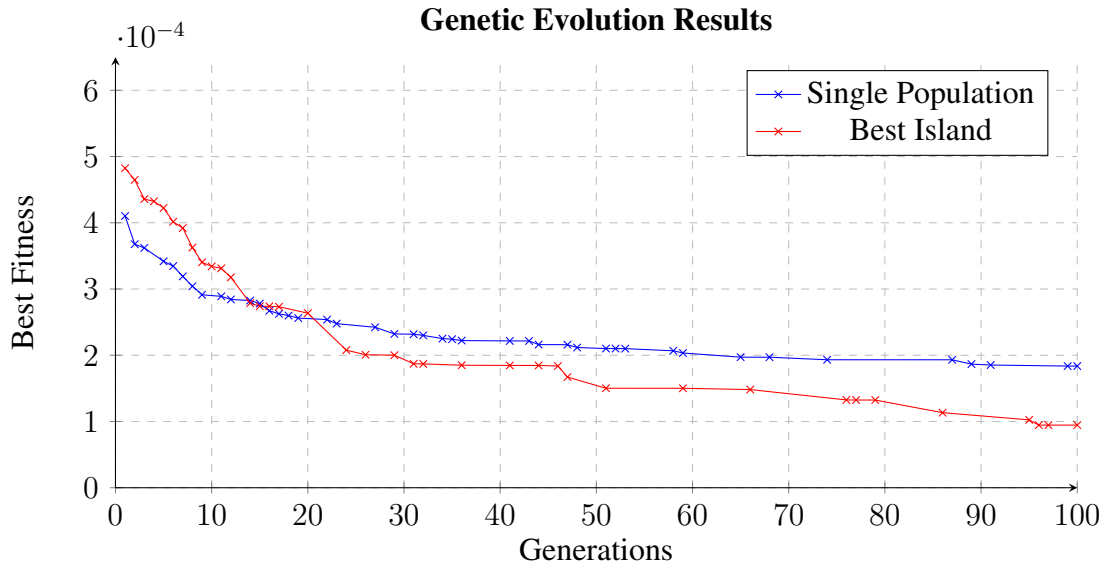


Figure 4.5: Fitness values of the best individuals for Single Population and Island Models (Average values of 5 runs with different seeds)

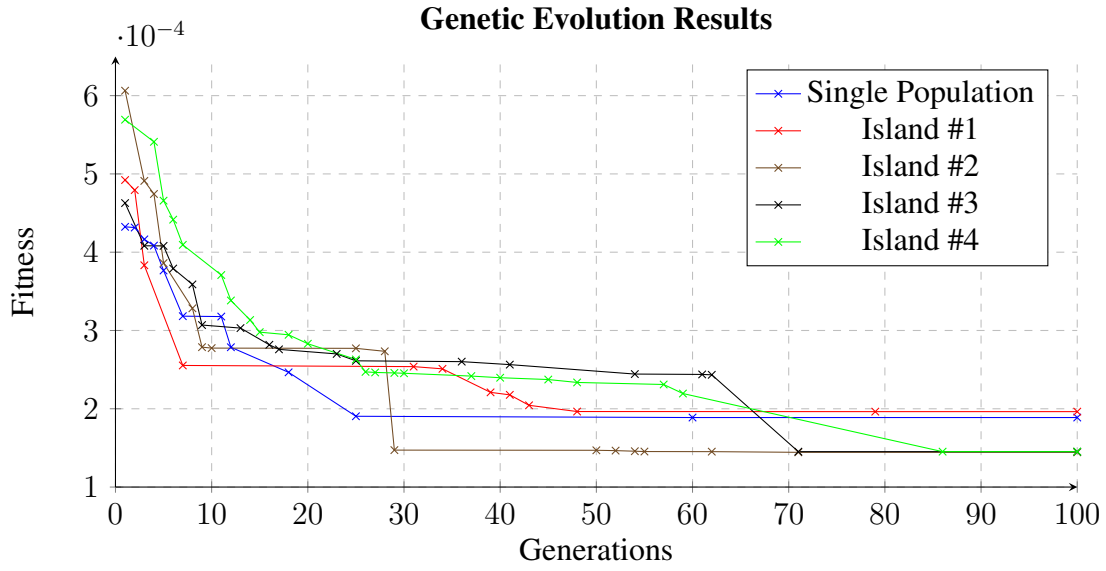


Figure 4.6: Comparison of the best fitness values of a single experiment for Single Population and Island Models

W component also entered a downward trend. During this trend, it was observed that thickness parameters were changed mostly. Nonetheless, it did not reach the minimal value of the first generations, due to limits of the best individual's fit shape that evolved throughout the generations. In more clear words, while searching for a

better individual, GA initially optimized the shape of the airfoil, and then it tried to minimize the weight of the wing.

