

**ON OPTIMIZATION OF SENSOR SELECTION FOR  
AIRCRAFT GAS TURBINE ENGINES**

**RAMGOPAL MUSHINI**

Bachelor of Engineering in Electronics and Instrumentation Engineering

Nagarjuna University, India

July, 1999

Submitted in partial fulfillment of requirements for the degree

**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**

at the

**CLEVELAND STATE UNIVERSITY**

December, 2004

This thesis has been approved

For the Department of **ELECTRICAL AND COMPUTER ENGINEERING**

And the College of Graduate Studies by

---

Dr. Dan Simon, Thesis Committee Chairperson

---

Department/Date

---

Dr. Zhiqiang Gao, Thesis Committee Member

---

Department/Date

---

Dr. Sridhar Ungarala , Thesis Committee Member

---

Department/Date

*To mom, dad, sudha and sunitha...*

## ACKNOWLEDGEMENT

I would like to express my sincere indebtedness and gratitude to my thesis advisor Dr. Dan Simon, for the ingenious commitment, encouragement and stimulating suggestions provided to me over the entire course of this thesis.

I would also like to thank my committee members Dr. Sridhar Ungarala and Dr. Yongjian Fu for their support and advice.

I would like to thank Sanjay Garg and Donald Simon at the NASA Glenn Research Center, and the NASA Aviation Safety Program, for generously providing funding for this work.

Finally, I wish to thank my lab mates at the Embedded Control Systems Research Laboratory and my roommates SaiKiran and Anand for their encouragement and intellectual input during the entire course of this thesis without which this work wouldn't have been possible.

# **ON OPTIMIZATION OF SENSOR SELECTION FOR AIRCRAFT GAS TURBINE ENGINES**

**RAMGOPAL MUSHINI**

## **ABSTRACT**

Many science and management problems can be formulated as global optimization problems. Conventional optimization methods that make use of derivatives and gradients are not, in general, able to locate or identify the global optimum. Sometimes these problems can be solved using exact methods like brute force. Unfortunately these methods become computationally intractable because of multidimensional search spaces. Hence the application of heuristics for a class of problems that incorporates knowledge about the problem helps solve optimization problems.

Sensor selection optimization can lead to significant improvements in the controllability and observability of a dynamic system. The aim of this research is to investigate optimal or alternate measurement sets for the problem of aircraft gas turbine engine health parameter estimation. The performance metric is defined as a function of the steady state error covariance and the cost of the selected sensors. A brute force search for the best sensor set is too computationally expensive. Therefore a probabilistic approach is used to perform a search for a near-optimal sensor set. In view of the need for global optimization of sensor selection for health parameter estimation, a genetic algorithm is also developed. A genetic approach will perform a search for a near-optimal sensor set. This will allow the health of the aircraft gas turbine engine to be estimated using fewer sensors while still obtaining an acceptable estimation error covariance, thereby minimizing the financial cost of the acceptable sensor set.

# TABLE OF CONTENTS

LIST OF FIGURES .....	ix
LIST OF TABLES .....	xi
<b>CHAPTER I -- INTRODUCTION.....</b>	<b>1</b>
1.1 LITERATURE SURVEY .....	2
1.2 THESIS ORGANIZATION .....	3
<b>CHAPTER II -- AIRCRAFT TRUBOFAN ENGINES.....</b>	<b>4</b>
2.1 INTRODUCTION.....	4
2.2 JET PROPULSION THEORY .....	5
2.3 GAS TURBINE ENGINES .....	6
2.3.1 Gas Turbine Engine Components .....	8
2.3.2 Types of Jet Engines .....	12
2.4 ENGINE PERFORMANCE .....	17
2.5 TURBOFAN ENGINE MODEL- MAPSS .....	19
2.6 ENGINE’S STATE SPACE MODEL .....	22
<b>CHAPTER III – GENETIC ALGORITHMS.....</b>	<b>26</b>
3.1 GENETIC ALGORITHMS .....	26
3.2 BASIC COMPONENTS.....	30
3.3 THE ELEMENTS OF A GENETIC ALGORITHM.....	32
3.3.1 Initial Population.....	32

3.3.2 Termination.....	33
3.3.3 Selection.....	33
3.3.4 Crossover .....	36
3.3.5 Mutation.....	37
3.3.6 New Population.....	38
3.3.7 Diversity Maintenance .....	39
3.4 THE SCHEMA THEOREM .....	40
3.5 EXPLORATION AND EXPLOITATION .....	45
<b>CHAPTER IV—SENSOR SELECTION .....</b>	<b>48</b>
4.1 THE SYSTEM MODEL.....	48
4.2 PROBABILISTIC APPROACH TO SENSOR SELECTION .....	50
4.3 GENETIC APPROACH TO SENSOR SELECTION .....	53
4.4 EIGENVECTOR APPROACH TO SENSOR SELECTION.....	57
<b>CHAPTER V -- RESULTS .....</b>	<b>60</b>
5.1 AIRCRAFT TURBOFAN ENGINE SENSOR SELECTION .....	60
5.2 BRUTE FORCE SENSOR SELECTION .....	61
5.3 PROBABILISTIC SEARCH FOR SENSOR SELECTION .....	65
5.4 EIGENVECTOR APPROACH FOR SENSOR SELECTION.....	67
5.5 GENETIC APPROACH TO SENSOR SELECTION .....	68
5.5.1 Genetic Algorithm Results using Identical Sensor Costs .....	69
5.5.2 Genetic Algorithm Results using Relative Sensor Costs.....	74

<b>CHAPTER VI – CONCLUSIONS AND FUTURE WORK .....</b>	<b>81</b>
6.1 CONCLUSIONS .....	81
6.2 FUTURE WORK .....	84
<b>BIBLIOGRAPHY .....</b>	<b>86</b>

## LIST OF FIGURES

FIGURE .....	PAGE
2.1 An aeolipile.....	5
2.2 An aircraft jet engine .....	7
2.3 A twin spool gas turbine .....	9
2.4 A modern jet engine with axial compressor .....	10
2.5(a) A turbojet.....	13
2.5(b) A turbojet with afterburner.....	13
2.6 A turboprop engine .....	14
2.7 A turboshaft engine.....	15
2.8 A low-bypass turbofan.....	16
2.9 A high-bypass turbofan.....	16
2.10 Schematic of turbofan engine model .....	19
2.11 MAPSS block diagram .....	20
2.12 CLM with engine components.....	21
3.1 Decoding and encoding .....	28
3.2 Evolutionary search cycle for global optimization .....	29
3.3 A pseudocode of genetic algorithm .....	29
3.4 Roulette wheel selection .....	34
3.5 One-point crossover of binary strings.....	37
5.1 Histogram of the distribution of 2000 sensor sets with relative cost .....	65
5.2 Sensor set cost vs cost frequency .....	66
5.3 GA run with setdiff crossover using identical cost for Sensors.....	70

5.4	GA run with two-point crossover using identical cost for Sensors .....	71
5.5	GA run with uniform crossover using identical cost for Sensors .....	72
5.6	GA run with segmented crossover using identical cost for Sensors .....	74
5.7	GA run with setdiff crossover using relative cost for Sensors.....	76
5.8	GA run with two-point crossover using relative cost for Sensors .....	77
5.9	GA run with uniform crossover using relative cost for Sensors .....	79
5.10	GA run with segmented crossover using relative cost for Sensors .....	80

## LIST OF TABLES

<b>TABLE</b> .....	<b>PAGE</b>
I. Coefficients in the Expansion .....	61
II. Relative sensor costs .....	63
III. Least cost sensor sets with two different cost models .....	64
IV. Top 10 sensor sets with sensor cost of 1000 .....	64
V. Top 8 sensor sets with relative sensor cost .....	64
VI. Sensor selection probabilities using random search method .....	66
VII. Sensor sets using random search algorithm .....	67
VIII. Sensor sets using probabilistic search algorithm .....	67
IX. Sensor sensitivities for MAPSS model .....	68
X. Least cost sensor sets obtained using setdiff crossover with identical cost for sensors .....	69
XI. Least cost sensor sets obtained using two-point crossover with identical cost for sensors .....	71
XII. Least cost sensor sets obtained using uniform crossover with identical cost for sensors .....	73
XIII. Least cost sensor sets obtained using segmented crossover with identical cost for sensors .....	74
XIV. Least cost sensor sets obtained using setdiff crossover with relative cost for sensors .....	75

XV. Least cost sensor sets obtained using two-point crossover with relative cost for sensors .....	77
XVI. Least cost sensor sets obtained using uniform crossover with relative cost for sensors .....	78
XVII. Least cost sensor sets obtained using segmented crossover with relative cost for sensors .....	80
XVIII: Least cost sensor sets obtained with a sensor cost of 1000 for each sensor .....	82
XIX: Least cost sensor sets obtained with relative cost of each sensor .....	82
XX: Relative cost optimization with identical sensor cost .....	83
XXI Relative cost optimization with relative sensor cost .....	83
XXII: Sensors in the best sensor set .....	84
XXIII: Sensors eliminated from the best sensor set.....	84



# **CHAPTER I**

## **INTRODUCTION**

Recently sensor selection algorithms have provided improved performance at the cost of computational demand. As the number of measurements and sensors increases, managing the selection of sensors becomes inevitable with such sensor selection algorithms. So an algorithm is needed which can optimize the selection of sensors leading to significant improvements in the controllability and observability of a dynamic system.

A physical dynamical system model is not perfect, especially one that has been linearized, but the accuracy within a neighborhood of the operating point may be acceptable. Also, sensor measurements are limited in their accuracy because of noise and resolution. Being able to quantify these limitations is a key to achieving good estimation.

A technique that is commonly used to model system degradation while still representing the system as time-invariant is to introduce biases into the linearized model. These biases model changes in the system state and output values due to system degradation. These biases can be appended to the system's state vector in which case they are sometimes called extended parameters [1]. In general, measurements change when the parameters associated with them change. It is necessary to determine the best measurement sets to minimize a combination of overall parameter uncertainty and cost using steady state estimation error covariance analysis.

## **1.1 Literature Survey**

Various algorithms have been proposed for sensor optimization of a jet engine. A search over all possible sensor combinations grows exponentially with the number of sensors; in our case we have a total of 22 sensors where each of the 11 sensors can be used twice. So a search over 177146 combinations (a brute force technique) will be unacceptable and a more efficient method is needed. The Eigenvalue/Minimum Sensors algorithm [3] results in a sensor combination with the fewest number of sensors that produces all positive eigenvalues in the error covariance matrix is the best sensor combination. In the Matrix Norm Algorithm, a sensor selection algorithm is used to minimize the norm of the inverse of the error covariance for the sensor combination whose inverse covariance can be calculated. Then check if the norm of the covariance is within a predefined boundary, thus selecting the combination that uses fewest sensors while keeping the covariance within the boundary [2]. In another approach the concept of randomization and super heuristics

is used to develop a computationally efficient model for generating an optimal sensor set [3].

## **1.2 Thesis Organization**

This thesis discusses two different concepts of finding an optimal sensor set. First, we randomly generate a limited number of sensor sets, evaluate the metric of each sensor set, and then estimate the probabilities that a given sensor is in a “good” sensor set. Then a directed search is conducted using the previously obtained probabilities in order to find a near-optimal sensor set. The second method is an application of genetic algorithms to find the near optimal sensor set.

In Chapter 2 we discuss principles of the turbofan engine and its operation, the turbofan model MAPSS used for this research, and the states, control inputs, health parameters and outputs of the system.

Chapter 3 describes the basic concept of genetic algorithms, the use of genetic algorithms for solving optimization problems, and the mathematical foundation which allows a genetic algorithm to solve optimization problems.

Chapter 4 introduces the sensor selection optimization problem for the turbofan engine model used for this research. The implementation of the directed search algorithm and genetic algorithms to sensor selection optimization for near optimal sensor set is described.

Chapter 5 discusses the results obtained using implementations of both algorithms to the sensor selection optimization problem and conclusions made thereafter.

Chapter 6 concludes with some conclusions and future work on selection optimization.

## **CHAPTER II**

### **AIRCRAFT TURBOFAN ENGINES**

#### **2.1 Introduction**

An application of optimizing the performance of a jet engine should be backed up with knowledge of the operation of turbofan engines. The history of gas turbine engines as a viable energy conversion device began with Sir Frank Whittle of Great Britain and Hans von Ohain in Germany. The success of the gas turbine in replacing the reciprocating engine as a power plant for high-speed aircraft is well known. The development of the gas turbine was less rapid as a stationary power plant in competition with steam for the generation of electricity and with the spark-ignition and diesel engines in transportation and stationary applications. Nevertheless, applications of gas turbines are now growing at a rapid pace as research and development produces performance and reliability increases and economic benefits [4]. In Section 2.2, jet propulsion theory is being discussed and continued with discussion of gas turbine engines in Section 2.3. The performance of the

jet engine is studied in Section 2.4. Section 2.5 describes the jet engine model implemented for this research.

## 2.2 Jet Propulsion Theory

The principle of jet propulsion was demonstrated by Hero of Alexandria as long ago as the first century AD. He demonstrated jet power in a machine called an aeolipile as in Figure 2.1 [5], a heated, water filled ball with nozzle spun as steam escaped. It consisted of a closed vessel in the shape of a sphere into which steam under pressure was introduced. When the steam escaped from two bent tubes mounted opposite one another on the surface of the sphere, the tubes became jet nozzles. A force was created at the nozzles that caused the sphere to rotate about an axis.

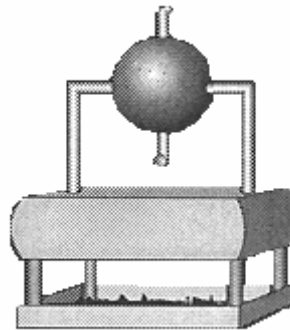


Figure 2.1: An aeolipile

The principle behind this phenomenon was not fully understood until 1690 A.D. when Sir Isaac Newton in England formulated the principle of Hero's jet propulsion "Aeolipile" in scientific terms as his third law: Every action produces a reaction, equal in force and opposite in direction. This principle was applied by Newton to build a forerunner of today's jet engines. Newton's invention was a wagon which he tried to propel by a steam

jet pointing rearward from a steam boiler mounted on the wagon. Then the earliest 'steam engine' on record was from Sir Frank Whittle. But the jet engine did not become a practical possibility until 1930. Later Sir Frank Whittle patented the design of his first reaction motor suitable for aircraft propulsion. The early jet engines were designed solely for aircraft propulsion. However, development was rapid and the range of applications has widened to include ships, hovercraft, power stations and industrial installations, all of which benefit from the jet engine's inherent qualities of high power, small size and low weight. All the jet engines designed operate on the same principle. Although there are piston engines which work similar to the jet engines, they are rarely used in aircraft. Both the engines convert the energy of the expanding gases in to mechanical force (thrust). The major drawback in the piston engines is that they impart relatively small acceleration to a large mass of air compared to the large acceleration of small mass of air in jet engine case.

### **2.3 Gas Turbine Engines**

A turbine is any kind of spinning device that uses the action of a fluid to produce work. The name "gas turbine" is somewhat misleading, because to many it implies a turbine engine that uses gas as its fuel. Actually a gas turbine, as shown in Figure 2.2 [6], has a compressor to draw in and compress gas; a combustor to add fuel to heat the compressed air; and a turbine to extract power from the hot air flow. The gas turbine is an internal combustion (IC) engine employing a continuous combustion process.

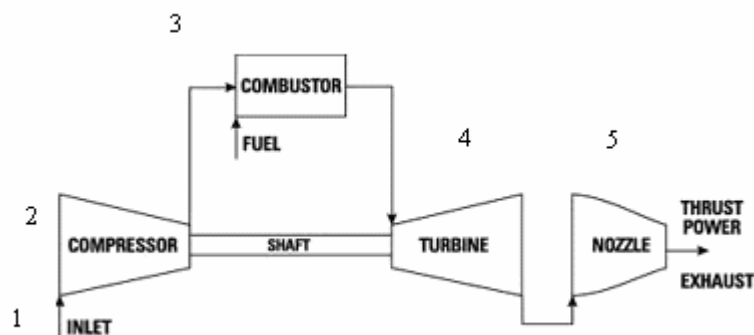


Figure 2.2: An aircraft jet engine

The gas turbine engine is the turbine engine that is operated by a gas, which is the product of the combustion that takes place when a fuel is mixed and burned with the air passing through the engine. Jet engine is the other name for gas turbine engine. The operation of the turbojet; simplest gas turbine engine is similar to that of the aeolipile. However, the sphere is replaced by a can-like horizontal container open at both ends. This horizontal container is called an engine case. This engine case has five major sections: inlet, compressor, combustor (burner), turbine and outlet (jet nozzle). Large quantities of air enter the engine through the inlet. The air entered then passes through the compressor to attain high pressures. The temperature of the high pressure air is increased by burning (mixing) it with the fuel in the combustor. The combustion results in high velocity hot gases which pass through the turbines, generating power to run the compressor. These high velocity hot gases coming out of the turbine are exhausted to the outside through the outlet (jet nozzle) creating a thrust to the move the horizontal container (engine) forward.

### **2.3.1 Gas Turbine Engine Components**

The gas turbine and its operation can be understood by considering three major components: the compressor, the combustor and the turbine. The features and characteristics of these will be touched on briefly.

