MASTER'S THESIS

TURBOJET ENGINE PERFORMANCE MODELLING USING MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

Hicham Rifai

Department of Mathematics
CHALMERS UNIVERSITY OF TECHNOLOGY
GÖTEBORG UNIVERSITY
Göteborg Sweden 2005

Thesis for the Degree of Master of Science

TURBOJET ENGINE PERFORMANCE MODELLING USING MULTI-OBJECTIVE OPTIMIZATION ALGORITHMS

Hicham Rifai

Department of Mathematics
Chalmers University of Technology and Göteborg University
SE-412 96 Göteborg, Sweden
Göteborg, August 2005

Matematiskt centrum
Göteborg 2005

Acknowledgements

Finally!! I have come to the end of my journey at Chalmers. I really would like to thank my supervisors: Markus Wallin, who gave me the possibility to get into the research field of Gas turbines and also many thanks to Michael Patriksson who keeps encouraging me to pursue my work in optimization. I thank Ivar Gustafsson, Ann-Britt Karlsson, Stig Larsson for this nice master program, they were a second family for me in Sweden. Thank you very much. I also thank all my friends and all my class mates. I would like to thank my dear family for their encouragement and their support, especially my mother Fatima Kabbour and my father Driss Rifai.

Abstract

Many real world problems are multi-objective in nature. There are many methods that solve this kind of optimization problem. They can be classified into two classes: genetic algorithms and classical methods. Instead of one solution, both methods find a whole set of solutions. Genetic algorithms seem to attract many researchers due to their robustness and also to their degree of generality, while others prefer methods based on classical optimization algorithms. The choice of one approach over another depends much on the nature of the problem. If we have enough information about the optimization problem, it is better to use methods based on classical optimization algorithms. If not, it is better to use multi-objective genetic algorithms or evolution algorithms. In this thesis we consider a design optimization problem. The task is to design a jet engine by specifying certain parameters such as pressure ratio in the turbine. We run a flight simulation in Matlab, which allows us to calculate the fuel consumption as well as the engine weight for different input parameters. Here the input parameters are decision variables while the objective functions are the fuel consumption and the engine weight. We applied two multi-objective evolution algorithms and one method based on classical optimization algorithms.

Contents

Background

In this thesis we will study the performance of two sets of algorithms applied to a multi-objective optimization problem: genetic Algorithms, and classical methods. The application is the design of an efficient engine that weighs less, and consumes less fuel. Thus we have two objectives, with variables that represent the inlet temperature of the turbine, the pressure ratio, and the engine diameter.

The first part of this thesis introduces the gas turbine theory, and describes procedures used in a flight simulation. The second part deals with multi-objective optimization. It defines the general problem, and lists some techniques and algorithms that solve this specific kind of problem. In the last part of this thesis we make some comparisons between the two methods based on the results in the last chapter.

1 Turbojet Engine

The gas turbine engine is based on converting the fuel energy into a high energy gas stream. Basically, there are five components in the turbojet engine: the intake, the compressor, the turbine, the combustion chamber, and finally the exhaust nozzle [?].

1.0.1 The intake

The intake is a critical part of the engine and has a significant effect on both the safety and the efficiency of the engine. The main task of the intake is to ensure that the air entering the compressor has a uniform pressure and velocity in all flight conditions in order to avoid surging within the compressor. There are different types of intakes; we could mention the S-bend shape and the straight-through intake. It was observed that the intake behaves like the nozzle at low forward speed, while at higher speed the intake decelerate the air and rise the pressure ratio from the atmospheric pressure p_a to p_1 where p_1 is the pressure ratio at the compressor inlet.

1.1 The compressor

There are two main types of compressors. The first one is called the centrifugal compressor. It consists of a stationary disc containing a rotating impeller with divergent passages. The center of the disc is called the impeller eye, the air is sucked into the impeller eye and whirled around at high speed.