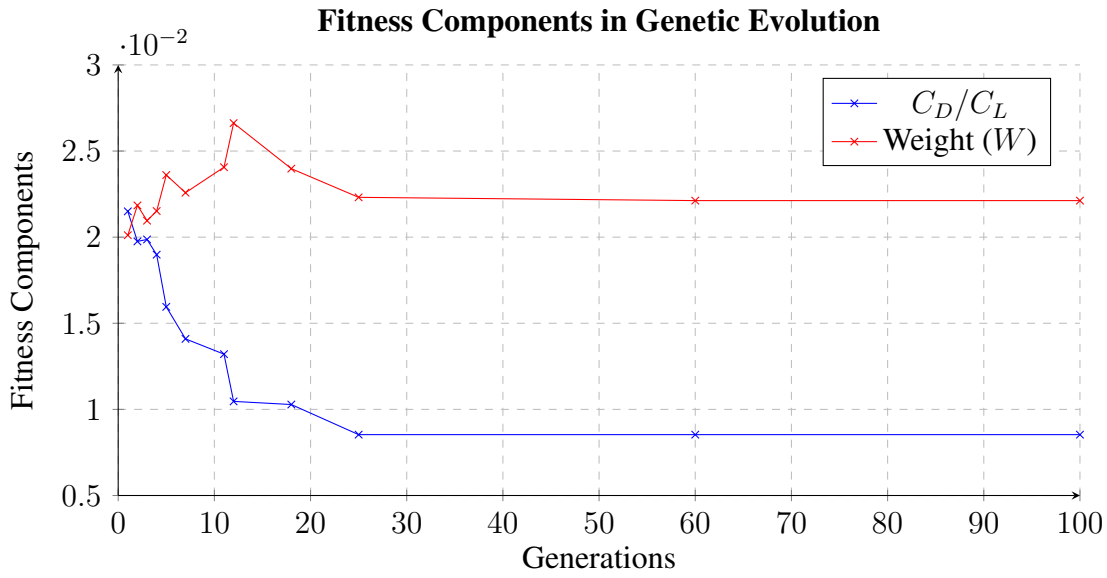


Figure 4.7: Fitness components of the single model GA of the sample experiment

As an another experiment, the evolution process was extended by increasing the number of generations. The result of this experiment is illustrated in Figure 4.8 and it can be inferred that there will always be a better individual in the future. For single population model, the algorithm found the local optima before the 100th generation and the fitness value converged on there prematurely. In the last 400 generations, just minor differences can be observed in single population model. But in island model, there were critical advances throughout the generations. The last better individual was found before the 400th generation. After that point, GA only found better individuals with minor differences. Thus, it can be said that the island model and its genetic diversity performs better when the maximum number of generations is increased.

In order to examine the adaptive penalty effects on the evolution, non-adaptive penalty experiments were conducted and the result of these experiments is given in Figure 4.9. In non-adaptive experiments, penalty weight ($\omega_{penalty}$) that is defined in Equation 4.2 was kept constant as 1.

Similar to the adaptive penalty experiments, single population model behaved better firstly and the island model reached better fitness value at the end. But in overall,

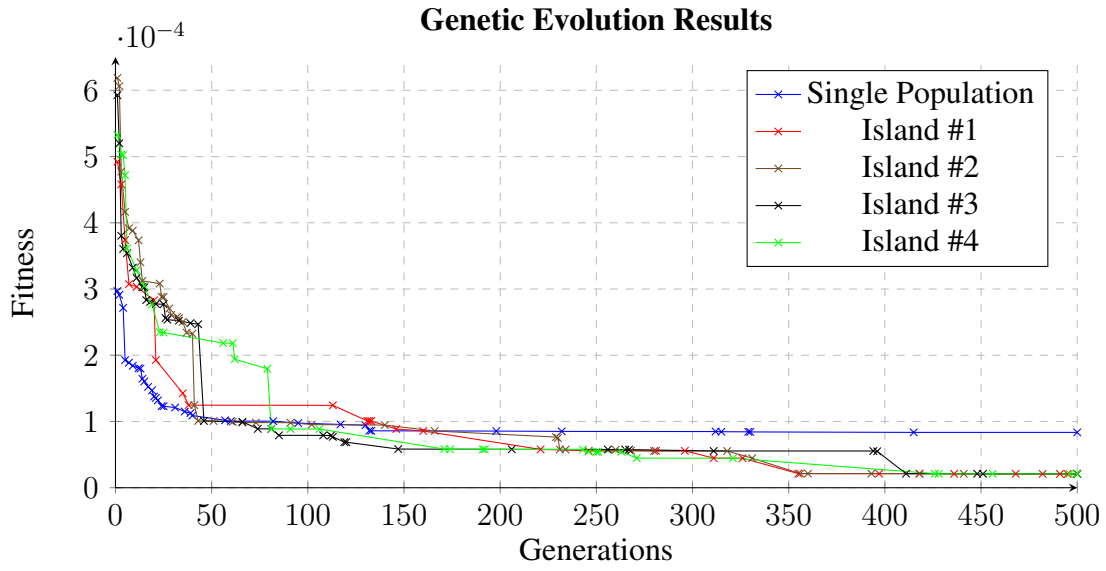


Figure 4.8: Comparison of best fitness values from Single Population and Island Model longer (500 generations) experiment for the best configuration

it can be clearly stated that GA with adaptive penalty performed better optimization when compared to its non-adaptive version. Thus, it was shown that initially set penalty constraint is so violent for defined fitness function and it blocked the evolution to reach to the global optima.