#### **Inlet**

The main task of the inlet is to straighten out the flow of air that enters into the engine through it, making it uniform and without much turbulence. This is important because compressors and fans need to be fed distortion-free air. The inlet is positioned just before the compressor. There are different types of inlets based on the speed of the aircraft like subsonic inlets, supersonic inlets and hypersonic inlets.

#### **Compressors**

The compressor is used to increase the pressure of air entering the through the inlet. The compressor components are connected to the turbine by a shaft in order to allow the turbine to turn the compressor. The air is forced through several rows of both spinning and stationary blades. As the air passes each row, the available space is greatly reduced, and so the air that exits this phase is thirty or forty times higher in pressure than it was outside the engine. The temperature of the air also gets increased because of the increase in pressure. Axial flow compressor and centrifugal compressor are the two main types of compressors used in turbofan engines. A single shaft gas turbine has only one shaft connecting the compressor and turbine components. A twin spool gas turbine as shown in Figure 2.3 [6] has two concentric shafts, a longer one connecting a low pressure

compressor to a low pressure turbine (the low spool) which rotates inside a shorter, larger diameter shaft. The shorter, larger diameter shaft connects the high pressure turbine with the higher pressure compressor (the high spool) which rotates at higher speeds than the low spool. A triple spool engine would have a third, intermediate pressure compressor-turbine spool. Gas turbine compressors are either centrifugal or axial, or can be a combination of both. Centrifugal compressors with compressed air output around the outer perimeter of the machine are robust, generally cost less and are limited to pressure ratios of 6 or 7 to 1.

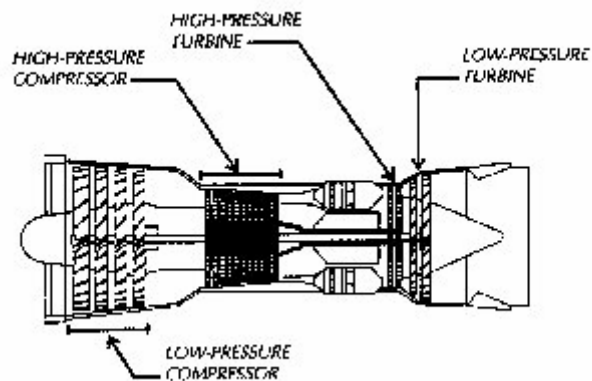


Figure 2.3: A twin spool gas turbine

Centrifugal compressors are found in early gas turbines or in modern, smaller gas turbines. The more efficient, higher capacity axial flow compressors with compressed air output directed along the center line of the machine are used in most gas turbines as shown in Figure 2.4 [6]. An axial compressor is made up of a relatively large number of stages, each stage consisting of a row of rotating blades (airfoils) and a row of stationary blades (stators), arranged so that the air is compressed as it passes through each stage.

## Turbines

Turbines are generally easier to design and operate than compressors, since the hot air flow is expanding rather than being compressed. The power used to drive the compressors is obtained from turbines. The turbine extracts the energy of the high temperature gas flow coming out of the burner by rotating the blades. This energy is transferred to the compressors by connecting shafts. The air leaving the turbine has low temperature and pressure when compared with the air coming out of the burner because of the energy extraction. Turbine blades must be made of special materials that can withstand the heat, or they must be actively cooled. There can be multiple turbine stages for driving different parts of the engine independently like compressor, fan (turbofan) or propeller (turboprop). Axial flow turbines will require fewer stages than an axial compressor. There are some smaller gas turbines that utilize centrifugal turbines (radial inflow), but most utilize axial turbines.

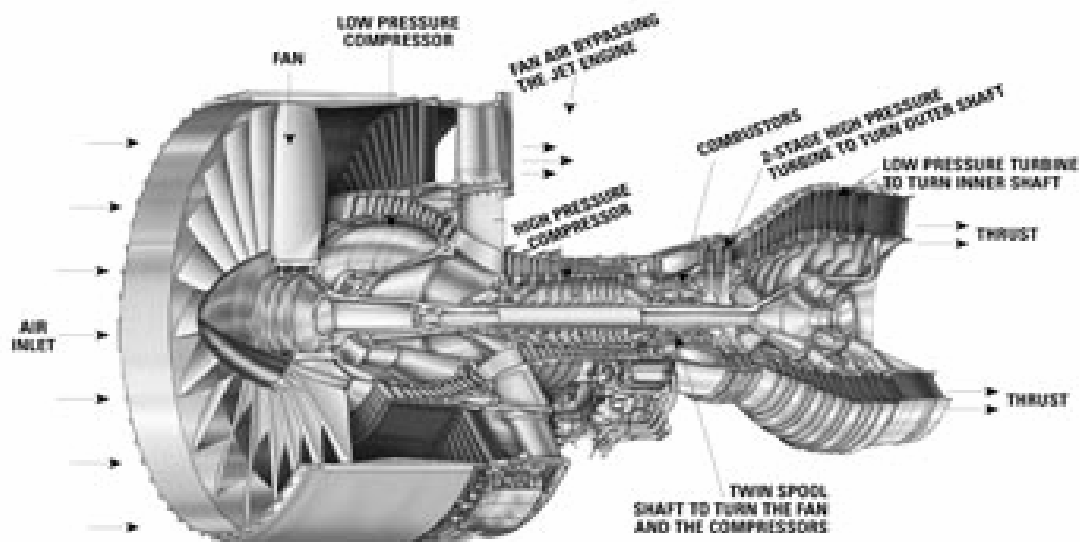


Figure 2.4: A Modern jet engine with axial compressors

## **Combustors**

A combustor consists of at least three basic parts: a casing, a flame tube and a fuel injection system. The casing must withstand the cycle pressures and may be a part of the structure of the gas turbine. It encloses a relatively thin-walled flame tube within which combustion takes place, and a fuel injection system. It is the component in which the actual reaction (combustion) takes place. The high pressure hot air coming out of the compressor is combined with the fuel and burned for combustion. The combustion results in very high temperature gases with high velocities. These high temperature exhaust gases are used to drive the turbine. Combustors, placed just after the compressors, are made from materials that can withstand the high temperatures of combustion. Annular, can, and can-annular combustors are the three different types of combustors mostly used.

A successful combustor design must satisfy many requirements and has been a challenge from the earliest gas turbines of Whittle and von Ohain. The relative importance of each requirement varies with the application of the gas turbine, and of course, most design requirements reflect concerns over engine costs, efficiency, and the environment.

## **Nozzle**

The actual thrust required to move the engine forward is produced in this specially shaped tube called a nozzle which is positioned after the turbine stage in the engine, through which hot gases flow. Air from the turbine, once cooled and expanded, is vented out the back of the engine. Nozzles are designed to maximize thrust, since venting hot air does not provide nearly as much thrust as venting fast-moving cool air. Like the air leaving a sphere of an aeolipile described in Section 2.2, the speed and flow rate of the air

leaving the nozzle provides the airplane with thrust. The inside walls of the nozzle are shaped so that the exhaust gases continue to increase their velocity as they travel out of the engine.

### **2.3.2 Types of Jet Engines**

"Jet engine" is often used as a generic name for a variety of engines, including the turbojet, turboprop, turbofan and ramjet. All these engines operate by the same basic principles, but each has its own distinct advantages and disadvantages. The basic operating principle of all jet engines is, forcing incoming air into a tube where the air is compressed, mixed with fuel, burned, and exhausted at high speed to generate thrust.

#### **Turbojet**

A simple turbojet as shown in Figure 2.5(a) [7] works by compression of incoming air. If uncompressed, the air-fuel mixture won't burn and the engine can't generate any thrust. Almost every jet engine employs a section of compressors, consisting of rotating blades, which slow the incoming air to create a high pressure. This compressed air is then forced into a combustion section where it is mixed with fuel and burned. As the high-pressure gases are exhausted, they are passed through a turbine section consisting of more rotating blades. In this region, the exhausting gases turn the turbine blades which are connected by a shaft to the compressor blades at the front of the engine. Thus, the exhaust turns the turbines which turn the compressors to bring in more air and keep the engine going. The

combustion of gases then continues to expand out through the nozzle creating a forward

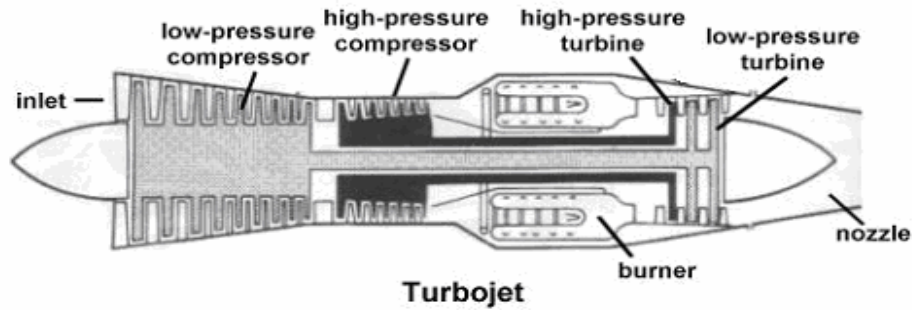


Figure 2.5(a): A turbojet

thrust. An afterburner is attached to a turbojet, as shown in Figure 2.5(b), to generate significant boost in thrust. It is simply a long tube placed in between the turbine and the nozzle in which additional fuel is added and burned.

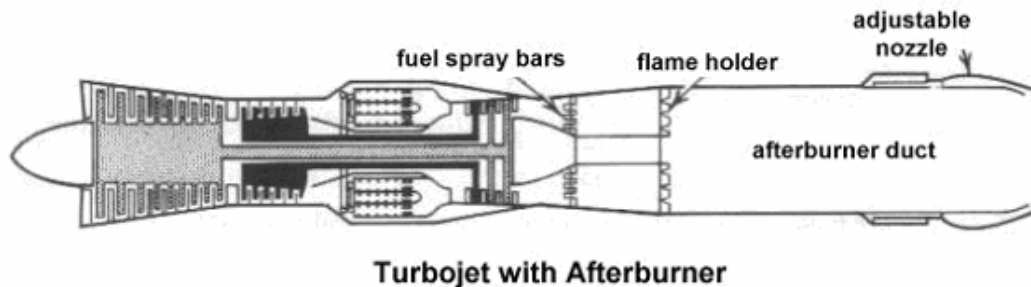


Figure 2.5(b): A turbojet with an afterburner

The afterburner increases the temperature of the gas ahead of the nozzle. The result of this increase in temperature is an increase of about 40 percent in thrust at takeoff and a much larger percentage at high speeds once the plane is in the air [8]. Afterburners greatly increase fuel consumption, so aircraft can only use them for brief periods.

## Turboprop

A turboprop engine in Figure 2.6 [7] is a jet engine attached to a propeller. The turbine at the back is turned by the hot gases and this turns a shaft that drives the propeller. The turboprop engine consists of a compressor, combustion chamber, and turbine, the air and gas pressure is used to turn the turbine, which then creates power to drive the compressor. Turboprop has better propulsion efficiency at flight speeds below about 500 miles per hour [8]. Now-a-days turboprops engines are equipped with propellers that have a smaller diameter but a larger number of blades for efficient operation at much higher flight speeds. These engines have greater fuel efficiency when compared to other jet engines. The drawbacks of turboprops are noise and vibration generated by the propeller and limitation to subsonic flight.

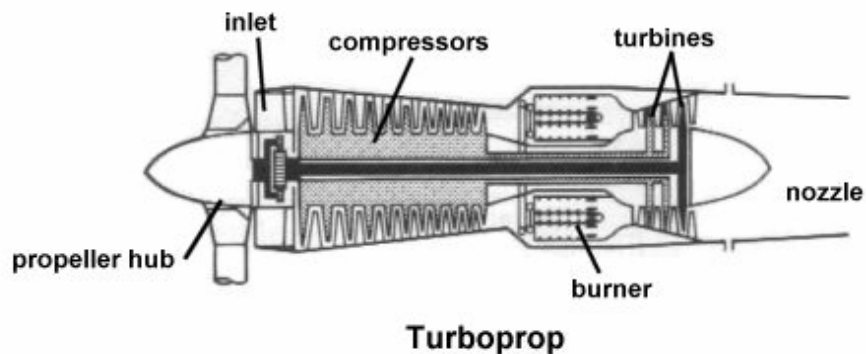


Figure 2.6: A turboprop engine

## Turboshaft

The turboprop engine in which the shaft of the free turbine is used to drive something other than the propeller is called the turboshaft. The other things that can be driven by the

free turbine shaft are the rotor of a helicopter, boats, ships, trains and automobiles. The shaft turbine engine is the other name for the turboshaft engine as shown in Figure 2.7 [7].

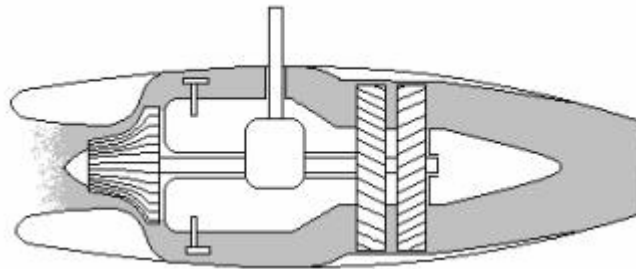


Figure 2.7: A turboshaft engine

Both the turboprop and turboshaft engines are more complicated and heavier than the turbojet engine. They produce more thrust at low subsonic speeds. Their propulsive efficiency (output divided by input) decreases as the speed increases, whereas it increases in the turbojet case.

### **Turbofan**

A turbofan engine shown in Figures 2.8 and 2.9 [7] have a large fan at the front, which sucks in air. The purpose of the fan is to drive large volume of air through the outer ducts that go around the engine core. Most of the air flows around the outside of the engine, making it quieter and giving more thrust at low speeds. Although this bypassed air flow travels at much lower speeds, the large volume of air that is accelerated by the fan produces a significant thrust in addition to that created by turbojet core without burning additional fuel. Thus a turbofan is much more fuel efficient than a turbojet.

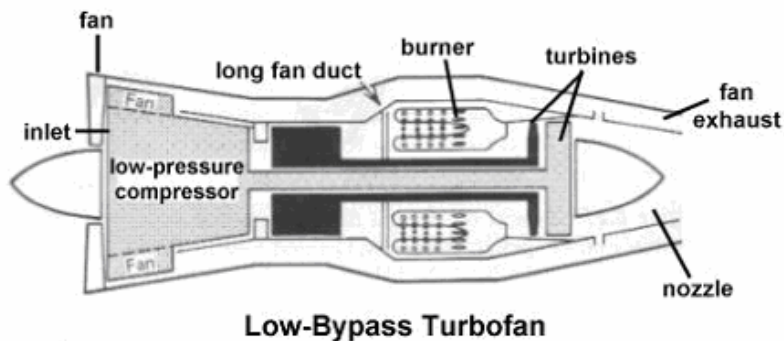


Figure 2.8: A low-bypass turbofan

Turbofans are classified into two different categories: Low-Bypass turbofan and High-Bypass turbofan. The bypass ratio refers to the ratio of incoming air that passes through the fan ducts compared to the incoming air passing through the jet core. A turbofan with a very small diameter fan and amount of air passing through the fan ducts is less is a low-bypass turbofan and it is compact. But a high-bypass turbofan can produce much greater thrust, is more fuel efficient and is much quieter.

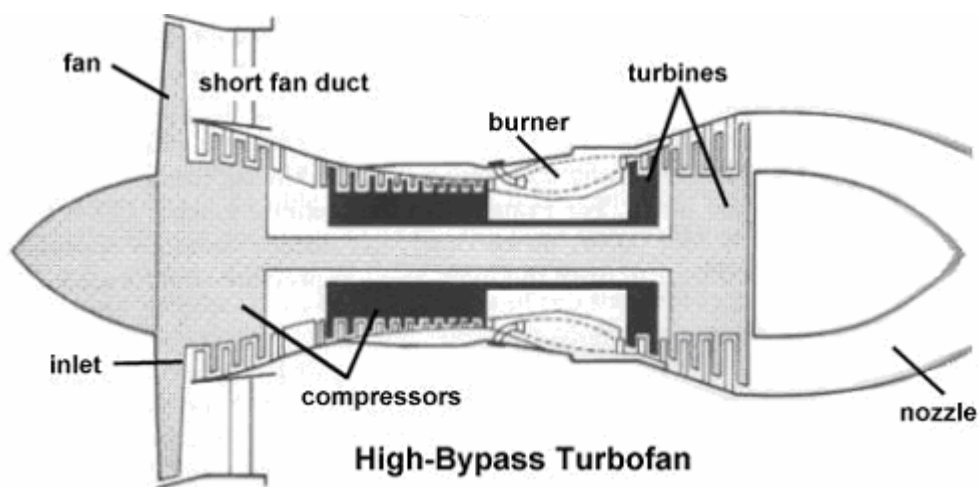


Figure 2.9: A high-bypass turbofan

Most of today's airliners are powered by turbofans. In a turbojet all the air entering the intake passes through the gas generator, which is composed of the compressor, combustion chamber, and turbine. In a turbofan engine only a portion of the incoming air goes into the combustion chamber. The remainder passes through a fan, or low-pressure compressor, and is ejected directly as a "cold" jet or mixed with the gas-generator exhaust to produce a "hot" jet. The objective of this sort of bypass system is to increase thrust without increasing fuel consumption. It achieves this by increasing the total air-mass flow and reducing the velocity within the same total energy supply.