Diffusion occurs at the divergent passages, it causes a rise in the stagnation pressure ratio and the deceleration of the air speed within the compressor. The centrifugal compressors were used by the early generation of fighters and also by the first civil transportation aircrafts. Later, centrifugal compressors were replaced by axial compressors because axial compressors produce more pressure rise ratio over the compressor. The axial compressor consists of a series of stages, each stage consists of a row of rotor blades followed by stator blades. Axial compressors accelerate the working fluid by the rotor and then decelerate it by the stator this process is repeated in many stages to yield the desired overall pressure rise ratio. This type of compressor is more complicated and requires a careful design, especially in designing the blades of the rotor or the stator.

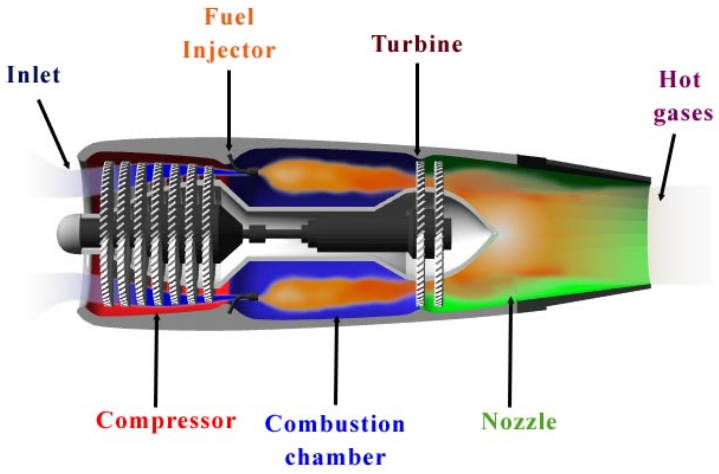

Figure 1: Turbojet engine components.

1.2 Performance of a compressor

The performance of a compressor is illustrated by plotting delivery pressure and temperature versus the mass flow fixed for various rotational speeds (Figure 2). We consider a dimensionless analysis technique [?]. To plot the compressor characteristic, we combine many physical variables that affect the performance of the compressor into a dimensionless group of variables:

$$\frac{p_{02}}{p_{01}}, \frac{T_{02}}{T_{01}}, \frac{m\sqrt{RT_{01}}}{D^2 p_{01}}, \frac{ND}{RT_{01}} Function(\frac{p_{02}}{p_{01}}, R\,\frac{T_{02}}{T_{01}}, \frac{m\,T_{01}}{p_{01}}, \frac{N}{\sqrt{T_{01}}}) = 0;$$

The compressor characteristic is drawn from two parameters; $m\sqrt{T_{01}}/p_{01}$ and $N\sqrt{T_{01}}$. These are equivalent to $m\sqrt{\theta}/\delta$ and $N\sqrt{\theta}$ respectively where $\theta = T_{01}/T_{ref}$ and $\delta = p_{01}/p_{ref}$. The reference is normally corresponding to the I.S.A at sea level.

In Figure 2 we can see seven curves, representing seven rotational speeds. We should keep in mind that these curves are obtained from the pressure ratio r_c in the compressor versus the mass flow $m\sqrt{\theta}/\delta$. The points that lie in the extremities of each curve represent a choking point or a surging point. The choking point represents the maximum delivery at maximum rotational speed when there is no further increase in the mass flow.

As stated earlier, for centrifugal compressors the static pressure rise occurs in the diffuser and the impeller eye. When the pressure rise reaches its maximum any increase in the mass flow will cause a decrease in the pressure