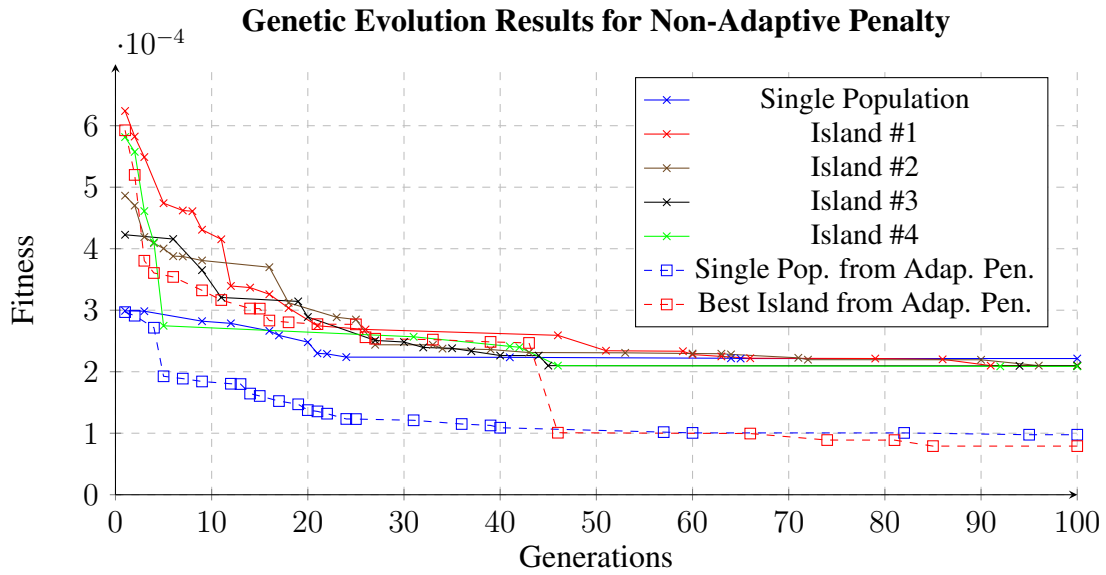


Figure 4.9: Fitness values of the generations

Another genetic variable that has been examined is the migration effect on the island

model of GA. Figure 4.10 shows the fitness value changes throughout the generation of non-migrated island model of GA. As we can see in the results of both migrated and non-migrated genetic evolution results, migration effect in the optimization of island model can be clearly seen. Since genetic diversity cannot be propagated without migration, each island behaved as the single population model with smaller population. So, without variation and enough population size, non-migrated islands performed far worse than the migrated ones. Additionally, it can be said that the increase in population size of non-migrated islands would end up with the similar results of the single population model that is also shown in the Figure 4.10.

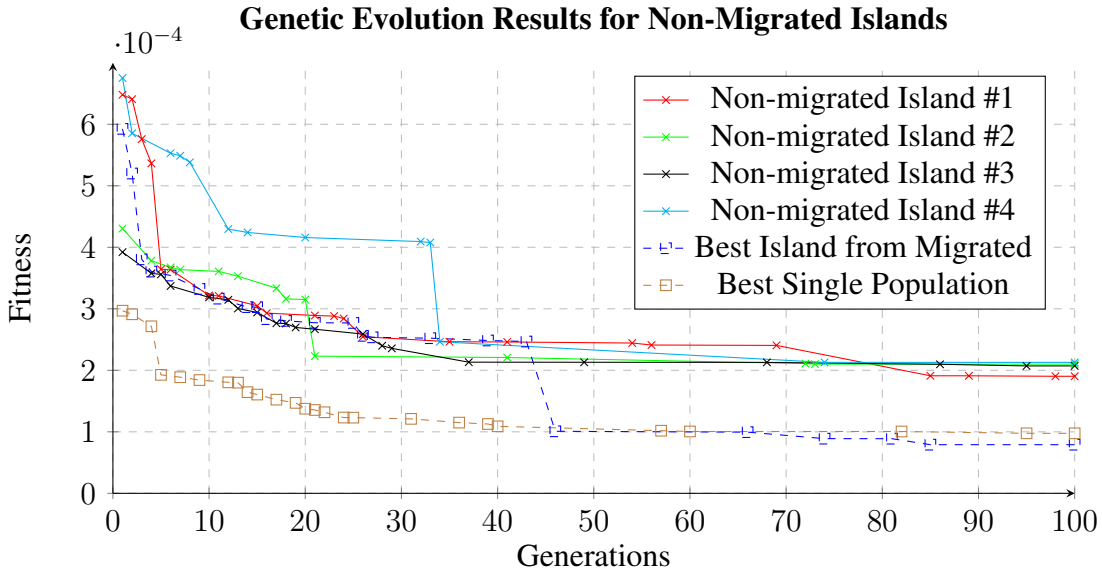


Figure 4.10: Fitness values of the non-migrated island model

Results of average execution times of our parallel experiments are shown in Figure 4.11 and Table 4.3. These average times are calculated by using five experiments with different seed values.

Execution time of single population model of PGA took almost 44 hours when it was running in one core. Correspondingly, when core count was doubled, execution time nearly halved. Since there are 60 fitness functions to be calculated in each generation, both cores are efficiently utilized. In two core experiments, each core calculates roughly 30 fitness functions in one generation. So, core utilization depends on average work per core, while fitness calculation during one generation.

The bound for parallel execution time is determined by fitness calculation per generation, because every fitness value needs to be known for advancing through generations. In other words, each core calculated at most two fitness functions in the experiments with 30 cores, since population size was 60 in our experiments. Additionally, using more than 60 cores would result in the nearly same execution time. So the asymptote of the exponential curve in the Figure 4.11 would be reached with 60 cores.