## **2.4 Engine Performance**

Engine performance efficiency depends on many other important components like actuators, sensors and other instruments mounted on each of its components. These components measure some critical parameters of the engine that would reveal the engine performance. Jet engines are precision machines composed of many expensive parts. A thorough understanding of the construction and operation of an engine and its components is vital to good jet engine maintenance. The maintenance in jet engine is divided in to two categories: a) preventive maintenance and b) corrective maintenance. The routine inspection of the various engine components, assemblies, and systems come under the preventive maintenance category. Corrective maintenance is the one in which the malfunctions and damaged parts are fixed or replaced as they occur.

Even though the operation manual of the engine described by the manufacturer gives the details of how often to perform maintenance schedules, the engine performance over a period of time is considered for scheduling the above maintenances to enhance the

engine's life. It is a well known fact that any machine's performance degrades for the period of time it works. Turbofan engine's performance also deteriorates over time as it is also some kind of machine. The engine's performance deterioration plays an important role in the engine maintenance schedule. Engine performance deterioration also reduces the fuel economy of the engine [7]. Just like the life of a living being is affected by its health, the life of a turbofan is also affected by its performance over a period. Engine's performance is also referred by other terms like health or condition. There are many parameters of the engine which would fully describe the engine's health. It would be difficult to consider all the parameters that would add to the engine's health. Hence, only a certain number of parameters are considered which would have a major effect on the engine's health. These parameters are called *health parameters*. Different engines have different sets of health parameters.

Health parameters of the engine are measured using engine condition monitoring devices. However, some of the parameters cannot be measured because of difficulties like the problem of mounting the device in a particular position, unavailability of devices for measuring certain parameters, getting inaccurate measurements from the devices or due to the complex design of the turbofan engine. Monitoring and evaluating these health parameters by some means would help in good maintenance and also increase the life of engine. Engine health evaluation can also be very helpful in some predictive control techniques.

## 2.5 Turbofan Engine Model – MAPSS

A computer aided control design and simulation package which allows graphical representation of dynamical systems in a block diagram form of a generic turbofan engine is being implemented in this research. The Modular Aero-Propulsion System Simulation (MAPSS) is a nonlinear, non-real-time system [9]. This turbofan engine model as shown in Figure 2.10 is similar to the models used in various areas of intelligent engine control research such as model-based control and nonlinear performance seeking control. MAPSS was developed in Simulink, which is quiet simple and user friendly. The current technology with advanced modeling software, such as MATLAB [10] and Simulink [11], to develop an engine model represented in a graphical simulation environment, has the capability to become an extremely powerful tool in control and diagnostic system development[9].

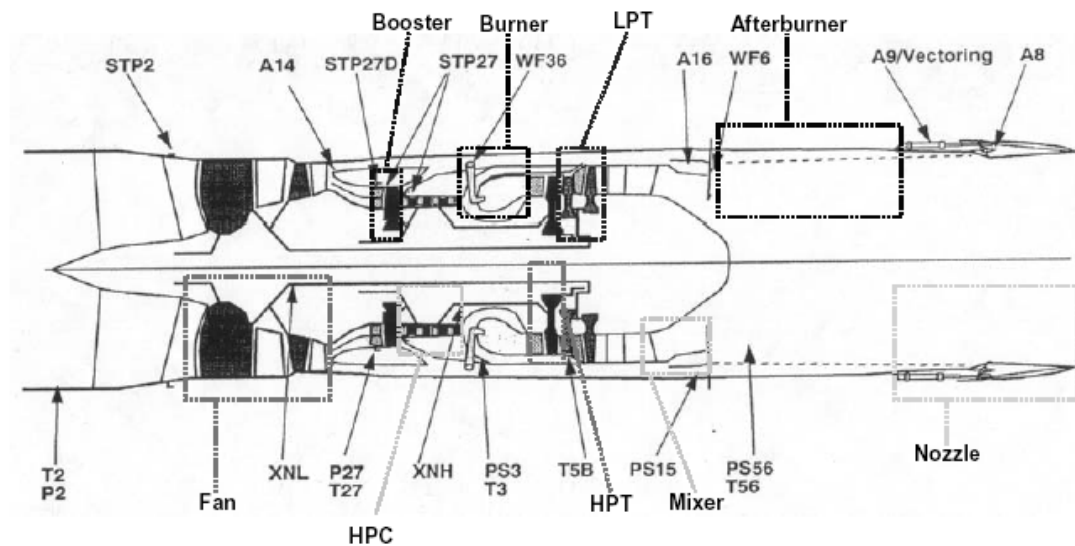


Figure 2.10: Schematic of turbofan engine model

MAPSS is a non real time, nonlinear system composed of the “Controller and Actuator Dynamics” (CAD) and “Component Level Model” (CLM) modules [9]. The computer engine models used in this type of intelligent engine control research are called component level models. The CAD and CLM modules of MAPSS are managed by a GUI interfaced to them as shown in Figure 2.11. The CLM module in the MAPSS environment represents the core engine model with all its components.

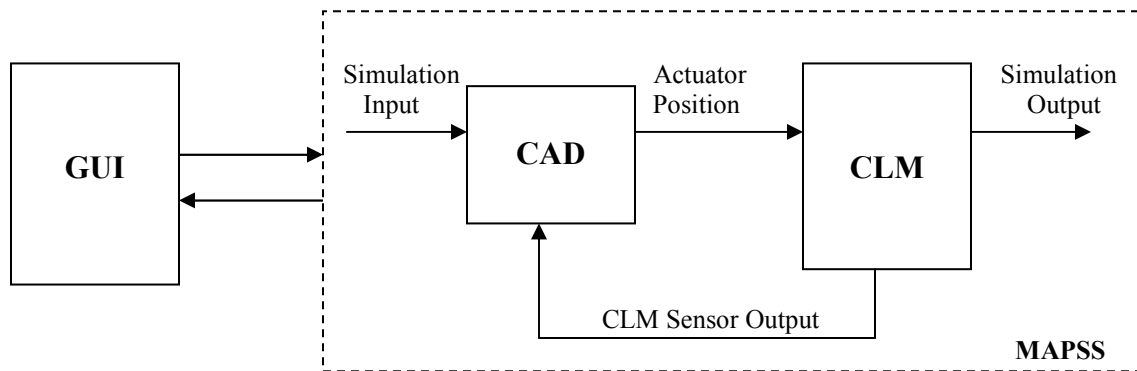


Figure 2.11: MAPSS block diagram (Module Interaction)

### **Component Level Model (CLM)**

The fundamental engine model is represented by CLM in MAPSS. The CLM has all of the principal components of a turbofan engine like fan, booster, high pressure compressor (HPC), burner, high pressure turbine (HPT), low pressure turbine (LPT), mixer, afterburner (AB), and nozzle. The engine components were modeled by mathematical equations before they can be implemented as they have different behaviors like chemical, mechanical, electrical and thermo dynamical which are difficult to implement directly in simulations. There are many subsystems inside each of the component block. These subsystems contain algebraic equations and maps that characterize the behavior of that

particular component when simulated. There is no inlet in the CLM typically, because it is not considered to be a part of the engine model. The block diagram of the CLM (engine) as shown in Figure 2.12, in MAPSS resembles the engine model shown in Figure 2.9 except for the inlet. The overall engine block requires certain iteration steps to ensure a balance of mass flows and energy. The typical parameters like pressure, temperature and flow of certain blocks re implemented as some kind of signals.

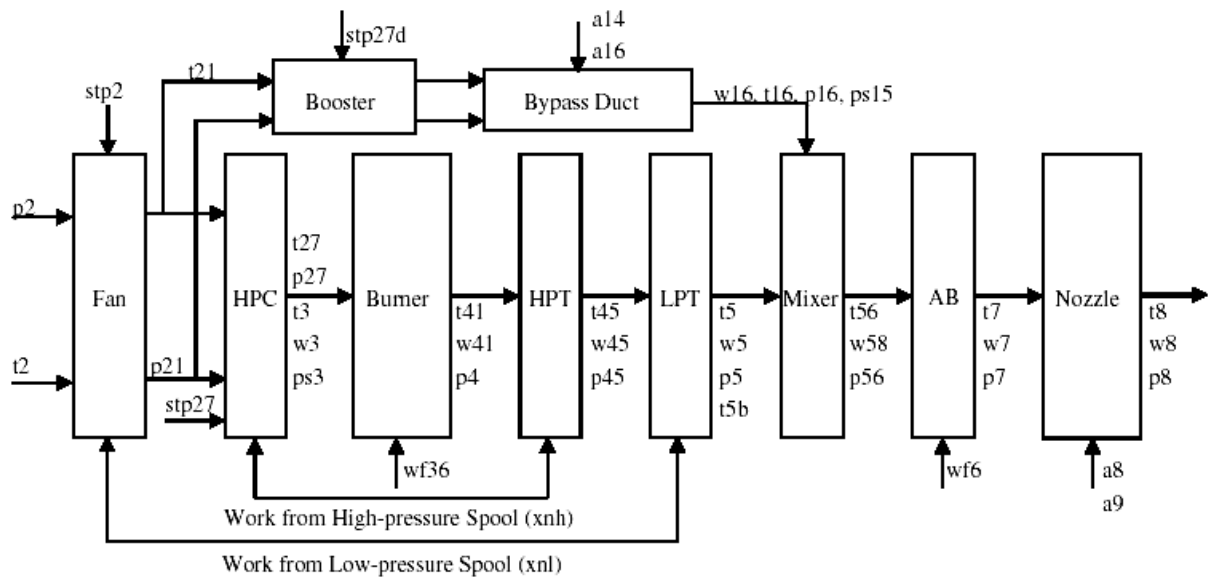


Figure 2.12: CLM with engine components

### CAD Module

The CAD module has the engine's controller and actuator dynamics. The controller in the CAD module emulates the digital controller for the turbofan engine. This module incorporates all the components or instruments that will be used in the control structure of the engine. The actuator dynamic's sub modules are designed based on the mathematical equations of the real time actuators used in the turbofan engines. They simulate instruments like torque motors and servomechanisms for engine components like fan,

HPC, booster etc. The actuators and CLM module use the same sampling rate to obtain mass-energy balance inside the components (engine).

## 2.6 Engine's State Space Model

Mathematical equations, typically differential or difference equations, are used to describe the behavior of processes and to predict their response to certain inputs [12]. These sets of equations put into a common framework called a *state space model* of the system. This process of describing the systems by state space models is termed *modeling*. These models have many useful features like giving an intuitive understanding of the behavior of many dynamical systems and can also be solved efficiently since they are mathematical equations.

$$\begin{aligned}\dot{x} &= f(x, u) \\ y &= h(x, u)\end{aligned}\tag{1}$$

Any state space model can be represented with combinations of three parameters of the system as shown in Equation 1. These three parameters, namely states, inputs and outputs, describe the system's behavior as a whole. The inputs and outputs constitute the external variables, whereas states form the internal variables of the system. The state of a dynamic system is defined as the smallest set of variables such that the knowledge of these variables together with the knowledge of inputs determines the behavior of system at any time.

The turbofan engine modeled as MAPSS has a set of states, control inputs and outputs that describe the engine's behavior. This aircraft gas turbine engine system has 3 states, 8 health parameters and 11 sensors.

A simple small signal model of MAPSS can be written as

$$\begin{bmatrix} \delta x_{k+1} \\ \delta p_{k+1} \end{bmatrix} = A \begin{bmatrix} \delta x_k \\ \delta p_k \end{bmatrix} + B \delta u_k + v_k \quad (2)$$

$$\delta y_k = C \begin{bmatrix} \delta x_k \\ \delta p_k \end{bmatrix} + D \delta u_k + e_k \quad (3)$$

In the above equations (2 and 3),  $\delta x, \delta p$  and  $\delta y$  are the variations in the original 3-element state variable vector, variations in the 8-element health parameter vector, and the variations in the 10-element output vector, respectively,  $v$  is the process noise vector and  $e$  is the measurement noise vector.  $A, B, C,$  and  $D$  are matrices of appropriate dimensions.

The covariances of  $v$  and  $e$  are given as

$$\begin{aligned} E[v_k v_k^T] &= Q \\ E[e_k e_k^T] &= R \end{aligned} \quad (4)$$

Where  $Q$  is the process noise covariance and  $R$  is the measurement noise covariance. The states and control inputs of the MAPSS model are given below.

### **STATES**

- 1) High pressure rotor speed (xnh)
- 2) Low pressure rotor speed (xnl)
- 3) Heat soak temperature (tmpe)

### **CONTROL INPUTS**

- 1) Main burner fuel flow
- 2) Variable nozzle area
- 3) Rear BP door variable area

The eleven sensor outputs of the model are as follows.

- 1) LPT exit pressure
- 2) LPT exit temperature
- 3) Percent low pressure spool rotor speed
- 4) HPC inlet temperature
- 5) HPC exit temperature
- 6) Bypass duct pressure
- 7) Fan exit pressure
- 8) Booster inlet pressure
- 9) HPC exit pressure
- 10) Core rotor speed
- 11) LPT blade temperature

The parameters which have major effect on the health of the turbofan engine are called the health parameters. The importance of these parameters was already discussed in Section 2.4. Estimating these health parameters over a period of time is the objective of this research. The ten health parameters of the MAPSS model are

- 1) Fan airflow
- 2) Fan efficiency
- 3) Booster tip airflow
- 4) Booster tip efficiency
- 5) Booster hub airflow

- 6) Booster hub efficiency
- 7) High pressure turbine airflow
- 8) High pressure turbine efficiency
- 9) Low pressure turbine airflow
- 10) Low pressure turbine efficiency

## **CHAPTER III**

### **GENETIC ALGORITHMS**

Natural optimization methods which have surfaced recently generate new points in the search space by applying operators to current points and statistically move toward more optimal places in the search space. These methods include the genetic algorithm (Holland 1975) [14]. Section 3.1 discusses the theory behind genetic algorithm and continued by discussing basic components of a genetic algorithm in Section 3.2. Section 3.3 discusses the elements of genetic algorithm. The mathematical background behind success of genetic algorithm is discussed in Section 3.4 as schema theorem. In Section 3.5 the concept of exploration and exploitation is discussed using a k-armed bandit problem

#### **3.1 Genetic Algorithms**

Genetic algorithms (GAs) are probably the best known evolutionary algorithms (EAs), receiving remarkable attention all over the world. GAs have been introduced by John Holland in the 1970s [14]. He used a population based algorithm to evolve rules for classifier systems. These algorithms were applied to parameter optimization for the first time by K. De Jong, who laid down the foundations of this application technique [14].

Nowadays, numerous modifications of the original GA, usually referred to as canonical GA, are applied to all fields of global optimization and/or machine learning

A general optimization problem is given in the following form:

$$\text{Find an } x_0 \in X, \text{ such that } f \text{ is maximal in } x_0, \text{ where } f: X \rightarrow R \text{ is an arbitrary real-valued function, i.e., } f(x_0) = \max_{x \in X} f(x). \quad (3.1)$$

However, it may be impossible to obtain a global solution in the strict sense of Equation 3.1. So our interest will be to find an  $x$ , where  $f(x)$  is as high as possible. The search space  $X$  can be seen as a set of competing individuals in the real world, where  $f$  is the function which assigns the value of fitness to each individual.

EAs are inspired by natural process of evolution and make use of same terminology. The peculiarity of EAs is the maintenance of a set of points (population) that are searched in parallel. Each point (individual) is evaluated according to the objective function (fitness function). Two genetic operators contribute to the two basic principles in evolution selection and variation. Selection focuses the search to a better region of the search space by giving individuals with better fitness values a higher probability to be a member of the next generation. On the other hand, variation operators (crossover and mutation) create new points in the search space. In the real world, reproduction and adaptation is carried out on the level of genetic information. Consequently, genetic algorithms do not operate on the values in the search space  $X$ , but on some coded versions of them.

Assume  $S$  to be a set of strings. Let  $X$  be the search space of an optimization problem as above, then a function

$$\begin{aligned}
 c: X &\rightarrow S \\
 x &\rightarrow c(x)
 \end{aligned}
 \tag{3.2}$$

is called a coding function. Conversely, a function

$$\begin{aligned}
 \tilde{c}: S &\rightarrow X \\
 s &\rightarrow \tilde{c}(s)
 \end{aligned}
 \tag{3.3}$$

is called a decoding function. The decoding and encoding of genotype and phenotype respectively is as shown in Figure 3.1.

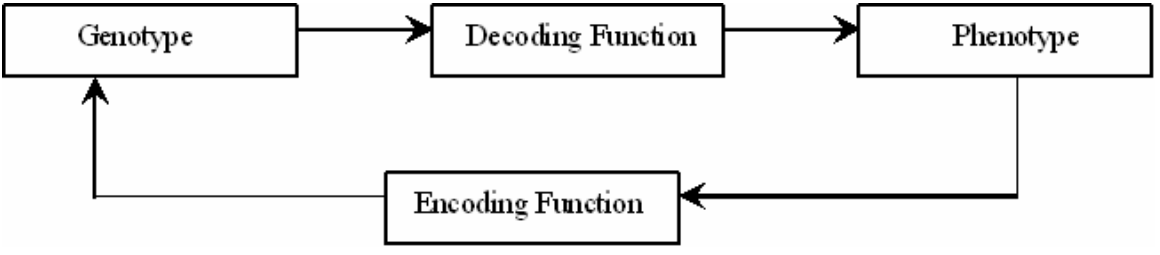


Figure 3.1 Decoding and encoding

A list of different expressions is described in the box below, which are common in genetics, along with their equivalent in the framework of GAs.