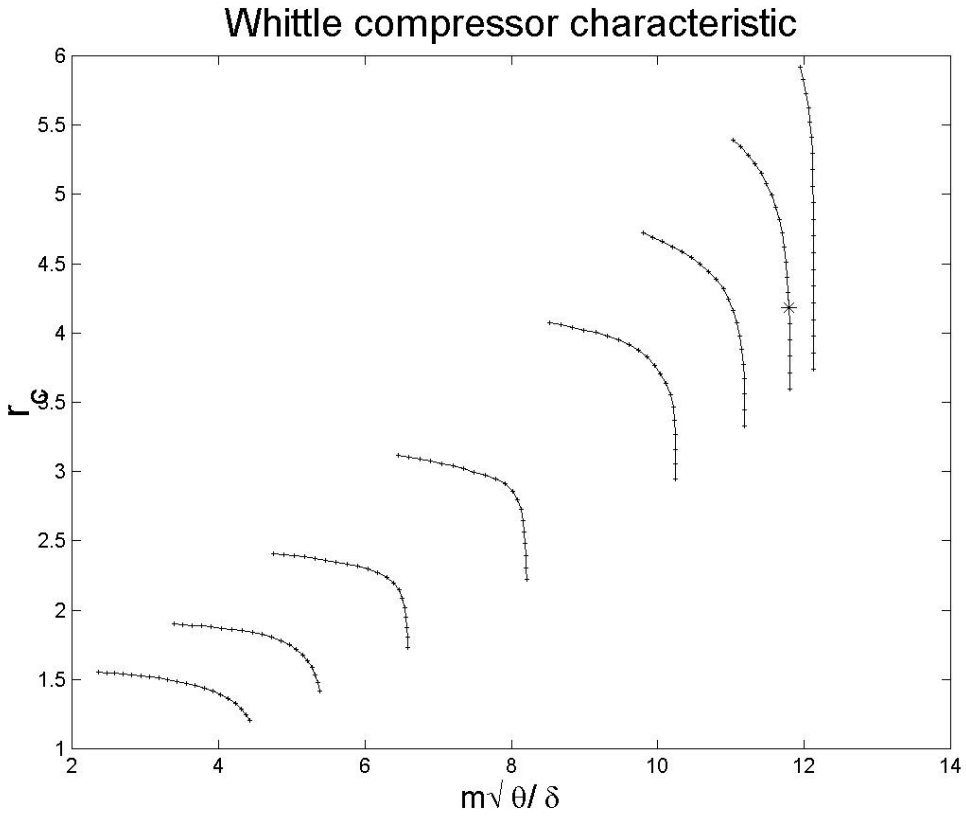

Figure 2: Whittle engine Compressor characteristic.

ratio causing a drop in the efficiency. Another, interesting phenomenon is called surging where the delivery pressure drops very quickly at some point causing the flow to reverse its direction, which may lead to a total breakdown of the engine. But, still this phenomena is complex and still subject to much research.

1.3 The turbine

Turbines are like windmills; they are similar to compressors in terms of design. We could get a turbine by putting a mirror along the compressor, although turbines need less stages to keep the compressor turning. Axial turbines consist of a series of stages, each stage consisting of a row of stators and a row of rotors. The main goal of the turbine is to transform the work from the working fluid into the turbine rotor in order to keep the compressor running when linked together by the shaft.

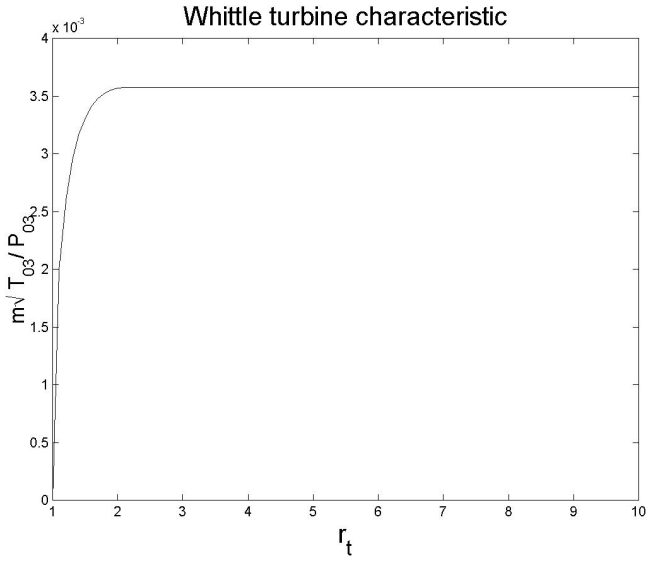

Figure 3: Turbine characteristic.

Turbines should be designed so that they stand high temperatures because the inlet temperature at the turbine is the outlet of the combustion chamber. In our case the temperature is 1050 Kelvin. The design of a turbine for a turbojet engine is limited by many factors such as the annulus area, number of blades, height of blades and the overall weight of the turbine. The performance of the turbine is frequently expressed by plotting $m\sqrt{T_{03}}/p_{03}$ (corrected mass flow) against pressure ratio r_t as shown in Figure 3.