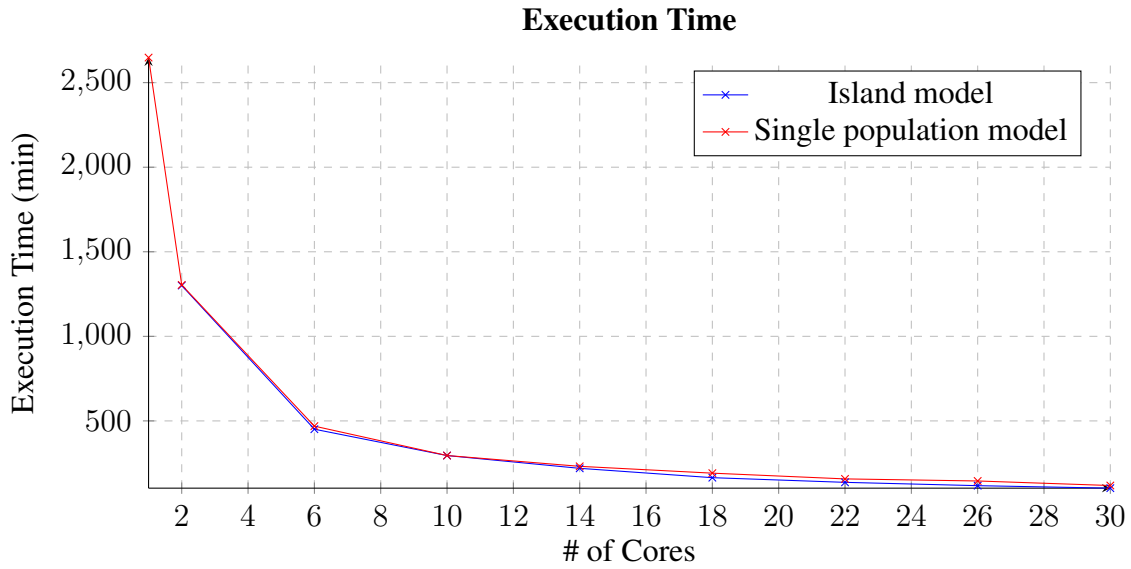


Figure 4.11: Execution times of models

	Single Population	Island Model
1 core	2648.22 mins	Not Applicable
2 cores	1303.87 mins	1300.43 mins
6 cores	469.07 mins	450.90 mins
10 cores	295.33 mins	295.08 mins
14 cores	231.42 mins	219.57 mins
18 cores	191.03 mins	164.35 mins
22 cores	157.10 mins	136.72 mins
26 cores	144.78 mins	117.09 mins
30 cores	117.95 mins	102.35 mins

Table 4.3: # of cores and execution time values of different PGA models

Another parameter for parallelism is the communication cost which increases with more cores. Without any communication cost, doubling the number of cores would have halved the execution time. However, according to the results specified in Table 4.3, this is not the case and this brings us to the speedup and the parallel efficiency terms. The speedup metric is calculated as follows:

$$S(p) = \frac{T_s}{T_p(p)} \tag{4.4}$$

where T_s is the execution time of the sequential algorithm and T_p is the execution time of the parallel algorithm with p cores, and

$$E(p) = \frac{S(p)}{p} \tag{4.5}$$

is the parallel efficiency of the parallel algorithm. By looking to these metrics of the PGA, we can discuss how efficient the parallel algorithm is. Any degradation in the performance of parallel algorithm will result in $E(p)$ being less than 1.

Performance of the parallel algorithm can be limited with a few factors, such as communication time and idle time of some cores [88]. Without these factors, efficiency would be 1 and speedup curve would be ideal linear, which are the cases for the ideal parallelization.

Figure 4.12 shows the speedup results and Figure 4.13 shows the performance results of our PGA experiments. Since communication cost increases when we increase the number of cores; the speedup curves in Figure 4.12 became distant from the ideal curve when executing PGA in our experiments with more cores. Also, population size was not always the exact multiple of the number of cores, so there were some idle cores during the execution of some experiments. Therefore, it can be clearly stated that parallel execution times of experiments were affected both communication time and idle time of some cores.

As a final result to discuss, we can compare parallel performances of both island and single population models of PGA. Table 4.3, Figure 4.12 and 4.13 demonstrate that the island model performed more efficiently than the single population model in terms of execution time, especially when the number of cores exceeds 10. This implies that at least one of the communication or the idle time was resulted better in the island model. Actually, it is more likely that both communication and idle

time were shortened. Because the communication workload of one master core in single population model was distributed in island model, which leads a decrease in communication cost.

In addition to communication cost, in some of the experiments that the population size was not the exact multiple of number of cores, the remaining idle cores were relatively smaller in the island model, since its population size was one quarter of the single population model. So the idle time of cores is also expected to decrease in the island model, since the workload was distributed more fairly than the single population model. Additionally, single population model with 26 cores has a solid decrease in the efficiency, that can be seen in Figure 4.13, as there were roughly 18 idle cores in the third fitness calculation step of each generation.