List of expressions and equivalents in GA

Natural Evolution	Genetic Algorithm
Genotype	Coded string
Phenotype	Uncoded point
Chromosome	String
Gene	String position
Allele	Value at a certain position
Fitness	Objective function value

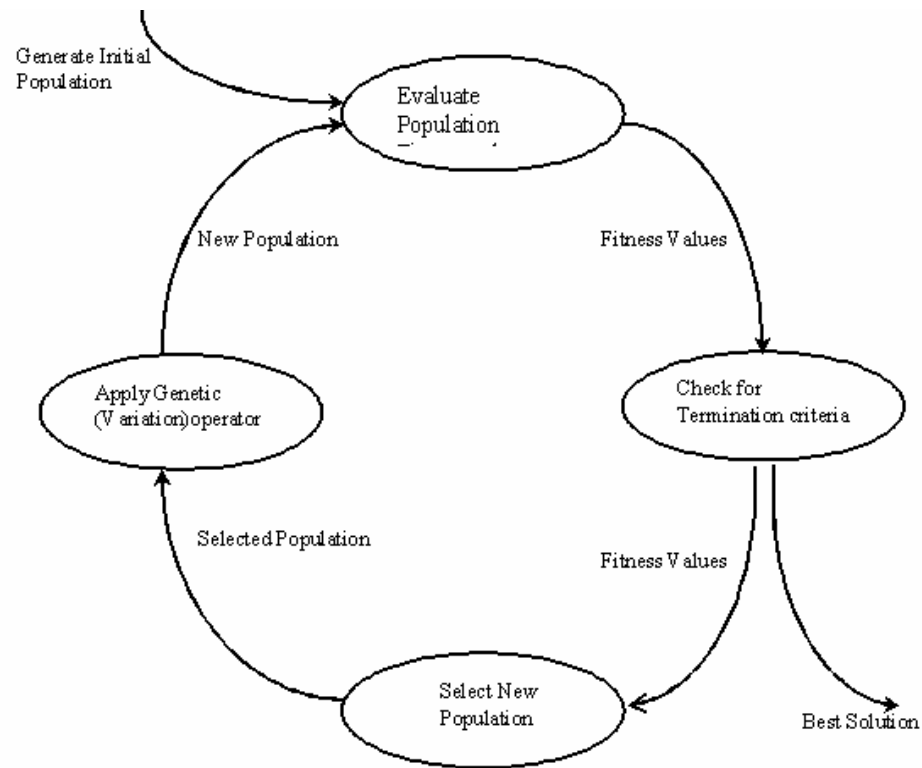


Figure 3.2 Evolutionary search cycle for global optimization

```

t := 0;
Compute initial population  $B_0$ ;
WHILE stopping condition not fulfilled DO
BEGIN
Select individuals for reproduction;
Create offsprings by crossing individuals;
Eventually mutate some individuals;
Compute new generation;
END
  
```

Figure 3.3 A pseudocode of genetic algorithm

### 3.2 Basic Components

It is obvious from the above Figure 3.2 and Figure 3.3, the transition from one generation to the next generation consists of four basic components.

**Selection:** Mechanism for selecting individuals (strings) for reproduction according to their fitness (objective function value).

**Crossover:** Method of merging the genetic information of two individuals. If the coding is chosen properly, two good parents produce good children.

**Mutation:** In real evolution, the genetic material can be changed randomly by erroneous reproduction or other deformations of genes, e.g., by gamma radiation. In genetic algorithms, mutation can be realized as a random deformation of the strings with a certain probability. The positive effect is preservation of genetic diversity and, as an effect, that local maxima can be avoided.

**Sampling:** This is a procedure that computes a new generation from the previous one and its offspring.

So compared with traditional continuous optimization methods, such as Newton or gradient descent methods, we can state the following significant differences:

1. Genetic algorithms (GAs) manipulate coded versions of the problem parameters instead of the parameters themselves, i.e., the search space is  $S$  instead of  $X$  itself.
2. While almost all convenient methods search from a single point, GAs always operate on a whole population of points (strings). This contributes much to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and, vice versa, reduces the risk of becoming trapped in a local stationary point.

3. Normal GAs do not use any auxiliary information about the objective function value such as derivatives. Therefore, they can be applied to any kind of continuous or discrete optimization problem. The only thing to be done is to specify a meaningful decoding function.
4. GAs use probabilistic transition operators while conventional methods for continuous optimization apply deterministic transition operators. More specifically, the way a new generation is computed from the actual one has some random components.

For conventional deterministic optimization methods, such as gradient methods, Newton- or Quasi-Newton methods, etc., it is rather usual to have results which guarantee that the sequence of iterations converges to a local optimum with a certain speed or order. But for probabilistic optimization methods, theorems of this kind cannot be formulated, because the behavior of the algorithm is not deterministic in general. Statements about the convergence of probabilistic optimization methods can only give information about the expected or average behavior. In the case of GAs, there are some circumstances which make it even more difficult to investigate their convergence behavior.

*The GA works on the Darwinian principle of natural selection... whether the specifications be nonlinear, constrained, discrete, multimodal, or even NP-hard, GA is entirely equal to the challenge [15].*

### 3.3 The Elements of a Genetic Algorithm

#### 3.3.1 Initial Population

Selection of initial population size has been approached from several theoretical points of view, although the underlying idea is always of a trade off between efficiency and effectiveness. For some predefined string length, there should be an optimal population size which would allow spanning the search space effectively. But sometimes a large value of population size would impair the efficiency of the method, that no solution could be expected in reasonable amount of computation. The complexity of the problem and its landscape are obviously unknown at the start of the solution search. So it is advised to search for a better population size by increasing the value linearly and checking if there is a large variation in final solution. If a large variation is found, population size should further be increased until a consistent solution is reached.

Now posing a question in slightly different way, what is the minimum value of the population size for a meaningful search to take place? The principle adopted was at the very least, every point in the search space should be reachable from the initial population only by crossover. This requirement was satisfied if there is atleast one instance of the allele at each locus in the whole population of binary strings. On the assumption that the initial population is generated randomly, the probability that no position in the string of length  $l$  has all bits same in random population will be given as

$$\begin{aligned}
 P &= [(1 - (1/2)^{N-1})^l] \\
 &\approx \exp(-l/2^{N-1})
 \end{aligned}
 \tag{3.4}$$

from which we can establish the population size  $N$  as

$$N = \lceil 1 + \log(-1/\ln(P)/\log(2)) \rceil \quad (3.5)$$

For example, if we desire that all possible chromosomes of length 100 are reachable from initial population by crossover, where  $P=99.99\%$ , is to have a population of 18.

### 3.3.2 Termination

A simple neighborhood search method terminates whenever a local optimum is reached. But, a stochastic search method in principle will run forever, which needs termination criteria to be set. Some common approaches as a termination criteria are number of fitness evaluations, computer clock time and rate of convergence with some predefined threshold.

### 3.3.3 Selection

Selection is the component which guides the algorithm to the solution by preferring individuals with high fitness over low-fit ones. It can be a deterministic operation, but in most implementations it has random components. A commonly used implementation is where the probability to choose a certain individual is proportional to its fitness. It can be regarded as a random experiment with

$$P[b_{j,t} \text{ is selected}] = \frac{f(b_{j,t})}{\sum_{k=1}^m f(b_{k,t})} \quad (3.6)$$

This formula only makes sense if all fitness values are positive. If this is not the case, a non-decreasing transformation  $\varphi: R \rightarrow R^+$  must be applied (a shift in the simplest case).

Then the probabilities can be expressed as

$$P[b_{j,t} \text{ is selected}] = \frac{\varphi(b_{j,t})}{\sum_{k=1}^m \varphi(f(b_{k,t}))} \quad (3.7)$$

We can force the property (6) to be satisfied by applying a random experiment which, in some sense, a generalized roulette game. In this roulette game, the slots are not equally wide, i.e. the different outcomes can occur with different probabilities. Consider a graphical representation of roulette wheel selection in Figure 3.4, where the number of alternatives  $m$  is 6. The numbers inside the arcs correspond to the probabilities to which the alternative is selected.

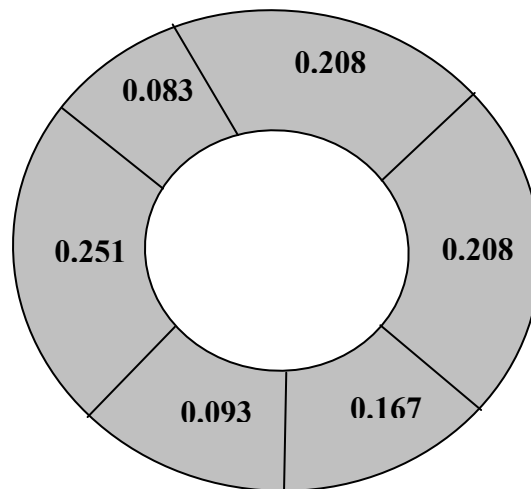


Figure 3.4 Roulette wheel selection

**Roulette Wheel Selection Algorithm:**

```

x := random[0,1];
i := 1;
WHILE i < m & x <  $\sum_{j=1}^i f(b_{j,t}) / \sum_{j=1}^m f(b_{j,t})$  DO
i := i + 1;
select  $b_{i,t}$ ;

```

For obvious reasons this method is often called proportional selection.

### **Stochastic Universal Sampling**

Stochastic universal sampling provides zero bias and minimum spread. The individuals are mapped to contiguous segments of a line, such that each individual's segment is equal in size to its fitness exactly as in roulette-wheel selection. Here, individuals are equally spaced and the number of selections cannot be less than the floor of the expected value of the member and cannot be greater than the ceiling of the expected value of the member.

### **Tournament Selection**

In tournament selection a number  $\tau$  of individuals is chosen randomly from the population and the best individual from this group is selected as parent. This process is repeated as often as individuals must be chosen. These selected parents produce uniform at random offspring. The parameter for tournament selection is the tournament size  $\tau$ . The number  $\tau$  takes values ranging from 2 to  $N_{ind}$  (number of individuals in population).

There are two different types of tournament selection:

#### **(1) Strict Tournament**

In this method  $\tau$  individuals are picked randomly and the most fit among them is selected for crossover. If the most fit individual is one of those  $\tau$  individuals, probability of that individual will be selected for crossover is one. If median fit is one of those  $\tau$  individuals, then probability of it getting selected can be given as  $(1/2)^{\tau-1}$ . Thus, the selection pressure, which is the relative probability that the fittest individual is selected as a parent relative to an individual with average fitness, for a tournament selection is always greater than 2. So, as selection pressure increases the probability of more fit being selected also increase

## **(2) Soft Tournament**

In this method  $\tau$  individuals are picked randomly and the most fit among them is selected for crossover with some probability and the rest can also be picked with the remaining probability. This method decreases the selection pressure thus increasing diversity and better global optimizing performance.

### **3.3.4 Crossover**

In sexual reproduction, as it appears in the real world, the genetic material of the two parents is mixed when the gametes of the parents merge. Chromosomes are randomly split and merged, with the consequence that some genes of the a child come from one parent while the others come from the other parent. This mechanism is called crossover. It is a very powerful tool for introducing new genetic material and maintaining genetic diversity, but with the outstanding property that good parents also produce well-performing children or even better ones. Several investigations have come to the conclusion that crossover is the reason why sexually reproducing species adapted faster than asexually reproducing ones.

Basically, crossover is the exchange of genes between the chromosomes of two parents. In the simplest case, it can be realized as cutting two strings at a randomly chosen position and swapping the two tails. This process, which we will call one-point crossover, is visualized as shown in Figure 3.5.

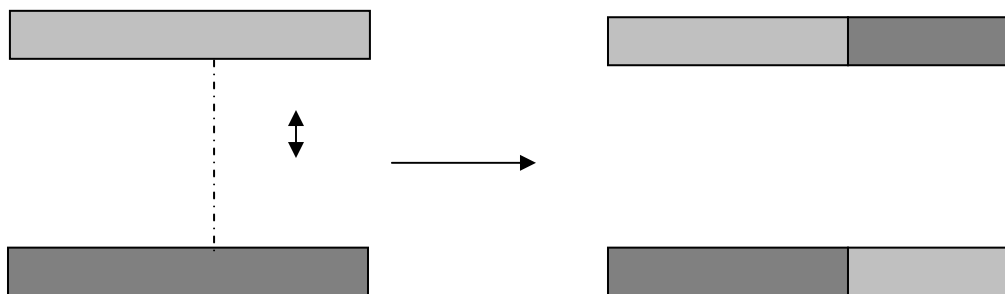


Figure 3.5 One-point crossover of binary strings

One-point crossover is a simple and often used method for GAs which operates on binary strings. For other problems or different codings, other crossover methods can be useful or even necessary.

**N-point Crossover:** Instead of one-point,  $N$  breaking points are chosen randomly. Every second section is swapped. Among this class, two-point crossover is most used crossover method.

**Segmented Crossover:** A switch rate of  $r$  between  $[0,1]$  is used to check if the bits in both parents at each position need to be swapped.

**Uniform Crossover:** A switch rate of 0.5 is used to check if the bits in both parents at each position need to be swapped.

**Shuffle Crossover:** First a randomly chosen permutation is applied to the two parents, then  $N$ -point crossover is applied to the shuffled parents, finally, the shuffled children are transformed back with the inverse permutation

### 3.3.5 Mutation

A random deformation of the genetic information of an individual by means of radioactive radiation or other environmental influences is termed as mutation. In real

reproduction, the probability that a certain gene is mutated is almost equal for all genes. A simple mutation technique for a given binary string  $s$ , where  $p_M$  the probability that a single gene is modified is, is given as an algorithm:

**Algorithm**

```
FOR  $i := 1$  TO  $n$  DO
  IF  $\text{Random}[0,1] < p_M$  THEN
    invert( $s[i]$ );
```

A low mutation probability value prevents the genetic algorithm from behaving chaotically like a random search. Even in case of mutation there are different techniques which depend on the coding and the problem itself. Some of them are inversion of single bits, bitwise inversion and random selection with some mutation probability. Another function of mutation is preserving population diversity and enables the process to escape from sub-optimal regions of the solution spaces [15]. Implementation of adaptive mutation is studied by different authors. Fogarty [16] experimented with different mutation rates at different loci. Reeves [5] varied the mutation probability according to the diversity in the population which is measured in terms of the coefficient of variation of fitness.

### 3.3.6 New Population

Holland's original genetic algorithm assumed a generational replacement strategy by applying selection, recombination, and mutation to a population of chromosomes until a new set of individuals had been generated, which will be the new population for next generation. But there is a risk of throwing away good solution thus preventing it from

taking part in reproduction for future generation. To avoid this De Jong introduces the concept of elitism and population overlaps [15]. Elitism strategy ensures the survival of the best individual so far by preserving it and replacing only the remaining members of the population with new chromosomes. Overlapping populations take this a stage further by replacing only a fraction of the population at each generation. Some implementations have assumed that parents must be replaced by their children. But he introduces a strong selection pressure on the search, which is prevented by large population sizes and high mutation rates and thus preventing loss in population diversity.

### **3.3.7 Diversity Maintenance**

An important aspect to be considered while implementing a genetic algorithm is maintaining population diversity. Niching and crowding of the population is to taken into consideration when dealing with diversity maintenance. The term niche has the roots in biological analogy, which means a set of conditions of which a specific group of phenotypes is particularly well adapted. In a GA, a niche is treated as a subset of chromosomes that are in some sense similar. The idea here is that a newly generated chromosome should replace one in its own niche, rather than potentially any one in the population. De Jong developed a crowding factor [18], which was an integer defining the size of a randomly chosen subset of the existing population, with which the newly generated chromosome was compared. The chromosome which it appears to be closest is the one that will be deleted. Holland says, “The very essence of good GA design is retention of diversity” [19]. The effect of selection is to reduce diversity and some

methods can reduce diversity very rapidly. This can be mitigated by having larger population or by having greater mutation rates, or by niching or crowding.

### 3.4 The Schema Theorem

A member or chromosome in the population is similar to other chromosomes in the population. This similarity can be generalized over a group of chromosomes in the population and is termed “schema.” A schema is a similarity template describing a subset of chromosomes with similarities at certain chromosome positions. The concept of schema will be discussed for an alphabet of cardinality  $k$ . An extra symbol is appended to this alphabet which is termed as a “don’t care” (\*) symbol.

An alphabet with cardinality  $k$  and chromosome of length  $l$  has  $k^l$  different chromosomes, but the number of schemata would be  $(k+1)^l$ . This shows that the problem is being made a bit complicated by increasing the search space. Thus each chromosome will belong to  $k^l$  schemata. So a population of  $n$ ,  $l$ -bit chromosomes belong to between  $k^l$  and  $n*k^l$  schemata. Let us make some fundamental definitions concerning schemata before getting into the math of schema theory.

A string  $S = (s_1, \dots, s_n)$  over the alphabet of cardinality  $k$  fulfills schema  $H = (h_1, \dots, h_n)$  if and only if it matches  $H$  in all non-wildcard positions:

$$\forall i \in \{j \mid h_j \neq *\} : s_i = h_i \quad (3.8)$$

#### Order of Schema

The number of specification of the a schema is called order and denoted as

$$O(H) = \left| \{i \in \{1 \dots n\} \mid h_i \neq *\} \right| \quad (3.9)$$

### Defining Length

The defining length of a schema  $H$  is the distance between the first and last specification

$$\delta(H) = \max \{i \mid h_i \neq *\} - \min \{i \mid h_i \neq *\} \quad (3.10)$$

### Instance

An instance of the schema  $H$  is a chromosome which belongs to  $H$ . So each schema has  $k^{l-O(H)}$  instances.