1.4 Combustion chamber

The design of the Combustion chamber is a complex task that involves many disciplines. The combustion chamber could be divided into three main parts. The first part, or the primary zone is where the air is mixed with fuel in order to obtain high temperature for a quick combustion. The second part is where the combustion takes place. In the third zone additional air is sucked in the combustion chamber in order to decrease the temperature.

Combustion chambers are divided into many cans so that they suit the centrifugal compressors design. Another way is to use tubes, or tubo-annular system where individual flames are spaced around an annular casing. The performance of the combustion chamber is related to many factors. We could list some important ones; firstly the temperature after combustion should be limited so that turbine does not melt. Secondly, the combustion should be held at a region where the air velocity is between 30 and 60 m/s. Thirdly,

the emission of oxide carbon and NO_x should be kept low. Finally smoke should be avoided at the exhaust because it contains the carbon deposits which can erode the blades and block the cooling air passages.

1.5 The nozzle

The nozzle has the simplest shape in the turbojet engine. In general it is simply a rounded tube. However, there exist many types of nozzles for different types of engines. The most common nozzle is the convergent nozzle. The shape of the nozzle might be simple but it has also a major part of the engine. The main task of the nozzle is to expand the exhaust gas into the ambient conditions. This process results in a thrust that gives the plane a forward speed. The nozzle operates on two conditions, a choking and a non-choking condition. The later one is where pressure ratio is below the critical value, and the first condition is where the pressure ratio is higher than the critical value. These characteristics of the nozzle are related to the pressure ratio p_{04}/p_{05} and $\frac{m\sqrt{T_{04}}}{p_{04}}$. The critical pressure ratio is calculated using equation 1, where η_j is the isentropic efficiency and γ is the gas constant:

$$\frac{p_{04}}{p_c} = 1/(1 - \frac{1}{\eta_j})\frac{\gamma - 1}{\gamma + 1}))^{\gamma/\gamma - 1}. \tag{1}$$

2 Design of the jet engine

The design of gas turbine components is a complex and time consuming engineering task that involves meeting several design objectives and constraints. This task is usually addressed in an iterative process, which is very costly in terms of computation time. Advancements in the field of computation allowed the design of high performance turbojet engines. In this section, we will state some methods and concepts used to find suitable design parameters for a jet engine.

2.1 The design process

Usually the design process is decomposed into smaller tasks in order to make it less complex. The smaller tasks have different features than the main one and thus we may loose sight of the main problem. Another difficulty with the design problem is the complexity of function and structure. In other words, the complexity of the design arises when all the components are assembled in one structure. This design process is an iterative procedure. In our case the different components of the jet engine operate at limited conditions when

they are assembled. There are many classifications of design processes. Tong [?] has made the following classification:

Routine design: We know in advance everything about the design process, including the knowledge needed. The solution however may not be completely known in advance. Due to repetition of similar problem-solving situations, more and more design tasks are becoming routine. Since subtask ordering is decided a priori, routine designs are done more efficiently, and possibly with better results.

Non-routine design: Problems where a high level of experience is missing.

Innovative design: Only the knowledge sources are known in advance.

Creative design: Neither the knowledge sources nor the problem-solving strategies are known in advance.

Redesign: Is concerned with changing prior design decisions. This includes modifying an existing design in response to changing requirements, and making changes to decisions already made during the design process. It can be a result of unsatisfied constraints. Suggestions about suitable changes, can be either pre-stored or generated by analyzing the situation.

2.2 Design of the turbo jet engine

It is important to consider the type of the application that will use the engine. In the case of an airplane it is important to specify the required thrust, fuel consumption, and engine weight. In case of industrial gas turbines other physical properties might be more important. In this design procedure we are using the Whittle engine, which is an early turbo jet engine used during the second world war.

2.2.1 Design approach

The most common way of designing a jet engine is described in Saravanamutto's book [?]. It is very similar to the redesign procedure, where we have to iterate in order to find the right parameter values (pressure ratio, rotational speed). The first step of the procedure is called the design point performance. The engine is designed according to some specific parameters. The second step is to find the performance of the engine at different conditions (different rotational speed of compressors, different altitudes). This part is termed the off-design performance. The idea of the off-design performance is to test the whole system (i.e. the entire engine) at different conditions.