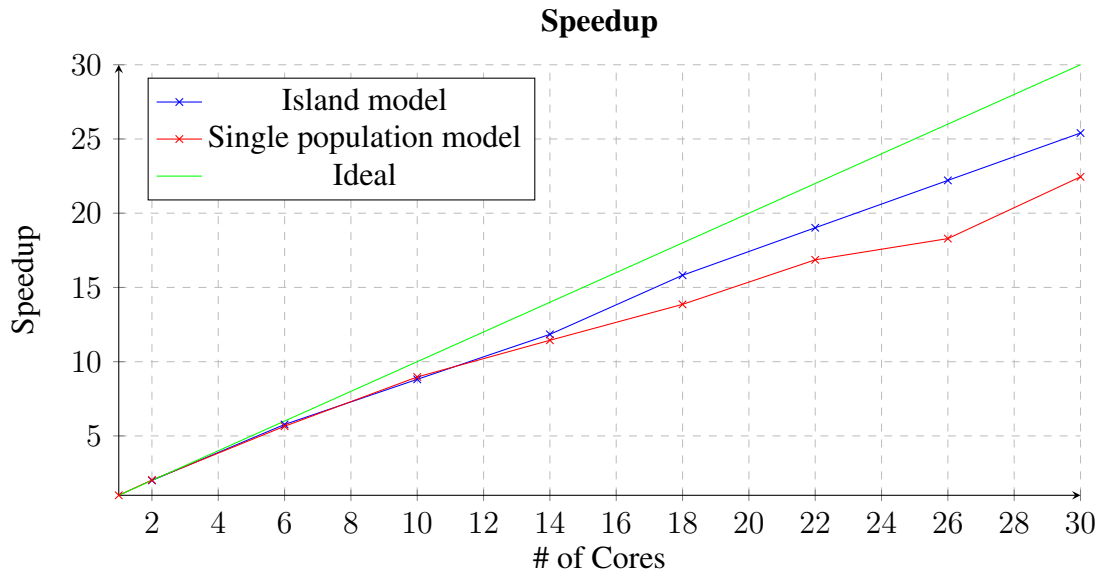


Figure 4.12: Speedup of parallelism

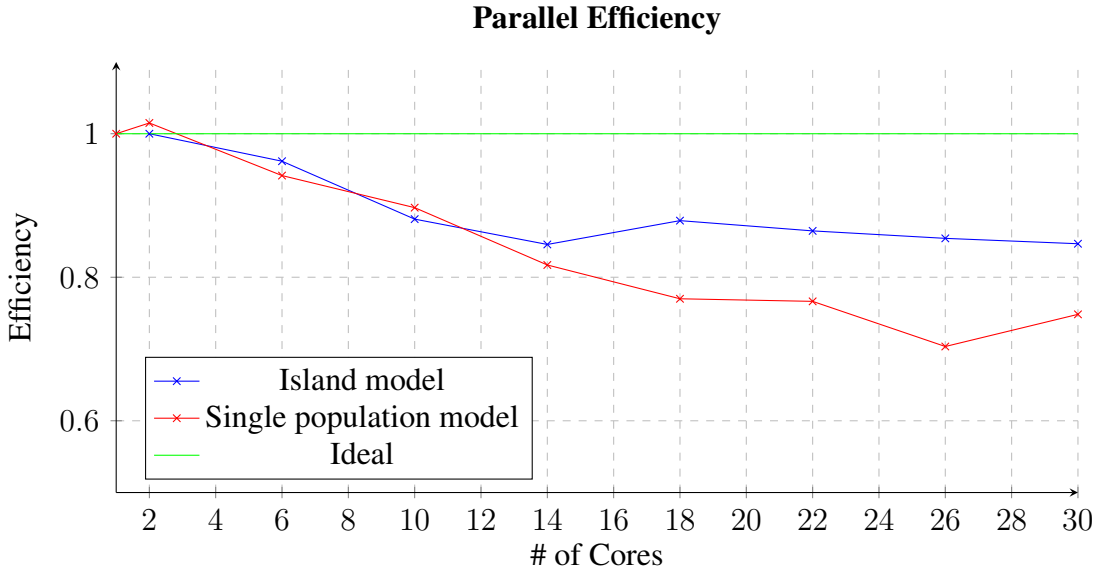


Figure 4.13: Parallel Efficiency

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this thesis, a platform for automated aerostructural shape optimization of 3D wings with GA is developed. The main focus of this thesis was the engineering designs and applications in which the shape optimization problem arises, such as geometry and mesh automation and multi-disciplinary parallel optimization.

The existing solutions for the multidisciplinary shape optimization [89, 90, 91, 92] do not propose optimization together with full automation of geometry and mesh generation, due to the computational cost of it. Also, separately CST and NURBS parameterization methods are used in the literature [14, 17, 90], but there is no known hybrid method yet. On the other hand, although genetic algorithm is one of the most common optimization techniques in the literature [92, 17], variations and parallelization of GA have not been compared previously in multidisciplinary shape optimization problems.

Development of an automated aerostructural shape optimization for 3D wings was our main objective. The proposed optimization technique applied in this thesis uses GA. However, since automation of geometry and mesh generation and aerostructural analysis take too much time, a parallel GA is implemented. The parallel GA is applied in both single population model and island model for fast optimization.

Firstly, we parameterize the 3D wing geometry to encode it for the optimization process. Next, mesh has to be generated on this parameterized geometry, where two different mesh is generated for both aerodynamic and structural solvers. Then, aero-

dynamic and structural analyzes are conducted to get how good the generated 3D wing is. Lastly, all these parametric 3D wing geometry generation, mesh generation from that geometry and aerostructural analyzes are automated.

Automation is achieved through the utilization of a Kulfan's CST parametrization method with NURBS surfaces. Developed hybrid method combines powerful design capabilities of CST with perfect modelling properties of NURBS surfaces. After designing with CST, 3D wing geometry is modelled with NURBS surface. By using this 3D wing geometry of NURBS surfaces, one structural mesh for aerodynamic analysis and one unstructured mesh for structural solver are generated automatically. Automation allows us to create fast geometry and as a result fast mesh on it, which is critically important because new geometry and mesh are generated for each individual of GA.