To provide an intrinsic behavior of genetic algorithms we formulate and prove the fundamental result, called the Schema Theorem.

Assume we have a genetic algorithm discussed in Figure 3.3 with proportional fitness selection and an arbitrary but fixed fitness function  $f$ . With the following notations in place:

1. The number of individuals which fulfill  $H$  at time step  $t$  are denoted as

$$r_{H,t} = |B_t \cap H| \quad (3.11)$$

In a given population  $B_t$ , the number of individuals that have schema  $H$  are  $r_{H,t}$ .

2. The expression  $\bar{f}(t)$  refers to the observed average fitness at time  $t$ .

$$\bar{f}(t) = \frac{1}{m} \sum_{i=1}^m f(b_{i,t}) \quad (3.12)$$

3. The  $\bar{f}(H,t)$  term stands for the observed average fitness of schema  $H$  in time step  $t$

$$\bar{f}(H,t) = \frac{1}{r_{H,t}} \sum_{i \in \{j \mid b_{j,t} \in H\}} f(b_{i,t}) \quad (3.13)$$

**Theorem (Schema Theorem – Holland 1975):** Assuming we consider a simple genetic algorithm as in Figure 3.2, the following inequality holds for every schema  $H$ :

$$E[r_{H,t+1}] \geq r_{H,t} \cdot \frac{\bar{f}(H,t)}{f(t)} \cdot \left(1 - p_c \cdot \frac{\delta(H)}{l-1}\right) \cdot (1 - p_M)^{O(H)} \quad (3.14)$$

**Proof.** The probability that we select an individual fulfilling  $H$  is

$$\frac{\sum_{i \in \{j | b_{j,t} \in H\}} f(b_{i,t})}{\sum_{i=1}^m f(b_{i,t})} \quad (3.15)$$

This probability doesn't change throughout the execution of the selection loop. Moreover, each of the  $m$  individuals selected is completely independent from the others. Hence, the number of selected individuals, which fulfill  $H$ , is binomially distributed with sample amount  $m$  and the probability in Equation 3.15. We obtain, therefore, that the expected number of selected individuals fulfilling  $H$  is

$$\begin{aligned} m \cdot \frac{\sum_{i \in \{j | b_{j,t} \in H\}} f(b_{i,t})}{\sum_{i=1}^m f(b_{i,t})} &= m \cdot \frac{r_{H,t}}{r_{H,t}} \cdot \frac{\sum_{i \in \{j | b_{j,t} \in H\}} f(b_{i,t})}{\sum_{i=1}^m f(b_{i,t})} \\ &= r_{H,t} \cdot \frac{\sum_{i \in \{j | b_{j,t} \in H\}} f(b_{i,t})}{\sum_{i=1}^m f(b_{i,t})} = r_{H,t} \cdot \frac{\bar{f}(H,t)}{f(t)} \end{aligned} \quad (3.16)$$

If two individuals are crossed, which both fulfill  $H$ , the two offsprings again fulfill  $H$ . The number of chromosomes fulfilling  $H$  can only decrease if one string, which fulfills  $H$ , is crossed with a chromosome which does not fulfill  $H$ , but, obviously, only in the case that the cross site is chosen somewhere in between the specifications of  $H$ .

The probability that the cross site is chosen within the defining length of  $H$  is

$$\frac{\delta(H)}{l-1} \quad (3.17)$$

Hence the survival probability  $p_s$  of  $H$ , i.e. the probability that a chromosome fulfilling  $H$  produces an offspring also fulfilling  $H$ , can be estimated as follows (crossover is done with a probability of  $p_c$ ):

$$p_s \geq 1 - p_c \cdot \frac{\delta(H)}{l-1} \quad (3.18)$$

Selection and crossover are carried out independently, so we may compute the expected number of chromosomes fulfilling  $H$  after crossover simply as

$$\frac{\bar{f}(H,t)}{\bar{f}(t)} \cdot r_{H,t} \cdot p_s \geq \frac{\bar{f}(H,t)}{\bar{f}(t)} \cdot r_{H,t} \cdot \left(1 - p_c \cdot \frac{\delta(H)}{l-1}\right) \quad (3.19)$$

After crossover, the number of chromosomes fulfilling  $H$  can decrease only if a chromosome fulfilling  $H$  is altered by mutation at a specification of  $H$ . The probability that all specifications of  $H$  remain untouched by mutation is

$$(1 - p_M)^{O(H)} \quad (3.20)$$

So the probability that the schema  $H$  survives mutation can be approximated using a Taylor series as

$$(1 - O(H)p_M) \quad (3.21)$$

Applying the same argument like above, the expected number of chromosomes fulfilling  $H$  after crossover and mutation is given as

$$E[r_{H,t+1}] = \frac{\bar{f}(H,t)}{\bar{f}(t)} \cdot r_{H,t} \cdot p_s \geq \frac{\bar{f}(H,t)}{\bar{f}(t)} \cdot r_{H,t} \cdot \left(1 - p_c \cdot \frac{\delta(H)}{l-1}\right) \cdot (1 - O(H)p_M) \quad (3.22)$$

Hence, short low order schema with above average fitness receive exponentially increasing number of chromosomes

**QED**

The arguments in the proof of schema theorem can be applied analogously to many other crossover and mutation operations.

For a genetic algorithm with roulette wheel selection, the inequality holds

$$E[r_{H,t+1}] \geq \frac{\bar{f}(H,t)}{\bar{f}(t)} \cdot r_{H,t} \cdot P_C(H) \cdot P_M(H) \quad (3.23)$$

for any schema  $H$ , where  $P_C(H)$  is a constant only depending on the schema  $H$  and the crossover method and  $P_M(H)$  is a constant which solely depends on  $H$  and the involved mutation operator. Considering different crossover and mutation methods we define different inequalities.

If one-point crossover is implemented

$$P_C(H) = 1 - p_c \cdot \frac{\delta(H)}{l-1} \quad (3.24)$$

If uniform crossover is implemented

$$P_C(H) = 1 - p_c \left(1 - (1/2)^{O(H)}\right) \quad (3.25)$$

If any other crossover method is implemented

$$P_C(H) = 1 - p_c \quad (3.26)$$

If the mutation operator is bitwise mutation

$$P_M(H) = (1 - p_m)^{O(H)} \quad (3.27)$$

If the mutation operator is inversion of a single bit

$$P_M(H) = 1 - p_M \cdot \frac{O(H)}{n} \quad (3.28)$$

If the mutation operator is bitwise inversion

$$P_M(H) = 1 - p_M \quad (3.29)$$

If the mutation operator is random selection

$$P_M(H) = 1 - p_M \cdot \frac{|H|}{n} \quad (3.30)$$

We observe that short, low order schema with above average fitness receive exponentially increasing number of chromosomes which will be called as building blocks.

This can be explained if we assume the quotient

$$\frac{\bar{f}(H, t)}{\bar{f}(t)} \quad (3.31)$$

is approximately stationary, i.e., independent of time and the actual generations, we immediately see that the number of chromosomes, which belong to above-average schemata with short defining lengths, grows exponentially.

### 3.5 Exploration and Exploitation

The building blocks receiving exponential increasing trails in future generations can be analyzed using a well-analyzed statistical decision theory, the  $k$ -armed bandit problem. Now consider the two-armed bandit problem, a gambling machine with two slots for coins and two arms [13]. The gambler can deposit the coin either into the left or the right slot. After pulling the corresponding arm, either a reward is paid or the coin is lost. For

mathematical simplicity, we just work with outcomes, i.e. the difference between the reward (which can be zero) and the value of the coin.

Let us assume that the left arm produces an outcome with mean value  $\mu_1$  and a variance  $\sigma_1^2$  while the right arm produces an outcome with mean value  $\mu_2$  and variance  $\sigma_2^2$ .

Without loss of generality, although the gambler does not know this, assume that

$$\mu_1 \geq \mu_2.$$

There is still a dilemma to which arm should be played. Since the gambler is unaware of the arm associated with higher outcome beforehand, we must make a sequence of decisions which arm to play; we have to collect, at the same time information about which is better arm. This trade-off between exploration of knowledge and its exploitation is the key issue in this problem and, as turns out later, in genetic algorithms, too.

Suppose we have  $N$  coins, if we first allocate an equal number  $n$  (where  $2n \leq N$ ) of trials to both arms, we can allocate the remaining  $N-2n$  trials to the observed better arm. Let  $q_n$  be the probability that estimated better arm is actually worse after  $2n$  trials. The expected loss is given as

$$L(N, n) = (\mu_1 - \mu_2) \cdot ((N - n)q_n + n(1 - q_n)) \quad (3.32)$$

In case that we observe that the worse arm is the best, which happens with probability  $q_n$ , the total number of trials allocated to the right arm is  $(N-n)$ . The loss is  $(\mu_1 - \mu_2) \cdot (N - n)$

In the reverse case, if we actually observe that the best arm is best, which happens with probability  $(1 - q_n)$ , the loss is only what we lose because we played the worse arm  $n$  times, i.e.  $(\mu_1 - \mu_2).n$ . Taking the central limit theorem into account, we can approximate  $q_n$  with a normal distribution.

$$q_n = \frac{1}{2\pi} \frac{e^{-x^2/2}}{X} \quad \text{where} \quad X = \frac{(\mu_1 - \mu_2)}{\sqrt{\sigma_1^2 - \sigma_2^2}} \sqrt{n} \quad (3.33)$$

According Holland [13], the optimal strategy is given by the following equation:

Optimal  $n$ ,

$$n^* = b^2 \ln \left| \frac{N^2}{8\pi b^4 \ln N^2} \right| \quad \text{where} \quad b = \frac{\sigma_1}{\mu_1 - \mu_2} \quad \text{for exploration phase} \quad (3.34)$$

Rest of trials  $(N - n^*)$  in exploitation phase is given by:

$$N - n^* = c \sqrt{\ln N^2} \exp \left( \frac{n^*}{2b^2} \right) - n^* \quad \text{where} \quad c = \sqrt{8\pi b^4} \quad (3.35)$$

So  $N - n^*$ , is the number of trials to be observed for better arm. As  $n^*$  increases,  $N - n^*$  increases exponentially thus giving exponentially increasing number of trails to the better arm, which is similar to schema theorem. Another application which depicts the ideal strategy of trial allocation is the three operator genetic algorithm. The schema theorem guarantees giving at least an exponentially increasing number of trials to the observed best building blocks. In this way genetic algorithm is a realizable yet near optimal procedure for searching among alternative solutions because it allocates exponentially increasing numbers to the observed best schemata just as we give exponentially increasing trials to observed best arm in the  $k$ -armed bandit problem [13].

## **CHAPTER IV**

### **SENSOR SELECTION**

For a physical dynamical system with time invariant extended parameters appended to the state vector, the minimum number of outputs required for observability is equal to the number of extended parameters [1]. A tradeoff should be made between the number of sensors and the financial cost. We could simply use all sensors to obtain the best possible health estimation. But if we can save a lot of money or effort at a very small decrease in estimation accuracy, then we may want to use fewer sensors. The goal is to find a measure of the quality of the estimation as a function of the sensor set used. This is accomplished by applying combinations of sensors to minimize a cost function generated using the covariance of each state estimate. Section 4.1 discusses about the turbofan system model. In Section 4.2 the probabilistic approach to sensor selection is discussed continued with application of genetic algorithm to sensor selection in Section 4.3. In the final Section 4.5 Eigenvector approach to sensor selection is being discussed.

## 4.1 The System Model

The aircraft gas turbine engine system model MAPSS used in this research has 3 states, 8 health parameters and 11 sensors. The model time-invariant equations can be summarized as follows:

$$\begin{aligned}\dot{x} &= f(x, u, p) + w(t) \\ y &= g(x, u, p) + e(t)\end{aligned}\tag{4.1}$$

Here  $x$  is a system state vector with 3 states,  $u$  is the input vector,  $p$  is the 8-element vector of health parameters, and  $y$  is the 10-element vector of measurements. The noise term  $w(t)$  represents inaccuracy in the model, and  $e(t)$  represents measurement noise. A linearized model of MAPSS from the nominal operating point with some small excursions is in equation (4.2)

$$\begin{aligned}\delta\dot{x} &= A_1\delta x + B\delta u + A_2\delta p + v_1(t) \\ \delta y &= C_1\delta x + C_2\delta p + D\delta u + e(t)\end{aligned}\tag{4.2}$$

An augmented system model is constructed in such way to estimate health parameters along with states of the system. The discrete time augmented system of MAPSS is as shown below in equation 4.3.

$$\begin{aligned}\begin{bmatrix} \delta x_{k+1} \\ \delta p_{k+1} \end{bmatrix} &= \begin{bmatrix} A_1 & A_2 \\ 0 & I \end{bmatrix} \begin{bmatrix} \delta x_k \\ \delta p_k \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \delta u_k + \begin{bmatrix} v_{1k} \\ v_{2k} \end{bmatrix} \\ \delta y_k &= [C_1 \quad C_2] \begin{bmatrix} \delta x_k \\ \delta p_k \end{bmatrix} + D\delta u_k + e_k\end{aligned}\tag{4.3}$$

A simple small signal model of MAPSS can be written as

$$\begin{aligned}\begin{bmatrix} \delta x_{k+1} \\ \delta p_{k+1} \end{bmatrix} &= A \begin{bmatrix} \delta x_k \\ \delta p_k \end{bmatrix} + B\delta u_k + \begin{bmatrix} v_{1k} \\ v_{2k} \end{bmatrix} \\ \delta y_k &= C \begin{bmatrix} \delta x_k \\ \delta p_k \end{bmatrix} + D\delta u_k + e_k\end{aligned}\tag{4.4}$$

In the above equation (4.4),  $\delta x, \delta p$  and  $\delta y$  are the variations in the original 3-element state variable vector, variations in the 8-element health parameter vector, and the variations in the 10-element output vector, respectively. When considered from the point of view of the linearized engine model [1], the health parameters are inputs, while from the filter's point of view they are state variables. Here  $v_{2k}$  is the noise term, which allows the health parameter estimates to change, since the state equations lock them in steady state ( $\delta p_{k+1} = \delta p_k$ ) [1], and  $e$  is the measurement noise vector. The covariances of  $v$  and  $e$  are given as

$$\begin{aligned} E[v_k v_k^T] &= Q \\ E[e_k e_k^T] &= R \end{aligned} \tag{4.5}$$

## 4.2 Probabilistic Approach to Sensor Selection

The system model built assumes that the health information is contained in the state variables. The goal is to find a measure of the quality of estimation as a function of the sensor set utilized. A trade off should be made between sensors, cost, weight, accuracy and reliability. So, the given system is modified by additional sensors to allow better estimates. The concept of using multiple sensors for a single measurement produces smaller variance than that produced using a single sensor alone [1]. In order to estimate the health parameters, the original 3 states are augmented with the 8 health parameters, so the  $A$  matrix is  $11 \times 11$ . The measurement matrix with 11 rows (corresponding to 11 sensors) can be duplicated with the same set of 11 sensors in case each sensor is used twice. This results in a measurement matrix  $C$  with 11 columns (one for each state) and

up to 22 rows (one for each sensor). A change in  $C$  also effects the measurement noise covariance  $R$ , which is constructed to be consistent with  $C$ . These system matrices are used to produce an error covariance  $P$ , which will be designated as  $P_{ref}$ , if 22 sensors are used.

Consider a linear stochastic system represented by

$$\begin{aligned} x_{k+1} &= Ax_k + B_u u_k + B_w w_k \\ y_k &= Cx_k + v_k \end{aligned} \quad (4.6)$$

Here  $x$  is the system state vector,  $y$  is the measurement vector,  $u$  is the input vector,  $w$  is the process noise vector and  $v$  is the measurement noise vector.  $A$ ,  $B_u$ ,  $B_w$  and  $C$  are matrices of appropriate dimensions.  $w$  and  $v$  in this case are assumed to be mutually independent and zero mean white noise. The covariances of  $w$  and  $v$  are given as

$$\begin{aligned} E[w_k w_k^T] &= Q \\ E[v_k v_k^T] &= R \end{aligned} \quad (4.7)$$

The state estimate equations before and after the measurements are processed are given as

$$\begin{aligned} \hat{x}_{k+1}^- &= A\hat{x}_k^+ + B_u u_k \\ \hat{x}_{k+1}^+ &= \hat{x}_{k+1}^- + K_k (y_{k+1} - C\hat{x}_{k+1}^-) \end{aligned} \quad (4.8)$$

Where  $K_k$  is the Kalman filter gain.