2.3 Design point performance

In this step we determine the performance of the engine at specified design parameters. The parameters include compressor pressure ratio, turbine inlet temperature, component efficiency, and pressure losses [?]. We also specify the engine mass (equation 2) and the engine diameter where δT_c is the temperature difference in the compressor.

$$\text{engine mass} = \text{mass flow}(11 + 0.15\delta T_c). \tag{2}$$

2.4 Off-design performance

This part is the most important part since it is where all the components are linked together over a wide range of conditions such as speed and power output. In this step we can also deduce also the power output or the thrust, and the specific fuel consumption. We follow the same method outlined in Saravanamuttoo [?], which could be summarized in the following steps:

- Select a constant speed line in the compressor characteristic and choose any point on the line. $\frac{m\sqrt{T_{01}}}{p_{01}}$, $\frac{p_{02}}{p_{01}}$, η_c, $\frac{N}{\sqrt{T_{01}}}$ are determined.
- The corresponding point on the turbine characteristics is obtained from consideration of compatibility of rotational speed and flow.
- The next step is to design the nozzle using the turbine characteristics and the diameter calculated in the design phase.

Saravanamutto's method could also be divided into two parts. The first part is balancing the gas generator (turbine, compressor, inlet). It yields the right pressure ratio in the turbine. The second part is matching the nozzle with the rest of the engine. It yields the required rotational speed for equilibrium.

2.5 Aerodynamics of the plane

A flying plane is subjected to four forces; the thrust T produced by the engine, the drag D, the lift L, and finally the weight W. The value of the lift is related to a coefficient called the Lift coefficient C_L while the drag is related to the drag coefficient C_D.

$$C_L = \frac{L}{q_\infty S_w} \tag{3}$$

$$C_D = K_1 C_L^2 + K_2 C_L + C_{D0} \tag{4}$$

$$q_\infty = \frac{1}{2}\rho V^2 \tag{5}$$

where S_w is the wing planeform area while K_1, K_2 and C_{D0} (Zero lift drag coefficient) are functions of Mach number that is the plane speed. From equations 4, 5 and 6 we can see that the lift and the drag are related to each other by the lift and the drag coefficients. m is the mass of the plane. We will study the behavior of the engine only in a horizontal flight. In that case the angle of attack α is equal to zero, hence we have the following equations as:

$$\begin{aligned}
\frac{d^2x}{dt^2} &= \frac{(T-D)\cos\alpha}{m} - \frac{L\sin\alpha}{m} = 0, \\
\frac{d^2y}{dt^2} &= \frac{(T-D)\cos\alpha}{m} - \frac{L\cos\alpha}{m} = 0, \\
\mathrm{T} &= \mathrm{D}, \\
\text{weight} &= \mathrm{L}, \\
\frac{dm}{dt} &= -\text{fuel flow}. \\
\text{weight} &= mg.
\end{aligned}$$

The weight of the plan decreases along the mission, which affects the lift and the latter affects the value of the drag in return.

2.6 Implementation

The above procedure is implemented using Matlab. There are two main procedures; the first one balances the gas generator, and the second one balances the overall engine. We follow the procedure outlined in [?] for the gas generator to find the appropriate compressor pressure ratio. The Matlab built in function fminbnd is used in order to minimize the error between T_3/T_1 from flow and work compatibility. The same Matlab function is used for balancing the nozzle with the other parts of the engine. The output of this procedure is the mass flow and fuel flow in addition to the pressure ratios of the compressor and the turbine. The next step is to calculate the thrust generated according to equation 6 where C_a is the speed of the airplane, m is the mass flow and A_5 is the area of the nozzle. If the resulted thrust does not match the required thrust we call the off-design routine again until we get the required thrust.

$$T = m(C_5 - C_a) + (p_5 - p_a)A_5 \tag{6}$$

2.7 Simulation

In order to test the performance of the engine, we created a simulation of a flight mission. In our case we only consider a horizontal steady flight, or

cruise. The design parameters are: the design point pressure ratio in the compressor, the design point temperature at the turbine inlet and the engine diameter. The reason why we only pick these three parameters because they are related to other variables. Thus we have fewer variables, which makes the optimization problem simpler. The engine weight is calculated in the design phase as a function of the mass flow and the temperature rise in the compressor according to equation 2. The mission is divided using time steps. At each time step we calculate the fuel consumed, the distance traveled, and the thrust required to overcome the drag.