In order to analyze the performance of the proposed parallel GA models, various experiments were conducted. Initially, a single population model of GA was studied. Next, the island model of GA was studied and the convergence rates of both models were reported. We noticed that the island model performed better with our experiment setup when compared to the single island model. However, it is also realized that the migration is the key point of the convergence of the island model. Therefore, it can be said that the genetic diversity takes an important place in the convergence rate, which means that the diversity leads GA to find global best rather than local best.

Furthermore, the parallelization of the GA is bound to the population size of the generation, since fitness calculation of the each individual must be done before advancing to the next generation. Nonetheless, our experiments didn't reach the bound and thus, we got almost linear speedup with parallelism below that bound. Moreover, it can be said that developed parallel algorithm is scalable up to 60 cores, because the parallel efficiency of our PGA is expected to remain horizontal until the population size bound. Also by looking at the results of parallel experiments, it is shown that the execution time can be reduced by nearly 26 times, where it is shortened from 2 days to 100 minutes.

The main advantages and powerful aspects of the developed tool are that any objective function with penalty constraint and parallel GA model can easily be applied, and

that it is based on new hybrid geometry parameterization technique which allows fast, accurate and automated geometry and mesh generations. Experimental results particularly indicate that the parameterization, automation and parallelism make a strong impact for fast and better convergence on the 3D wing optimization process.

5.2 Future Work

We suggest that the parameter count of the proposed hybrid parameterization technique can be increased, and the convergence of the GA optimization can be discussed with this increased complexity. As an another suggestion that increases the computational complexity, constant wing parameters (span length, rib and spar count, etc.) can be added to chromosome as genes.

The parallel number of cores can be increased to the population size of GA, as it is ideal parallelism for the GA. Another future direction is that the different evolutionary optimization algorithms (particle swarm optimization, ant colony optimization, etc.) can be applied for an aerostructural 3D wing optimization problem. These different optimization methods can show different characteristics when reaching a better individual, so they can widen the horizon for the upcoming studies on the multidisciplinary shape optimization problems.

REFERENCES

[1] H. Samet, "An overview of quadtrees, octrees, and related hierarchical data structures," in *Theoretical Foundations of Computer Graphics and CAD* (R. A. Earnshaw, ed.), (Berlin, Heidelberg), pp. 51–68, Springer Berlin Heidelberg, 1988.

[2] G. E. Farin, *NURBS: From Projective Geometry to Practical Use*. Natick, MA, USA: A. K. Peters, Ltd., 2nd ed., 1999.

[3] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 2012.

[4] C. de Boor, *A Practical Guide to Spline*, vol. Volume 27. 01 1978.

[5] W. J. Gordon and R. F. Riesenfeld, "B-spline curves and surfaces," in *Computer aided geometric design*, pp. 95–126, Elsevier, 1974.

[6] K. J. Versprille, *Computer-aided Design Applications of the Rational B-spline Approximation Form.* PhD thesis, Syracuse, NY, USA, 1975. AAI7607690.

[7] L. Piegl, "Modifying the shape of rational b-splines. part 2: surfaces," *Computer-Aided Design*, vol. 21, no. 9, pp. 538 – 546, 1989.

[8] W. Tiller, "Rational b-splines for curve and surface representation," *IEEE Computer Graphics and Applications*, vol. 3, no. 6, pp. 61–69, 1983.

[9] D. F. Rogers, *An introduction to NURBS : with historical perspective*. San Francisco ; London : Morgan Kaufmann, 2001.

[10] L. Yang and X.-M. Zeng, "Bézier curves and surfaces with shape parameters," *International Journal of Computer Mathematics*, vol. 86, no. 7, pp. 1253–1263, 2009.

[11] N. P. Salunke, J. A. R. A., and S. Channiwala, "Airfoil parameterization techniques: A review," *American Journal of Mechanical Engineering*, vol. 2, no. 4, pp. 99–102, 2014.

[12] B. Kulfan and J. Bussoletti, "Fundamental parameteric geometry representations for aircraft component shapes," 09 2006.

[13] B. M. Kulfan, "Universal parametric geometry representation method," vol. 45, pp. 142–158, 01 2008.

[14] K. Lane and D. Marshall, "A surface parameterization method for airfoil optimization and high lift 2d geometries utilizing the cst methodology," 01 2009.

[15] D. Marshall, "Creating exact bezier representations of cst shapes," 06 2013.

[16] A. Sóbester and A. Forrester, *Aircraft Aerodynamic Design: Geometry and Optimization*. Aerospace Series, Wiley, 2014.

[17] E. Orman and G. Durmuş, "Comparison of shape optimization techniques coupled with genetic algorithm for a wind turbine airfoil," in *2016 IEEE Aerospace Conference*, pp. 1–7, March 2016.

[18] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[19] J. D. Bagley, *The behavior of adaptive systems which employ genetic and correlation algorithms*. PhD thesis, University of Michigan, 1967.

[20] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[21] K. A. De Jong, *Analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[22] M. Mitchell, *An Introduction to Genetic Algorithms*. Complex Adaptive Systems, The MIT Press, first printing. ed., 1996.

[23] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of an International Conference on Genetic Algorithms and their applications*, pp. 101–111, Hillsdale, New Jersey, 1985.

[24] J. F. C. M. Kimura, *An introduction to population genetics theory*. New York, Harper & Row [1970].

[25] D. Schlierkamp-Voosen and H. Mühlenbein, "Predictive models for the breeder genetic algorithm," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, 1993.

[26] A. Brindle, *Genetic algorithms for function optimization*. PhD thesis, The University of Alberta, 1981.