The estimation error is defined as follows:

$$e_{k+1} \equiv x_{k+1} - \hat{x}_{k+1}^-$$

From Equations (1) and (3) the estimation error satisfies the equation

$$e_{k+1} = (A - AK_k C)e_k + B_w w_k - AK_k v_k \quad (4.9)$$

Using the noise characteristics in Equation (4) the steady state error covariance  $P$  becomes the solution to the following equation:

$$P = (A - AKC)P(A - AKC)^T + B_wQB_w^T + (AK)R(AK)^T \quad (4.10)$$

where  $P$  is defined as

$$P = E[ee^T]$$

where  $K$  is the Kalman gain for the given sensor set. This is the error covariance when Kalman filtering is used for state estimation. The error covariances are divided into the first 3, representing the original state estimation error variances, and last eight, representing the health parameter estimation variances. As we will be interested only in the health parameter error covariance we introduce a weighting function, which only considers the 4th through 11th elements of the augmented state. In order to compare the quality of estimation the following metric is defined.

$$J = \sum_{i=1}^n w_i \left( \sqrt{\frac{P_{i,i}(k|k)}{P_{i,i}^{ref}(k|k)}}} \right) + \frac{\text{financial cost}}{\text{ref financial cost}}$$

Where  $w_i = \begin{cases} 0 & i=1,\dots,3 \\ 1 & i=4,\dots,11 \end{cases}$  ,

(4.11)

The sensor selection of 22 sensors can have 177146 combinations of sensors to be selected from. Initially a random search algorithm generates sensor sets and evaluates the cost function as shown in equation (4.11). This random search algorithm is executed several times capturing the data for each execution. The probability of each sensor being in the top  $x\%$  cost of sensor sets is evaluated, where  $x$  is a user-specified threshold. Now, based on the probability of each sensor, a probabilistic algorithm, which generates sensor

sets per their probability, is executed. The probabilistic approach improves the cost value when compared to random search by generating sensor sets with low cost. But this approach cannot ensure that the obtained solution has the least cost sensor set. So a genetic algorithm was implemented to obtain the least cost sensor set which is validated using a brute force technique.

### **4.3 Genetic Approach to Sensor Selection**

A genetic algorithm, with sensor numbers coded into genetic strings called chromosomes, is implemented. Each of these chromosomes has an associated fitness value, which is determined by the objective function in Equation (4.11) to be minimized. Each chromosome contains substrings called genes, which in this problem are the sensors, which contribute to fitness of the chromosome. The genetic algorithm (GA) proceeds by taking the population, which is comprised of different chromosomes, each of which is a set of sensors with fitness evaluated for each chromosome. In each successive generation the highest fit survives and thus increases the average fitness. When the GA is being used in the context of function optimization, success is measured by the discovery of strings that represent values yielding an optimum (or near optimum) of the given function [14].

The sensor selection optimization problem using GA can be expressed as minimization equation in (4.11) and with a constraint that the number of sensors for each chromosome being 11 sensors.

Genetic algorithms start with an initial population of individuals. Each individual in the initial population should meet the constraint. The population is arbitrarily initialized within specified bounds. Randomized processes of selection, crossover, and mutation

help population evolve towards better and better regions in the search space. The GA parameters in this study, determined by manual tuning, are given as follows.

Initial population size = 100

Population size = 50

Crossover Probability = 0.9

Mutation Probability = 0.003 per sensor

Maximum Generations = 15

Number of Sensors = 11

**Selection** is a means to prefer better individuals to worse individuals, where better or worse is defined by the fitness function. A nice feature of the selection mechanism is its independence of representation of the individuals, as only the fitness values of the individuals are taken into account. The selection procedure of the individuals in GAs is fitness dependent. For the sensor selection problem a roulette wheel selection algorithm is implemented to select the two individuals for mating to generate a new population.

This can be psuedocoded as:

*Required Fitness = rand \* Total Population Fitness*

*Individual=1;*

***While*** Fitness > *Required Fitness*

*Individual=individual+1;*

***End***

*Select the chromosome at individual index*

**Crossover** is the main variation operator that recombines useful segments from different individuals. An exogenous operator, the crossover probability, indicates the probability

per individual to undergo recombination. For the sensor selection problem the four crossover methods considered are given below.

1. Setdiff Method
2. Two-Point Crossover
3. Uniform Crossover
4. Segmented Crossover

**The Setdiff Method** is applied using the Matlab setdiff command, which returns values of the first set that are not in the second set.

*Setdiff*(A, B) returns values in A that are not in B

Pseudocode for setdiff method crossover can be written as follows.

$$x = \text{setdiff}(A, B); y = \text{setdiff}(B, A)$$

$$\text{child1} = A(1:\text{end-length}(y) \ y)$$

$$\text{child2} = B(1:\text{end-length}(x) \ x)$$

A set difference operation between two chromosomes  $A$  and  $B$  returns the genes of  $A$  that are not in  $B$ . So  $x$  and  $y$  are two distinct sets obtained using the set difference command. A child can be produced by truncating the chromosome  $A$  at one end with a value obtained from finding length of  $y$ . Then append the set  $y$  at this end of  $A$ . Thus a new chromosome with same length is born. This is done similarly to the second child which is born using chromosome  $B$  and set  $x$ .

**Two-Point Crossover** can be implemented by generating two random crossover points.

Before crossover,

A=

S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
----	----	----	----	----	----	----	----	----	-----	-----

B=

S'1	S'2	S'3	S'4	S'5	S'6	S'7	S'8	S'9	S'10	S'11
-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------

Two random crossover points are chosen and after crossover the two child chromosomes will be the following

Child1=

S1	S2	S3	S'4	S'5	S'6	S'7	S8	S9	S10	S11
----	----	----	-----	-----	-----	-----	----	----	-----	-----

Child2=

S'1	S'2	S'3	S4	S5	S6	S7	S'8	S'9	S'10	S'11
-----	-----	-----	----	----	----	----	-----	-----	------	------

### Uniform Crossover

At each sensor, a switch rate of 0.5 is used to check if the sensors in both parents at each position need to be swapped.

Pseudocode for uniform crossover can be written as follows.

*for i=1:length of chrom*

*If rand>0.5*

*swap (s(i) ,s'(i))*

*end*

### Segmented Crossover

For each sensor, a switch rate  $r$  between  $[0, 1]$  is used to check if the sensors in both parents at each position need to be swapped. Pseudocode for segmented crossover can be written as follows.

```

r=rand
for i=1:length of chrom
  If rand>r
    swap (s(i) ,s'(i))

```

Note that this is the same as uniform crossover except in uniform crossover the switch rate is equal to 0.5.

**Mutation**, traditionally referred to as a background operator, operates on the chromosome with a mutation probability per parameter. A simple mutation operator is applied for the sensor selection optimization problem with a probability of 0.003 per sensor. The sensors values are mutated *nmutation* times by selecting a random position in the sensor set.

$$nmutation = pmutate * no. \text{ of parameters}$$

#### 4.4 Eigenvector approach to Sensor Selection

The eigenvector approach is used to determine how much effect the health parameters have on each output [20]. The linear model of the engine for observability analysis is

$$\begin{aligned} \delta x_{k+1} &= \Phi \delta x_k + \Gamma \delta p_k + V_{1k} \\ \delta y_k &= C \delta x_k + D \delta p_k + e_k \end{aligned} \tag{4.12}$$

Now we define

$$\begin{aligned} \tilde{x}(k+1) &= \delta x_{k+1} \\ \tilde{x}(k) &= \delta x_k \\ \text{similarly } \tilde{p}(k+1) &= \delta p_{k+1} \end{aligned}$$

In order to represent the engine model more appropriately, the states are augmented with the parameters. The equations for the augmented model are

$$\begin{aligned} \begin{bmatrix} \tilde{x}(k+1) \\ \tilde{p}(k+1) \end{bmatrix} &= \begin{pmatrix} \Phi & \Gamma \\ 0 & I \end{pmatrix} \begin{bmatrix} \tilde{x}(k) \\ \tilde{p}(k) \end{bmatrix} + V_1(k) \\ \tilde{y}(k) &= [C \quad D] \begin{bmatrix} \tilde{x}(k) \\ \tilde{p}(k) \end{bmatrix} + e(k) \end{aligned} \quad (4.13)$$

To find a relationship between sensors and parameters we perform the analysis assuming that the system is in steady state:

$$\begin{aligned} \tilde{x}(k+1) &= \tilde{x}(k) \\ \text{and } \tilde{p}(k+1) &= \tilde{p}(k) \end{aligned}$$

Hence solving for  $\tilde{y}(k)$  [6] we arrive at

$$\begin{aligned} \tilde{y}(k) &= (C(I - \Phi)^{-1}\Gamma + D)\tilde{p}(k) + C(I - \Phi)^{-1}V_1 + e(k) \\ \text{let } (C(I - \Phi)^{-1}\Gamma + D) &= K_p \end{aligned} \quad (4.14)$$

Now focusing on the relationship between the parameters and the measurements to obtain the principal directions in parameter space from the most sensitive to the least sensitive sensor we use singular value decomposition.

$$\begin{aligned} y &= U\Sigma V^T p \\ &= [U_1 \quad \dots \quad U_s] \begin{pmatrix} \sigma_1 & 0 & \mathbf{0} \\ 0 & \ddots & 0 \\ \mathbf{0} & 0 & \sigma_n \end{pmatrix} [V_1 \quad \dots \quad V_p]^T p \end{aligned}$$

$$\text{let } U\Sigma V^T = K$$

$$[U \quad \Sigma \quad V] = \text{svd}(K_p)$$

$V_1, \dots, V_p$  are the directions in the parameter space with corresponding observabilities  $\sigma_i$ .

$U_1, \dots, U_s$  are the corresponding directions in measured variable space.

The sensitivity of each sensor can be evaluated as follows.

$$KV_i = U\Sigma V^T V_i$$

$$V^T V_i = (V_1 \dots V_p)^T V_i \text{ for } i=1, \dots, p$$

$$V^T V_1 = \begin{bmatrix} V_1^T \\ \vdots \\ V_p^T \end{bmatrix} V_1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$KV_1 = U \Sigma \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= U \begin{pmatrix} \sigma_1 & 0 & \mathbf{0} \\ 0 & \ddots & 0 \\ \mathbf{0} & 0 & \sigma_n \end{pmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$= U \begin{pmatrix} \sigma_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$= \begin{pmatrix} U_{11} & \dots & U_{1s} \\ \vdots & \ddots & \vdots \\ U_{s1} & \dots & U_{ss} \end{pmatrix} \begin{pmatrix} \sigma_1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

$$KV_1 = \begin{pmatrix} \sigma_1 U_{11} \\ \sigma_1 U_{21} \\ \vdots \\ \sigma_1 U_{s1} \end{pmatrix} = \sigma_1 U_1$$

$$\vdots$$

$$KV_p = \sigma_p U_p$$

Thus the sensitivity of the  $i$ -th sensor  $S_i$  can be defined as

$$S_i = \sum_{j=1}^p \sigma_j |U_{ij}| \quad \text{or} \quad S_i = \sum_{j=1}^p \sigma_j (U_{ij})^2 \quad (4.15)$$

$i=1 \dots 12$  (number of sensors)

## CHAPTER V

### RESULTS

#### 5.1 Aircraft Turbofan Engine Sensor Selection

The aircraft turbofan engine model is modified in order to estimate the health parameters. The original 3 states are augmented with the 8 health parameters, so the  $A$  matrix is  $11 \times 11$ . The measurement matrix with 11 rows (corresponding to 11 sensors) can be duplicated with the same set of 11 sensors in case each sensor is used twice. This results in a measurement matrix  $C$  with 11 columns (one for each state) and up to 22 rows (one for each sensor). A change in  $C$  also effects the measurement noise covariance  $R$ , which is constructed to be consistent with  $C$ . These system matrices are used to produce an error covariance  $P$ , which will be designated as  $P_{ref}$  if 22 sensors are used. The total number of possible sensor sets that can be generated is 177146. But in order to reduce computational effort the search space can be narrowed by only concentrating on sets which have 11 sensors and no more than two duplications for each sensor in a sensor set. So the number of distinct sets that can be generated with 11 sensors in each sensor set

and not more than two duplicates of each sensor will be the coefficient of the term  $x^{11}$  in the expression in equation (5.1) [21].

$$(1 + x + x^2)^{11} \quad (5.1)$$

Table I: Coefficients in the expansion

<b>X^n term</b>	<b>Coefficients</b>
<b>X^0 and X^22</b>	<b>1</b>
<b>X^1 and X^21</b>	<b>11</b>
<b>X^2 and X^20</b>	<b>66</b>
<b>X^3 and X^19</b>	<b>275</b>
<b>X^4 and X^18</b>	<b>880</b>
<b>X^ 5 and X^17</b>	<b>2277</b>
<b>X^6 and X^16</b>	<b>4917</b>
<b>X^7 and X^15</b>	<b>9042</b>
<b>X^8 and X^14</b>	<b>14355</b>
<b>X^9 and X^13</b>	<b>19855</b>
<b>X^10 and X^12</b>	<b>24068</b>
<b>X^11</b>	<b>25653</b>

The above expression is built based on number of repetitions allowed for each sensor. The one in the expression describes the absence of a sensor, the term  $x$  corresponds to a sensor used only once; the term  $x^2$  depicts the use of duplicate sensors. The power of the

expression is the number of distinct sensors used, which is 11 in case of our aircraft turbofan engine problem. The coefficient of  $x^{11}$  using this expression is 25653, which is the number of distinct sensor sets that can be generated using 11 sensors for each sensor set. The generation of all 25653 sensor sets taking into consideration constraints such as no more than one repetition of each sensor will be a tedious approach.

Results from two different search algorithms to find the best sensor set from the narrowed search space is discussed in this chapter. Section 5.2 discusses the brute force sensor selection for all 25653 sets. Section 5.3 gives the sensitivities of each sensor obtained using the eigenvector approach. Section 5.4 discusses the results obtained using the probabilistic approach and Section 5.5 explains the results obtained using two different sensor cost models using a genetic algorithm.

## **5.2 Brute Force Sensor Selection**

A brute force algorithm was designed which came up all 25653 distinct sensors sets. The fitness of these sensor sets was found using two different cost models for the sensors. In the first case all the sensors are assumed to have a cost of 1000. The other more realistic model used relative cost by considering the lifetime cost of each sensor and by taking into consideration the associated costs of added weight, data acquisition and signal processing, software verification and validation, maintenance cost to replace a faulty sensor, and the physical cost of sensor itself. As per the information from NASA Glenn research center, rotor speed sensors are the cheapest, pressure sensors the most

expensive, and temperature sensors somewhere in between [22]. The sensors that reside in higher temperature regions of the engine are probably going to be more expensive than those that reside in more benign regions. The cost of duplicating a sensor is considered to be 75% of the cost of the first sensor as the design; cabling and software validation and verification costs are common. For the purpose of this research the relative costs in Table I were provided by the NASA Glenn Research Center.

Table II: Relative sensor costs

<b>SENSOR</b>	<b>RELATIVE COST</b>
Core rotor speed	1.0
Percent low pressure spool rotor speed	1.0
Fan exit pressure	2.0
Booster inlet pressure	2.0
HPC inlet temperature	1.5
HPC exit temperature	1.5
Bypass duct pressure	2.0
HPC exit pressure	2.5
LPT blade temperature	2.5
LPT exit temperature	2.0
LPT exit pressure	2.5

The results obtained from a brute force search show that sets with repetitions were less fit than the nominal set which had all 11 sensors in it. The position of the sensor set with sensors 1 through 11 is at 978 out of 25653 sets when all sensors are assumed to have equal sensor costs. When relative sensor costs are taken into account the position of the

full sensor set is at 1102. The least cost sensor set obtained after evaluating the fitness of the sensor sets using the two different sensor cost models are tabulated in Table III.

Table III: Least cost sensor sets with two different sensor cost models

COST	Sensor set with sensor cost of 1000 for each sensor	Sensor set with relative sensor cost as shown in Table II
	1, 2, 4, 5, 5, 6, 6, 7, 7, 8, 10	1, 1, 2, 2, 4, 5, 5, 6, 7, 8, 10
	2.1687	2.0957

A histogram in Figure 5.1 shows the top 2000 sensor sets with same sensor cost for each sensor. The nominal sensor set having sensors 1 through 11 has a cost of 2.2693. The distribution of these sets can be approximated as a Gaussian probability density function. The top ten sensor sets corresponding to the two different cost models are tabulated below in Table IV and Table V.