3 Multi-objective optimization

Many real world problems have many goals to optimize simultaneously. Often they are conflicting with each other. Consider an example where we have a student room with a small space and a set of objects. We want to put the maximum number of objects inside the room and at the same time keep a maximum free space. This is certainly a multi-objective optimization problem where the space and the number of the objects are conflicting with each other. In this chapter we will discuss the special features of multi-objective optimization. We will also discuss the different methods used in order to solve these kind of problems.

3.1 Formulation

Multi-objective optimization contains a set of objective functions that have to be minimized or maximized at the same time. The objective functions are sometimes in conflict with each other, as when if one objective function decreases the other increases and vice versa. The general mathematical definition of a multi-objective optimization problem is described below, where $g(x)$ represents the inequality constraints while $h(x)$ represents the equality constraints. The set X represents the decision variables set, while the set Y represents the objective function space.

$$\begin{aligned} \min \ / \ \max y &= (f_1(x), f_2(x), f_3(x), f_4(x), ..., f_n(x)), \\ \text{s.t } g(x) &= (g_1(x), g_2(x), ..., g_m(x))^T \leq 0^m, \\ h(x) &= (h_1(x), h_2(x), ..., h_l(x)) = 0^l, \\ x &= (x_1, x_2, ..., x_k)^T \in X, \\ y &= (y_1, y_2, ..., y_n)^T \in Y. \end{aligned}$$

The usual optimization methods are unable to find a solution for such categories of problems. In single objective optimization problems the feasible

set is an ordered set. **Definition** : Let us suppose an objective function space Y spanned by the objective function f. For given x_1, x_2 we have either $f(x_1) \leq f(x_2)$ or $f(x_2) \leq f(x_1)$ thus the objective space is an ordered set.

In the case of multi-objective optimization, the objective function space is not an ordered set. There is a need for a new way that evaluates all the objectives at once and indicates the quality of the solution. In other words, we need to order the set of objective functions. One way is to assign to each vector in the objective function space $(f_1(x_i), f_2(x_i), ..., f_n(x_i))$ a fitness value or rank, that determines the order of this vector in the objective space set. The resulting set is the space spanned by the objective functions. Generally multi-objective optimization algorithms are classified into three categories [?]:

Decision making before search: In this case we know beforehand what objective is the most important to optimize; the algorithm aggregates all the objectives in one objective and includes a weight that reflects the relative importance of each objective. Such methods are also called preference methods.

Search before decision making: This method is the opposite of the previous one. It performs the search without any available preference information. It has certain characteristics that make the sequence of iteration vectors converge to many points instead of only one. The user has to choose among the solution points in order to decide the optimal solution that suits her/his own goals.

Decision making during search: In this method the search is done first without any preference and then the user interferes to decide which direction the search should go at the next iteration. It is a guided search method.

3.2 The Pareto fronts

The Pareto concept was first introduced in the 19th century by the French-Italian economist and sociologist Vilfredo Pareto. He established the concept of optimality based on a multi-criteria objective. The Pareto front is a set that contains decision variables that constitute the best trade off among all the other variables, i.e. decision vectors which cannot be improved in any objective without degradation in other objectives. The best points are called non-dominated points. In general non-dominance could be defined as follows: if $x \in X$ we say that x is non-dominated if and only if for every $y \in X$ we have:

$$\forall i \in \{1, 2, ..., n\} \; f_i(x) < f_i(y). \tag{7}$$

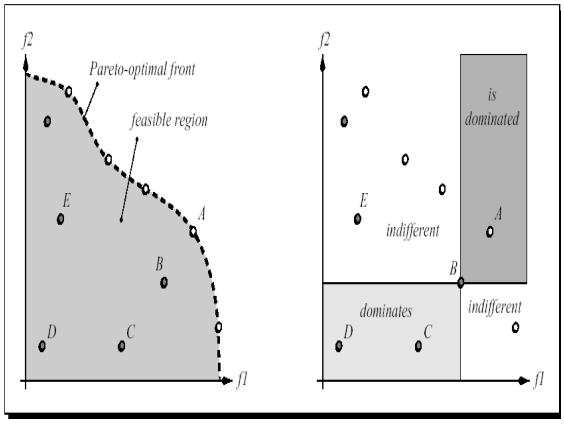

Figure 4: The dominance concept for maximization problem. The point A is not dominated by any points in the shaded area.[3]