[27] J. Y. Suh and D. Van Gucht, *Distributed genetic algorithms*. Computer Science Department, Indiana Univ., 1987.

[28] H. Mühlenbein, "Parallel genetic algorithms, population genetics and combinatorial optimization," in *Workshop on Parallel Processing: Logic, Organization, and Technology*, pp. 398–406, Springer, 1989.

[29] D. E. Goldberg, B. Korb, K. Deb, *et al.*, "Messy genetic algorithms: Motivation, analysis, and first results," *Complex systems*, vol. 3, no. 5, pp. 493–530, 1989.

[30] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Urbana*, vol. 51, pp. 61801–2996.

[31] S. N. Sivanandam and S. N. Deepa, *Introduction to Genetic Algorithms*. Springer Publishing Company, Incorporated, 1st ed., 2007.

[32] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[33] W. M. Spears and K. A. De Jong, "An analysis of multi-point crossover," in *Foundations of genetic algorithms*, vol. 1, pp. 301–315, Elsevier, 1991.

[34] G. Syswerda, "Simulated crossover in genetic algorithms," in *Foundations of genetic algorithms*, vol. 2, pp. 239–255, Elsevier, 1993.

[35] Z. Michalewicz, T. Logan, and S. Swaminathan, "Evolutionary operators for continuous convex parameter spaces," in *Proceedings of the 3rd Annual conference on Evolutionary Programming*, pp. 84–97, World Scientific, 1994.

[36] N. Soni and T. Kumar, "Study of various mutation operators in genetic algorithms," 2014.

[37] D. FOGEL and I. N. N. Council, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. IEEE transactions on neural networks, IEEE Press, 1995.

[38] D. J. Cavicchio, *Adaptive search using simulated evolution*. PhD thesis, University of Michigan, 1970.

[39] M. J. Schutten and D. Torrey, "Genetic algorithms for control of power converters," in *Power Electronics Specialists Conference, 1995. PESC'95 Record., 26th Annual IEEE*, vol. 2, pp. 1321–1326, IEEE, 1995.

[40] L. J. Schmitt and M. M. Amini, "Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study," *European Journal of Operational Research*, vol. 108, no. 3, pp. 551–570, 1998.

[41] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," in *Machine Learning Proceedings 1995*, pp. 38–46, Elsevier, 1995.

[42] P. Grosso, "Computer simulations of genetic adaptation: Parallel subcomponent interaction in multilocus model," *Ph. D. Dissertation, University of Michigan*, 1985.

[43] D. Whitley, S. Rana, and R. Heckendorn, "The island model genetic algorithm: On separability, population size and convergence," *Journal of Computing and Information Technology*, vol. 7, 12 1998.

[44] R. Tanese, "Parallel genetic algorithm for a hypercube," 01 1987.

[45] R. Tanese, "Distributed genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*, (San Francisco, CA, USA), pp. 434–439, Morgan Kaufmann Publishers Inc., 1989.

[46] S. Xue, S. Guo, and D. Bai, "The analysis and research of parallel genetic algorithm," in *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 1–4, Oct 2008.

[47] A. L. Corcoran and R. L. Wainwright, "A parallel island model genetic algorithm for the multiprocessor scheduling problem," in *SAC*, vol. 94, pp. 483–487, Citeseer, 1994.

[48] M. Gorges-Schleuter, "Explicit parallelism of genetic algorithms through population structures," in *PPSN*, 1990.

[49] D. Whitley and T. Starkweather, "Genitor ii: a distributed genetic algorithm," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 2, no. 3, pp. 189–214, 1990.

[50] T. Starkweather, D. Whitley, and K. Mathias, "Optimization using distributed genetic algorithms," in *Parallel Problem Solving from Nature* (H.-P. Schwefel and R. Männer, eds.), (Berlin, Heidelberg), pp. 176–185, Springer Berlin Heidelberg, 1991.

[51] C. C. Pettey and M. R. Leuze, "A theoretical investigation of a parallel genetic algorithm," in *ICGA*, 1989.

[52] E. Cantú-Paz, "A survey of parallel genetic algorithms," *CALCULATEURS PARALLELES*, vol. 10, 1998.

[53] A. J. Chipperfield and P. Fleming, "Parallel genetic algorithms: A survey," 1994.

[54] R. O. Winder, "Partitions of n-space by hyperplanes," *SIAM Journal on Applied Mathematics*, vol. 14, no. 4, pp. 811–818, 1966.

[55] R. P. I. I. P. Laboratory and D. Meagher, *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. 1980.

[56] D. Meagher, "Geometric modeling using octree encoding," *Computer graphics and image processing*, vol. 19, no. 2, pp. 129–147, 1982.

[57] H. Samet, "The quadtree and related hierarchical data structures," *ACM Comput. Surv.*, vol. 16, pp. 187–260, June 1984.

[58] R. Castro, T. Lewiner, H. Lopes, G. Tavares, and A. Bordignon, "Statistical optimization of octree searches," in *Computer Graphics Forum*, vol. 27, pp. 1557–1566, Wiley Online Library, 2008.

[59] S. Narasimhan, R.-P. Mundani, and H.-J. Bungartz, "An octree-and a graph-based approach to support location aware navigation services.," in *PSC*, pp. 24–30, 2006.

[60] M. G. Choi, E. Ju, J.-W. Chang, J. Lee, and Y. J. Kim, "Linkless octree using multi-level perfect hashing," in *Computer Graphics Forum*, vol. 28, pp. 1773–1780, Wiley Online Library, 2009.