Table IV: Top 10 sensor sets with sensor cost of 1000

SENSOR SET											COST
<b>1</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>6</b>	<b>7</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.1687</b>
1	2	4	4	5	5	6	7	7	8	10	2.1697
1	3	4	4	5	5	6	7	7	8	10	2.1699
1	2	4	5	5	6	7	7	8	8	10	2.1721
1	3	4	5	5	6	7	7	8	8	10	2.1726
1	3	3	4	5	5	6	7	7	8	10	2.1731
1	1	2	2	4	5	5	6	7	8	10	2.1737
1	1	2	4	5	5	6	7	7	8	10	2.1741
1	1	3	4	5	5	6	7	7	8	10	2.1745
1	2	2	4	5	5	6	7	7	8	10	2.1745

Table V: Top 8 sensor sets with relative sensor cost

SENSOR SET											COST
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	3	5	5	6	7	8	10	2.1040
1	1	2	4	5	5	6	6	7	8	10	2.1114
1	2	2	4	5	5	6	6	7	8	10	2.1117
1	1	2	4	5	5	6	7	7	8	10	2.1146
1	2	2	4	5	5	6	7	7	8	10	2.1150
1	1	2	4	4	5	5	6	7	8	10	2.1185
1	2	2	4	4	5	5	6	7	8	10	2.1192

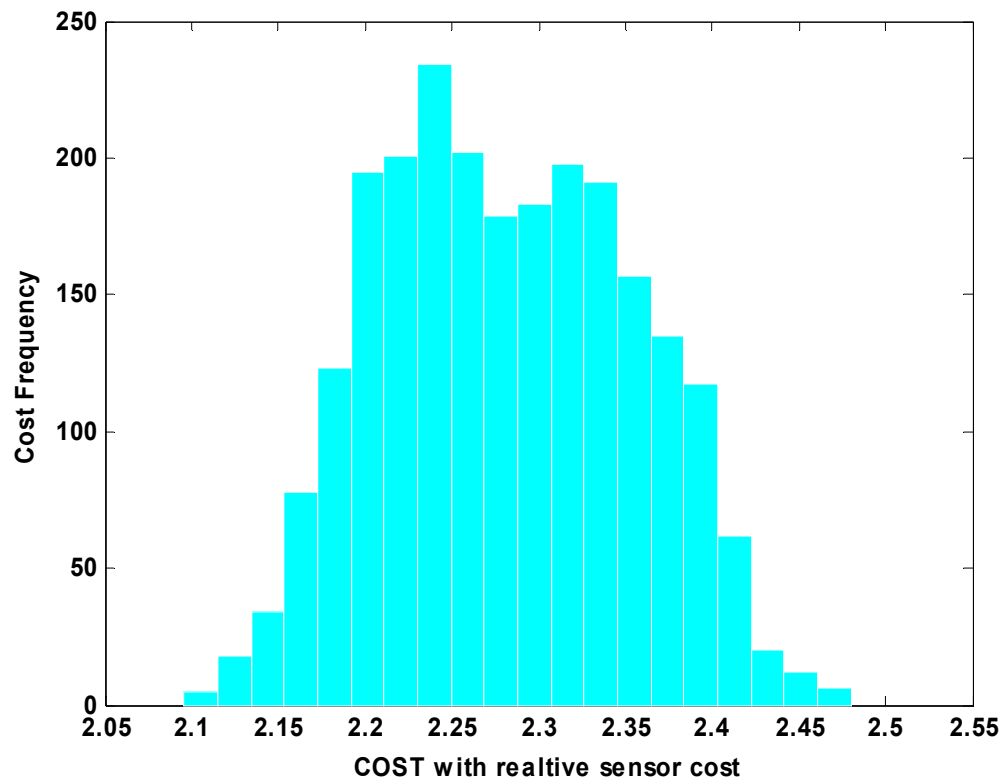


Figure 5.1: Histogram of the distribution of the top 2000 sensor sets with relative cost

### 5.3 Probabilistic Search for Sensor Selection

In this section a directed search approach is used to generate an optimal sensor set for health parameter estimation. Analysis of sensor selection optimization was considered by

considering 11 sensors generated for each sensor set. A financial cost of 1000 is assumed for each sensor, although this work can be extended to any cost for each sensor, since it may be that different sensors have different financial costs. During this sensor selection process we limited the number of duplications for each sensor so that each sensor could be selected no more than twice. In a single simulation run the best 100 sensor sets are selected from 10000 sets that are randomly generated. This results in a probability for each sensor belonging to the top 1% of all sensor sets. The final probabilities of each sensor are shown in Table VI.

Table VI: Sensor selection probabilities using random search method

<b>SENSOR</b>	1	2	3	4	5	6	7	8	9	10	11
<b>PROBABILITY</b>	0.11	0.09	0.08	0.08	0.09	0.08	0.08	0.11	0.07	0.07	0.08

The probability of each sensor being in top one percent is used to generate sensor sets in the probabilistic approach. The result is plotted as a histogram as shown in Figure 5.2. On observation it was found that nearly 30% of the sensor sets generated using the probabilistic approach had a lower metric than the best metric obtained by the random search approach. The best cost obtained using the probabilistic approach was 3 – 4% less than the best cost obtained using a random approach for sensor selection.

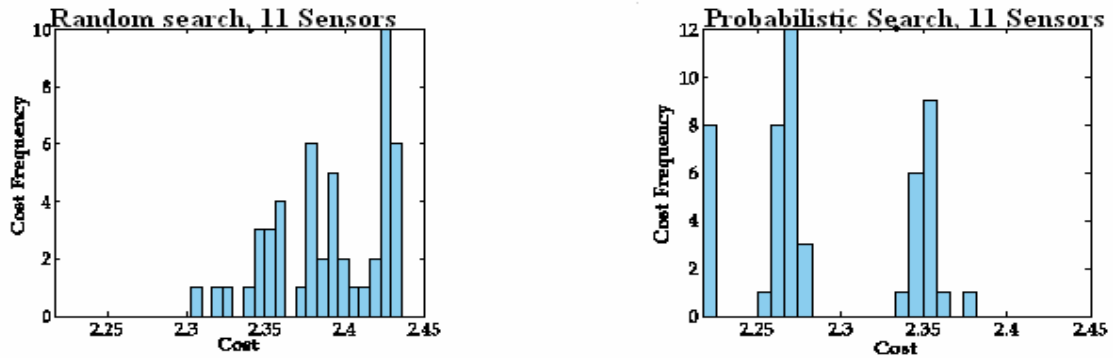


Figure 5.2: Sensor set cost vs. cost frequency

The top three sensor sets obtained using a random search algorithm and the probabilistic search algorithms are shown in Tables VII and VIII.

Table VII: Sensor sets using random search algorithm

Sensor Set	Metric
1, 1, 2, 4, 5, 7, 8, 9, 10, 11, 11	2.3064
1, 1, 3, 3, 4, 4, 5, 6, 8, 9, 11	2.31
1, 2, 2, 3, 3, 5, 7, 8, 9, 10, 11	2.3153

Table VIII: Sensor sets using probabilistic search algorithm

Sensor Set	Metric
1, 2, 4, 5, 6, 7, 8, 8, 9, 10, 11	2.2028
1, 1, 2, 3, 4, 5, 5, 6, 7, 8, 11	2.206
1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 11	2.2122

The results of the probabilistic approach show that it was unable to find the least cost set which was obtained using brute force. So a genetic algorithm discussed in Section 4.3 was implemented to search for the least cost set.

#### 5.4 Eigenvector approach for sensor selection

In this section an eigenvector approach for sensor selection. This is based on determining the effect that the health parameters have on each sensor output. So the sensitivity of each of the sensors is calculated using the method defined in Section 4.4. The sensitivity of each sensor is tabulated in Table IX.

Table IX: Sensor sensitivities for the MAPSS model

SENSOR#	SENSOR DESCRIPTION	SENSITIVITY
2	Percent low pressure spool rotor speed	8.4
11	LPT exit pressure	10.0
3	Fan exit pressure	10.5
7	Bypass duct pressure	13.7
4	Booster inlet pressure	16.7
5	HPC inlet temperature	159.0
8	HPC exit pressure	231.5
9	LPT blade temperature	541.1
1	Core rotor speed	588.1
6	HPC exit temperature	741.2
10	LPT exit temperature	901.4

These sensitivities can be used to select the sensors in a sensor set.

## 5.5 Genetic approach for Sensor Selection

The GA approach for sensor selection proceeds as described in Section 4.3. The initial population is generated randomly as per the problem-specific constraints (e.g., maximum number of allowable duplicates). The GA parameters in this study, determined by manual tuning, are given as follows.

Initial population size = 100

Population size = 50

Crossover Probability = 0.9

Mutation Probability = 0.003 per sensor

Maximum Generations = 15

Number of Sensors = 11

The above parameters were obtained by performing simulations with five different values for each parameter and running a GA. The sensitivity of genetic algorithm is not much affected by choosing the parameters. But to minimize the time required computing the fitness was achieved by using above set of parameters to converge to global minimum. The behavior of the GA for different crossover methods and the least cost sensor set obtained using these methods is discussed in this section. The crossover methods are defined in detail in Section 4.3 and are given as follows.

1. Setdiff crossover
2. Two-Point crossover
3. Uniform crossover
4. Segmented crossover

### 5.5.1 GA results using identical sensor costs

The least cost sensor sets obtained using Setdiff crossover through 15 generations is listed below in Table X. The genetic algorithm for sensor selection at the end of 15 generations ended up with a least cost 2.1699, which is at position three in the brute force approach. Figure 5.3 shows the plots for average, minimum, maximum and sum fitness over 15 generations using the Setdiff crossover method.

Table X: Least cost sensor sets obtained using setdiff method with identical cost for sensors

SENSOR SET											COST
1	1	2	4	4	5	6	7	8	8	10	2.2186
1	1	2	4	4	5	6	7	8	8	10	2.2186
1	1	2	3	4	5	5	6	7	8	10	2.1924
1	3	3	4	4	5	5	6	7	8	10	2.1789
1	3	3	4	4	5	5	6	7	8	10	2.1789
1	3	3	4	4	5	5	6	7	8	10	2.1789
1	3	3	4	4	5	5	6	7	8	10	2.1789
1	3	3	4	5	5	6	6	7	8	10	2.1786
1	3	3	4	5	5	6	6	7	8	10	2.1786
1	3	3	4	5	5	6	6	7	8	10	2.1786
<b>1</b>	<b>3</b>	<b>4</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.1699</b>
1	3	4	4	5	5	6	7	7	8	10	2.1699
1	3	4	4	5	5	6	7	7	8	10	2.1699
1	3	4	4	5	5	6	7	7	8	10	2.1699

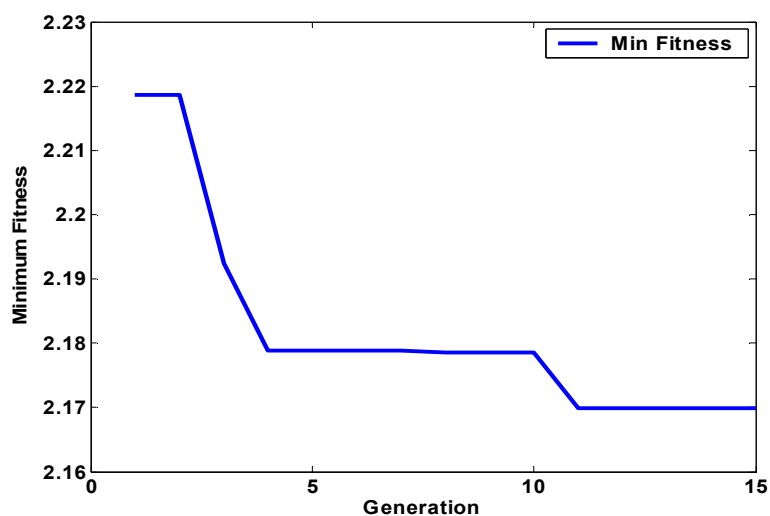


Figure 5.3 GA run with setdiff crossover using identical cost for sensors

The average cost at the end of 15 generations is 2.1820. At the end of 15 generations the minimum fitness and mean fitness are almost same which shows the convergence of population to the least cost set.

Two-point crossover was implemented and the least cost sensor sets are as shown in Table XI. The least cost set obtained using two-point crossover is the same as the best set obtained using brute force. Figure 5.4 shows the plot for the minimum cost over 15 generations using two-point crossover.

Table XI: Least cost sensor sets obtained using two-point crossover with identical cost for sensors

SENSOR SET											COST
1	3	4	5	5	6	7	8	9	11	11	2.2334
1	1	3	4	5	5	6	7	8	9	10	2.2113
1	1	2	3	3	5	6	7	7	8	10	2.2037
1	1	3	4	5	5	6	7	8	10	10	2.1927
1	1	3	4	4	5	5	6	7	8	10	2.1798
1	1	2	2	4	5	5	6	7	8	10	2.1737
1	1	2	2	4	5	5	6	7	8	10	2.1737
1	1	2	2	4	5	5	6	7	8	10	2.1737
1	2	4	4	5	5	6	7	7	8	10	2.1697
1	2	4	4	5	5	6	7	7	8	10	2.1697
<b>1</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>6</b>	<b>7</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.1687</b>
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687

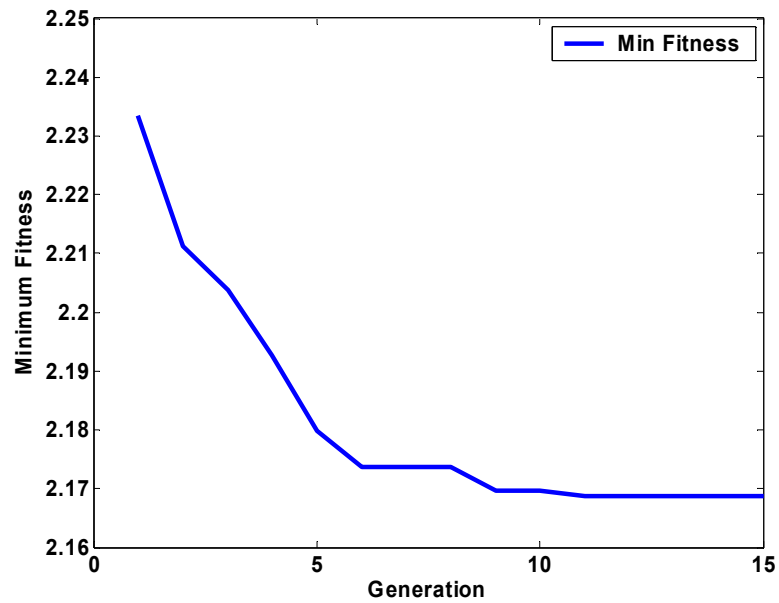


Figure 5.4 GA run with two-point crossover using identical cost for sensors

Using uniform crossover the genetic algorithm ended with a least cost of 2.1687, which is the same as the least cost set obtained using brute force. The least cost sets obtained in 15 generations are shown in Table XII. But from Figure 5.5 it is seen that the genetic algorithm obtains a sensor set with a cost of 2.1617, which actually less than that is obtained using brute force. Because of mutation, sometimes the sensor sets end up with more than the allowed number of duplicate sensors. So this particular sensor set has five triplets, which evaluates to a fitness value even less than the fitness obtained using brute force. As the algorithm constrains sensor sets to have only one duplicate, this set is modified to have met the constraints using the sensitivity array obtained using eigenvector approach.

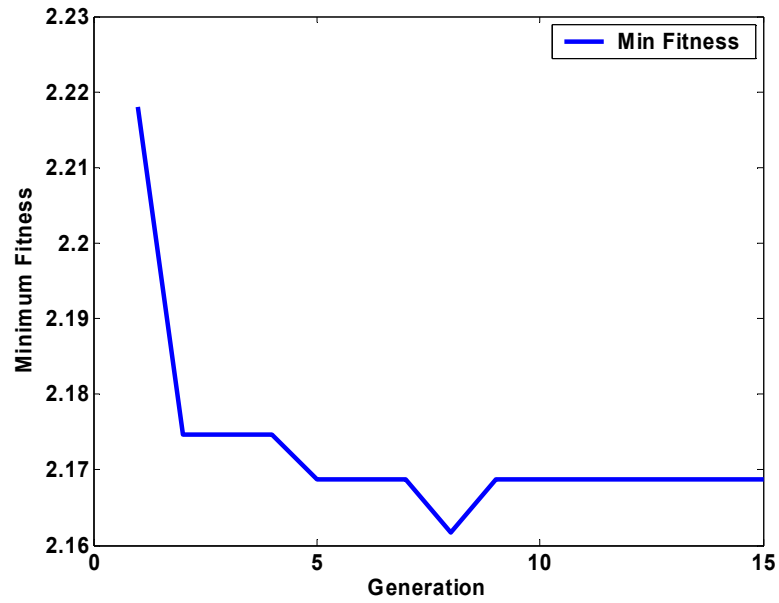


Figure 5.5 GA run with uniform crossover using identical cost for sensors

Table XII: Least cost sensor sets obtained using uniform crossover with identical cost for sensors

SENSOR SET											COST
1	3	4	5	5	6	7	8	9	11	11	2.218
1	1	3	4	5	5	6	7	8	9	10	2.1747
1	1	2	3	3	5	6	7	7	8	10	2.1747
1	1	3	4	5	5	6	7	8	10	10	2.1747
<b>1</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>6</b>	<b>7</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.1687</b>
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>5</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.1617</b>
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687

Implementation of segmented crossover in the GA for the sensor selection problem gave good results. The least cost sensor sets for all 15 generations are in Table XIII. The least cost obtained using segmented crossover is 2.1687, which same as the least cost obtained using brute force.

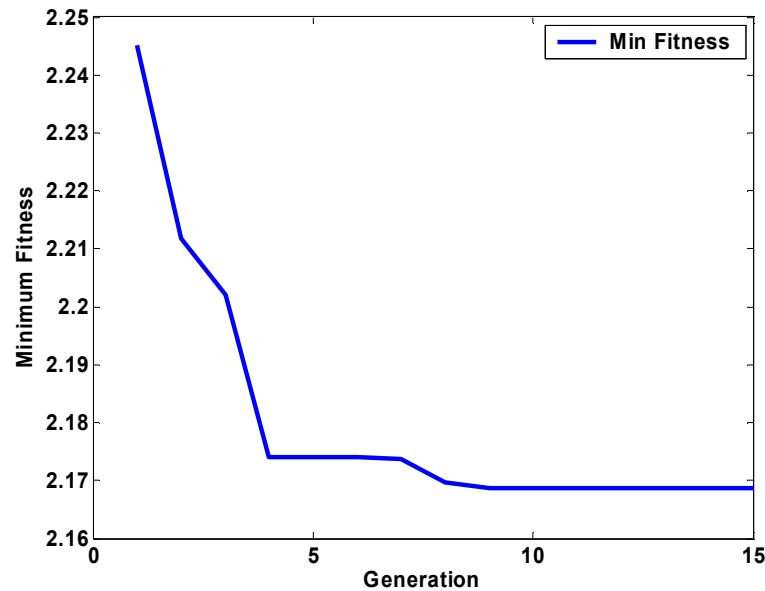


Figure 5.6 GA run with segmented crossover using identical cost for sensors

Table XIII: Least cost sensor sets obtained using segmented crossover with identical cost for sensors

SENSOR SET											COST
1	1	2	4	4	5	6	7	8	10	11	2.2452
1	1	2	4	5	5	6	7	8	10	11	2.2118
1	2	4	4	5	5	6	7	8	9	9	2.2020
1	1	2	4	5	5	6	7	7	8	10	2.1741
1	1	2	4	5	5	6	7	7	8	10	2.1741
1	1	2	4	5	5	6	7	7	8	10	2.1741
1	1	2	2	4	5	5	6	7	8	10	2.1737
1	2	4	4	5	5	6	7	7	8	10	2.1697
<b>1</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>6</b>	<b>7</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.1687</b>
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687
1	2	4	5	5	6	6	7	7	8	10	2.1687

### 5.5.2 GA results using relative sensor costs

The results for the sensor selection problem with the relative sensor costs as shown in Table II are shown in this section. Almost all the methods reached the absolute minimum cost (which was obtained using a brute force search) of 2.095. The results are tabulated here according to the type of crossover applied and the plots are shown below. The least cost sets obtained using the segmented crossover with relative sensor cost is shown in Table XIV. This is same as the least cost obtained using brute force with relative sensor cost. Figure 5.7 shows the minimum fitness curve for 15 generations.