This is called the concept of strong non-dominance. The concept of weak dominance is defined similarly as follows:

$$\forall i \in \{1, 2, ..., n\}\ f_i(x) \leq f_i(y) \wedge \exists j \in \{1, 2, ..., n\}\ f_j(x) < f_j(y). \qquad (8)$$

In other words, this definition says that a vector x is Pareto optimal if only and only if there exists no feasible vector of decision variables y, which would decrease some criterion without causing a simultaneous increase in at least one other criterion [?].

3.3 Methods of generating the Pareto front

The question now is how to get the Pareto front. This issues is the subject of many papers. We could also identify two classes of methods: The first one is based on classical algorithms for a single objective, and the second one on the evolution strategies algorithms.

3.4 Algorithms based on classical optimization

Classical algorithms for multi-objective optimization are based on algorithms for a single optimization problem. In general, the multi-objective is transformed into only one objective and then any classical method is used for a single optimization problem. There are many classical methods. We list below some of them.

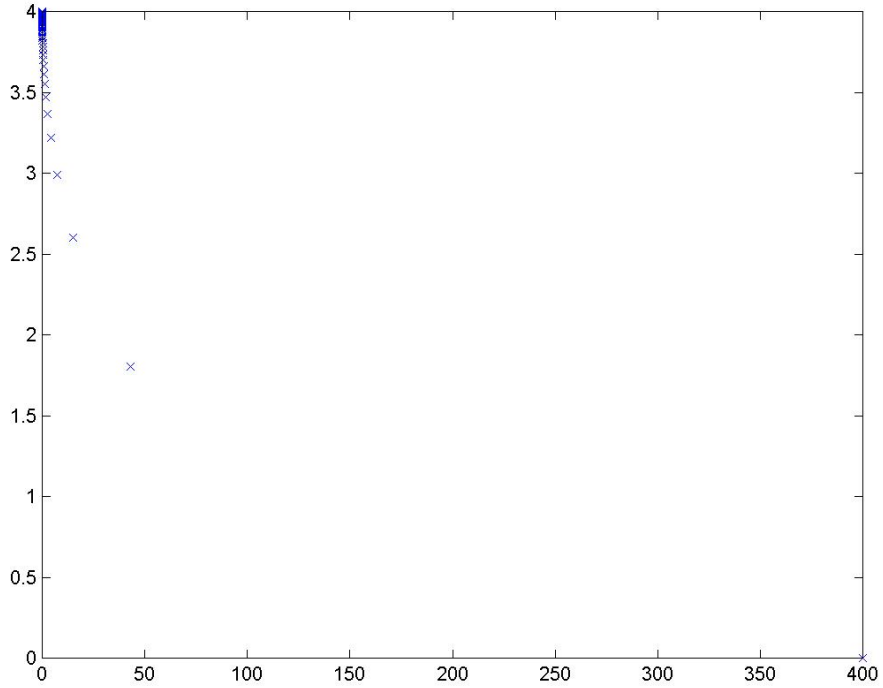

Figure 5: The weighted sum approach with one objective bigger than the other one i.e all the Pareto points are located in one side.

3.4.1 The weighted sum approach

This method creates one objective function from a convex combination of several objective functions; each objective function is multiplied with a weight that represents the importance of the objective function. Then we apply an ordinary optimization algorithm.

$$\text{maximize } y = w^T f(x).$$
$$s.t \sum_{i=1}^{n} w_i = 1.$$
$$w_i \geq 0.$$

We get the entire Pareto front by generating an optimal solution for every possible choice of weight vector. This algorithm might be good in the case of a convex Pareto front, but sometimes it is not reasonable to sum all the