[61] P. Bhattacharya, "Efficient neighbor finding algorithms in quadtree and octree," Master's thesis, Indian Institute of Technology, Kanpur, 2001.

[62] J. Bonet and J. Peraire, "An alternating digital tree (adt) algorithm for 3d geometric searching and intersection problems," *International Journal for Numerical Methods in Engineering*, vol. 31, no. 1, pp. 1–17, 1991.

[63] D. Knuth, "The art of computer programming, sorting and searching, vol. 3. 1973."

[64] U. Naumann and O. Schenk, *Combinatorial Scientific Computing*. Chapman & Hall/CRC, 1st ed., 2012.

[65] M. Aftosmis, J. Melton, and M. Berger, "Adaptation and surface modeling for cartesian mesh methods," in *12th Computational Fluid Dynamics Conference*, p. 1725, 1995.

[66] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[67] R. L. Carmichael and L. Erickson, "Pan air - a higher order panel method for predicting subsonic or supersonic linear potential flows about arbitrary configurations," 06 1981.

[68] D. Chao and C. Van Dam, "Airfoil drag prediction and decomposition," *Journal of Aircraft*, vol. 36, no. 4, pp. 675–681, 1999.

[69] G. R. Saaris, *A502I User's Guide-PAN AIR Technology Program for Solving Potential Flow about Arbitrary Configurations*.

[70] A. D. Young, "Boundary layers," *NASA STI/Recon Technical Report A*, vol. 91, 1989.

[71] M. Drela, "Xfoil: An analysis and design system for low reynolds number airfoils," in *Low Reynolds Number Aerodynamics* (T. J. Mueller, ed.), (Berlin, Heidelberg), pp. 1–12, Springer Berlin Heidelberg, 1989.

[72] "SAPeda.." `http://www.caeeda.com/index.php/en/products/sapeda`. (online; accessed: 30.08.2019).

[73] E. Oktay, H. Akay, and O. Merttopcuoglu, "Parallelized structural topology optimization and cfd coupling for design of aircraft wing structures," *Computers & Fluids*, vol. 49, no. 1, pp. 141 – 145, 2011.

[74] E. Oktay, H. U. Akay, and O. T. Sehitoglu, "Three-dimensional structural topology optimization of aerial vehicles under aerodynamic loads," *Computers & Fluids*, vol. 92, pp. 225 – 232, 2014.

[75] R. v. Mises, "Mechanik der festen körper im plastisch- deformablen zustand," *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, vol. 1913, pp. 582–592, 1913.

[76] E. Oktay, A. Arpacı, O. Şehitoğlu, and H. U. Akay, "A parallel aerostructural shape optimization platform for airplane wings," 05 2019.

[77] O. C. S.A.S., "Open cascade technology," 2018. https://www.opencascade.com/doc/occt-7.2.0/overview/html/index.html.

[78] L. P. Chew, "Constrained delaunay triangulations," *Algorithmica*, vol. 4, no. 1-4, pp. 97–108, 1989.

[79] "MESHeda.." `http://www.caeeda.com/index.php/en/products/mesheda`. (online; accessed: 30.08.2019).

[80] "CAEeda.." `http://www.caeeda.com`. (online; accessed: 30.08.2019).

[81] E.-G. Talbi, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[82] J. Humeau, A. Liefooghe, E.-G. Talbi, and S. Verel, "Paradiseo-mo: From fitness landscape analysis to efficient local search algorithms," *Journal of Heuristics*, vol. 19, no. 6, pp. 881–915, 2013.

[83] A. Liefooghe, L. Jourdan, and E.-G. Talbi, "A unified model for evolutionary multi-objective optimization and its implementation in a general purpose software framework," in *2009 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM)*, pp. 88–95, IEEE, 2009.

[84] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer, "Evolving objects: A general purpose evolutionary computation library," *Artificial Evolution*, vol. 2310, pp. 829–888, 2002.

[85] K. Rasheed, "An adaptive penalty approach for constrained genetic-algorithm optimization," in *Proceedings of the Third Annual Genetic Programming Conference*, pp. 584–590, Morgan Kaufmann Publishers, 1998.

[86] H. U. Akay, A. S. Baddi, and E. Oktay, "Large-scale parallel computation of fluid-solid interaction problems for aeroelastic flutter prediction," no. AIAC 2005-002, 2005.

[87] "SINeda.." `http://www.caeeda.com/index.php/en/products/sineda`. (online; accessed: 30.08.2019).

[88] E. Cantú-Paz and D. E. Goldberg, "Efficient parallel genetic algorithms: theory and practice," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2, pp. 221 – 238, 2000.

[89] C. M. Boozer, "Multidisciplinary shape optimization of a composite blended wing body aircraft," Master's thesis, University of South Carolina, 2017.

[90] J. Samareh, "Multidisciplinary aerodynamic-structural shape optimization using deformation (massoud)," in *8th Symposium on Multidisciplinary Analysis and Optimization*, p. 4911, 2000.

[91] J. Mariens, "Wing shape multidisciplinary design optimization," Master's thesis, Delft University of Technology, 2012.

[92] T. Kumano, S. Jeong, S. Obayashi, Y. Ito, K. Hatanaka, and H. Morino, "Multidisciplinary design optimization of wing shape for a small jet aircraft using kriging model," in *44th AIAA Aerospace Sciences Meeting and Exhibit*, p. 932, 2006.