Table XIV: Least cost sensor sets obtained using setdiff crossover with relative sensor cost

SENSOR SET											COST
1	3	4	5	5	6	7	8	9	11	11	2.2314
1	1	3	4	5	5	6	7	8	9	10	2.2104
1	1	2	3	3	5	6	7	7	8	10	2.1256
1	1	3	4	5	5	6	7	8	10	10	2.1146
1	1	2	2	3	5	5	6	7	8	10	2.1040
1	1	2	2	3	5	5	6	7	8	10	2.1040
1	1	2	2	3	5	5	6	7	8	10	2.1040
1	1	2	2	3	5	5	6	7	8	10	2.1040
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.0957</b>

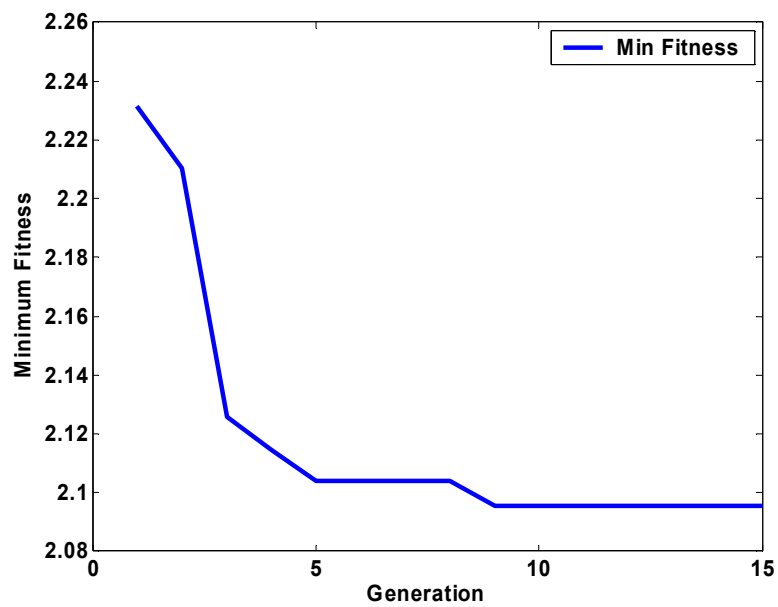


Figure 5.7 GA run with setdiff crossover using relative cost for sensors

Table XV and Figure 5.8 give the results obtained for least cost set using Two-point crossover. The least cost set has the same minimum cost obtained using brute force.

Table XV: Least cost sensor sets obtained using two-point crossover with relative cost for sensors

SENSOR SET											COST
1	1	2	2	3	5	5	6	7	8	9	2.1429
1	1	2	2	3	5	5	6	7	8	9	2.1429
1	1	2	2	3	5	5	6	7	8	9	2.1429
1	1	2	2	3	5	5	6	7	8	10	2.104
1	1	2	2	3	5	5	6	7	8	10	2.104
1	1	2	2	3	5	5	6	7	8	10	2.104
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.0957</b>

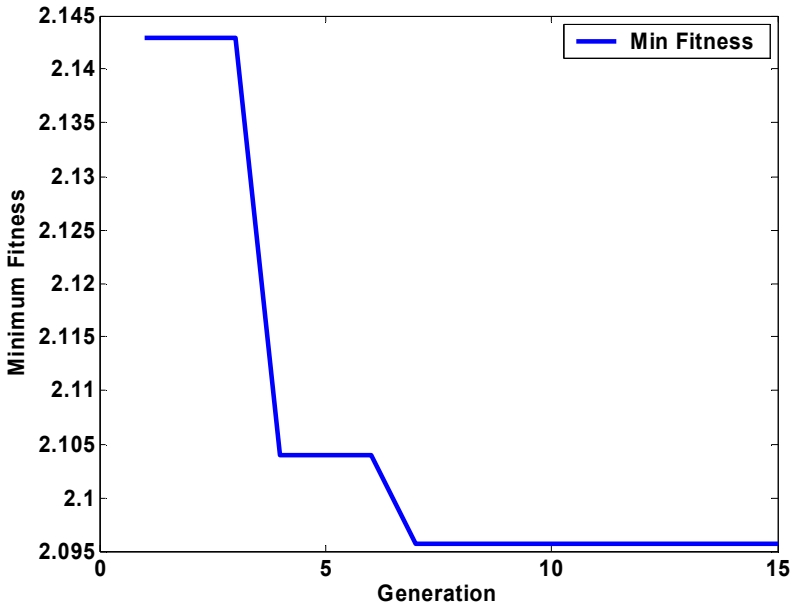


Figure 5.8 GA run with two-point crossover using relative cost for sensors

The least cost sensor sets obtained using uniform crossover over 15 generations are shown in Table XVI. The minimum fitness plot in Figure 5.9 shows convergence of GA to least cost set at generation 4.

Table XVI: Least cost sensor sets obtained using uniform crossover with relative cost for sensors

SENSOR SET											COST
1	3	4	5	5	6	7	8	9	11	11	2.2347
1	1	3	4	5	5	6	7	8	9	10	2.1995
1	1	2	3	3	5	6	7	7	8	10	2.1891
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
1	1	2	2	4	5	5	6	7	8	10	2.0957
<b>1</b>	<b>1</b>	<b>2</b>	<b>2</b>	<b>4</b>	<b>5</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>10</b>	<b>2.0957</b>

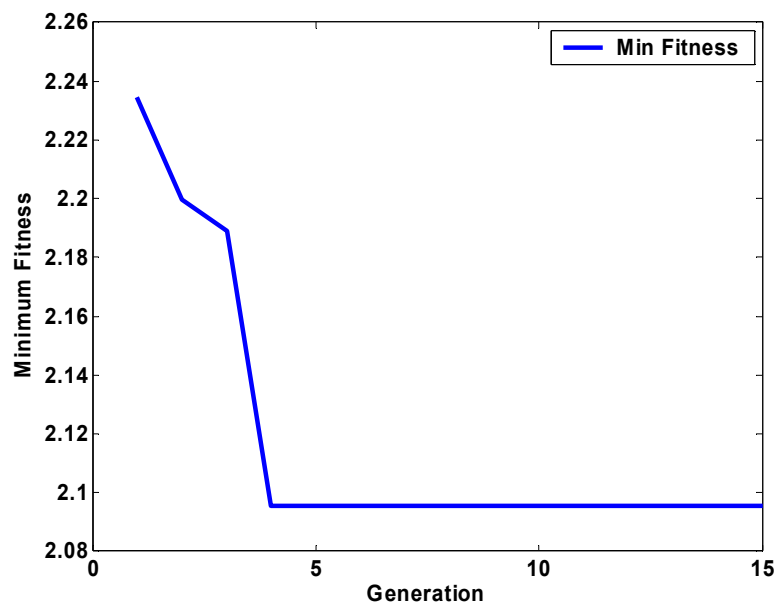


Figure 5.9 GA run with uniform crossover using relative cost for sensors

Segmented crossover implementation to GA resulted in minimum cost set obtained using brute force. The sensor sets obtained over 15 generations are listed in Table XVII. Figure 5.10 shows the convergence of cost to the least cost over 15 generations.



## CHAPTER VI

### CONCLUSIONS AND FUTURE WORK

#### 6.1 Conclusions

The brute force approach of searching all 177146 combinations of sensors could be too computationally expensive. So the search space can be narrowed down by selecting only sets with no more than one repetition per sensor. This results in 25653 distinct sensor sets. The results of the brute force helped prove the how good the algorithms searched the entire search space. Finding all possible sets having 11 sensors with no more than one duplicate was a tedious task. So a randomization technique to generate the optimal sensor set was implemented which reduced the computational complexity. This method proceeds by randomly generating a small number of sensor sets, computing their metric, obtaining the probability of each sensor being in a good sensor set, and then using those probabilities to generate a sensor set with minimum cost. Comparing the results of the probabilistic approach with brute force, we saw that the least cost obtained using the former method is 2.2028, compared to the least cost of 2.1687 that was obtained using

brute force. So a GA was developed by coding the sensor sets into a chromosome. The results of the GA for sensor selection were approximately the same with different crossover methods implemented. The least cost sets obtained using different crossover methods for a sensor cost of 1000 for each sensor are tabulated in Table XVIII.

Table XVIII: Least cost sensor sets obtained with a sensor cost of 1000 for each sensor

Method	Best Sensor Set											Cost
Setdiff Crossover	1	3	4	4	5	5	6	7	7	8	10	2.1699
Two-point Crossover	1	2	4	5	5	6	6	7	7	8	10	2.1687
Uniform Crossover	1	2	4	5	5	6	6	7	7	8	10	2.1687
Segmented Crossover	1	2	4	5	5	6	6	7	7	8	10	2.1687

When relative costs of sensor are taken into account, all of the crossover methods resulted in a least cost sensor set that was the same as that obtained using brute force, as shown in Table XIX. We verified that the systems that resulted were observable.

Table XIX: Least cost sensor sets obtained with relative cost of each sensor

Method	Sensor Set											Cost
Setdiff Crossover	1	1	2	2	4	5	5	6	7	8	10	2.0957
Two-point Crossover	1	1	2	2	4	5	5	6	7	8	10	2.0957
Uniform Crossover	1	1	2	2	4	5	5	6	7	8	10	2.0957
Segmented Crossover	1	1	2	2	4	5	5	6	7	8	10	2.0957

Relative optimization of cost with different methods can be summarized in the following Table XX and Table XXI.

Table XX: Relative cost optimization with identical sensor cost

Method	Sensor Sets Evaluations required	Computation (Minutes)	Best Cost	Sensor set
Exhaustive Search (Identical Sensor Cost)	25653	78	2.1687	1,2,4,5,5,6,6,7,7,8,10
Probabilistic Search (Identical Sensor Cost)	10000	30	2.2028	1,2,4,5,6,7,8,8,9,10,11
Genetic Algorithms (Identical Sensor Cost)	850	3	2.1687	1,2,4,5,5,6,6,7,7,8,10

Table XXI Relative cost optimization with relative sensor cost

Method	Sensor Sets Evaluations required	Computation (Minutes)	Best Cost	Sensor set
Exhaustive Search (Identical Sensor Cost)	25653	78	2.0957	1,1,2,2,4,5,5,6,7,8,10
Genetic Algorithms (Identical Sensor Cost)	850	3	2.0957	1,1,2,2,4,5,5,6,7,8,10

The sensors present in the best sensor set are listed below in Table XXII and the sensors eliminated from the best set are listed below in Table XXIII.

Table XXII: Sensors in the best sensor set

Sensor Number	Sensor	Relative Cost
1	Core rotor speed	1.0
2	Percent low pressure spool rotor speed	1.0
4	Booster inlet pressure	2.0
5	HPC inlet temperature	1.5
6	HPC exit temperature	1.5
7	Bypass duct pressure	2.0
8	HPC exit pressure	2.5
10	LPT exit temperature	2.0

Table XXII: Sensors eliminated from the best sensor set

Sensor Number	Sensor	Relative Cost
3	Fan exit pressure	2.0
9	LPT blade temperature	2.5
11	LPT exit pressure	2.5

## 6.2 Future work

This research involved the use of probability theory to obtain the confidence that the final sensor set that is selected with this method is within some percentage of the absolute best sensor set that is available. For future work, joint probabilities can be obtained and used

in the directed search in the same way that single probabilities have been used thus far. A financial cost of 1000 for each sensor needs a variable to scale it so that significance of trace for the error covariance matrix is not lost. So the present cost function can be modified to have term  $\alpha$  in front of financial cost as in equation 6.1.

$$J = \sum_{i=1}^n w_i \left( \sqrt{\frac{P_{i,i}(k|k)}{P_{i,i}^{ref}(k|k)}}} \right) + \alpha \times \frac{\text{financial cost}}{\text{ref financial cost}} \quad (6.1)$$

$$\text{Where } w_i = \begin{cases} 0 & i=1,\dots,3 \\ 1 & i=4,\dots,11 \end{cases},$$

Thus different values of  $\alpha$  will scale the financial cost so that the trace obtained from error covariance matrix P has significance will be other future work. Particle swarm optimization [23] is a recently proposed algorithm by James Kennedy and R. C. Eberhart in 1995, motivated by social behavior of organisms such as bird flocking and fish schooling. PSO as an optimization tool provides a population-based search procedure in which individuals called *particles* change their *position* (state) with time. In a PSO system, particles *fly* around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a *neighboring* particle, making use of the best position encountered by itself and its neighbor. Thus, as in modern GAs, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation. So implementation of evolutionary algorithms like particle swarm optimization is another idea to solve the sensor selection problem.

## BIBLIOGRAPHY

- [1] Jonathan Litt, Sensor placement for aircraft propulsion system health management, NASA Glenn Research Center. (Unpublished Report)
- [2] Lucy Y. Pao, Michael Kalandros, John Thomas, Controlling Target Estimate Covariance in Centralized Multisensor Systems, American Control Conference, Philadelphia, pp 2749-2753, June 1998.
- [3] Michael Kalandros, Lucy Y. Pao, Yu-Chi Ho, Randomization and Super-Heuristics in choosing Sensor Sets for Target Tracking Applications, Proceeding of the IEEE Conference on Decision and Control, Phoenix, AZ, December 1999
- [4] The aircraft gas turbine engine and its operation, United Technologies Corporation, Aug 1988.
- [5] SIFCO industries Inc, <http://www.sifco.com/bookhtml/Aeol2.GIF>
- [6] American Society of Mechanical Engineers, <http://www.asme.org/igti/resources/articles/intro2gta.html> for turbofan images
- [7] Aerospace web, <http://www.aerospaceweb.org/question/propulsion/q0033.shtml>
- [8] Mary Bellis, <http://inventors.about.com/library/inventors/bljetenginetypes.htm>
- [9] Khary I. Parker and Kevin J. Melcher, "Modular Aero Propulsion System Simulation (MAPSS) User's Guide," NASA/TM 2004-212968
- [10] MATLAB, "Getting Started with MATLAB," The Math Works, Inc., 1996
- [11] Simulink, "Using Simulink," The Math Works Inc., 2000
- [12] Panos J. Antsaklis and Anthony N. Michel, "Linear Systems," The McGraw-Hill Companies, Inc. 1997

- [13] David.E.Goldberg, "Genetic Algorithms in search, Optimization and Machine learning," Addison-Wesley Publishing Company Inc, 1989.
- [14] Kiego Watanabe, M.M.A.Hashem "Evolutionary Computations -New Algorithms and their Applications to Evolutionary Robots," Series: Studies in Fuzziness and Soft Computing, Vol. 147, Springer, 2004.
- [15] Collin Reeves, Johnathan.E.Rowe, "Genetic algorithms-Principles and Perspectives," Coventry University, UK, 2002.
- [16] T.C.Fogarty, "Varying the probability of mutation in the genetic algorithm," J.D.Schaffer, Proceedings of 3<sup>rd</sup> International Conference on Genetic Algorithms, Morgan Kaufmann, Los Altos, CA, 104-109, 1989.
- [17] C.R.Reeves, "A genetic algorithm for flowshop sequencing," Computers & Operations Research, 22, pp. 5-13, 1994
- [18] K.A.De Jong, "An analysis of the behaviors of a class of genetic adaptive systems," Doctoral Dissertation, University of Michigan, Ann Arbor, Michigan.[[http://cs.gmu.edu/~eclab/kdj\\_thesis.html](http://cs.gmu.edu/~eclab/kdj_thesis.html) (PDF)], 1994
- [19] J.H.Holland "Building Blocks, cohort genetic algorithms, and hyperplane-defined functions," Evolutionary Computation, Vol 8, pp. 373-391, 2000.
- [20] Brent J.Brunell, Daniel E.Viassolo, Ravi Prasanth, "Model Adaptation and Nonlinear Model Predictive Control of an Aircraft Engine," ASME Turbo Expo, Vienna, 2004.
- [21] Dr. Math, Math Forum, <http://www.mathforum.com>, March 2003
- [22] D.L. Simon, NASA Glenn Research Center, Personal communication, Nov. 2004

[23] James Kennedy, Russell C. Eberhart, with Yuhui Shi, *Swarm Intelligence*, Morgan Kaufmann, 2001.