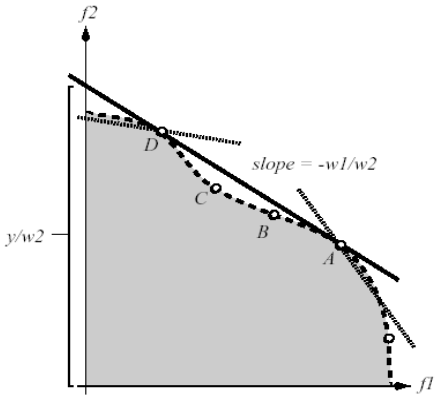

Figure 6: The weighted sum approach[3].

objective functions in one because they may represent different properties and thus the values of the objective functions will differ. For example let us consider a multi-objective minimization problem $f = (f_1, f_2)$ such that $f_1 \ll f_2$. From Figure 5 we notice that all the Pareto points are concentrated in the extremity of the lower valued objective function. The other problem is that the weighted sum approach does not lead to a Pareto front in case of a non-convex Pareto front [?]. Since each of the objective functions is the basis for the objective function space their linear combination must be convex, leading to a convex Pareto front. Geometrically, if we consider the above maximization problem, we could see that the problem could be reformulated as $max f_2(x) = \frac{w_1}{w_2} f_1(x) + \frac{y}{w_2}$. Finding the maximum for the latter problem is equivalent of moving the line with the slope $\frac{w_1}{w_2}$ that pass trough the point $\frac{y}{w_2}$ upward into the extremity of the feasible region. From Figure 7 we could see that the points B and C will never maximize f . If the slope is increased, D achieves a greater value of f (upper dotted line); if the slope is decreased, A has a greater f value than B and D (lower dotted line) [?].

3.4.2 Target vector optimization

The main idea is to achieve a target that is specified beforehand. This can be achieved in two ways; the first one is to minimize the distance to the vector y_g in the objective space min $s(x) = \| f(x) - y_g \|$ (note that x must be feasible) where $\| . \|$ is the Euclidian distance. The second one is goal attainment; this method allows the objectives to be under- or overachieved. The problem is formulated as below, where w represents the weight vector while F^* represents the goal to attain. The set $\Re$ and Ω represents the

objective function set and the decision variable set respectively.

$$\underset{y\in\Re,\, x\in\Omega}{\text{minimize}}\ \gamma.$$
$$\text{s.t}\ \ f_i(x) - w_i\gamma \leq F^*\ i = 1, 2, ..., m.$$

The term $w_i\gamma$ introduces an element of slackness to the problem. The value of γ is minimized so that $F_i(x) - w_i\gamma$ is less than or equal to the target vector of the feasible set. These methods require some knowledge about the goal functions. In addition they are unable to find the Pareto front in the case of a non-convex Pareto front.

3.4.3 ϵ-Constraint Method

This method was designed in order to solve problems having a non-convex Pareto front. The main idea of this method is to optimize one objective and to put all the others in the constraint space by restricting their values to be below a target value μ_m. Sometimes this method is used together with other heuristic methods. We should have accurate values of ϵ_m, otherwise we will be outside the range of the Pareto front Figure 7. We could find the range of the Pareto front by just finding the minimum for each individual function.

$$\begin{aligned}
&\text{minimize} f_\mu(x)\\
&\text{s.t}\, f_m(x) \leq \epsilon_m,\ m = \ 1, 2..., M \text{ and } m \neq \mu, \mu \in 1, 2, ..., M\\
&\quad g_j(x) \geq 0,\ j = 1, 2, ...n;\\
&\quad h_k(x) = 0,\ k = 1, 2, ..., K;\\
&\quad x_i^{(L)} \leq x_i \leq x_i^{(U)},\ i = 1, 2, ..., n.
\end{aligned}$$

3.5 Multi-objective evolution algorithms

Multi-objective optimization using evolution strategies could be classified into two main sets of algorithms. The first one is a non-Pareto based method; as it does not use the concept of dominance directly, it is easy to implement. However, it is unable to produce a complete Pareto front especially when it is non-convex. The second set of methods is termed Pareto based methods. These methods use the concept of non-dominance.

3.6 Simple genetic algorithm

The idea behind genetic algorithms is the theory of evolution by Charles Darwin. He claims that species have to adapt to their changing environment

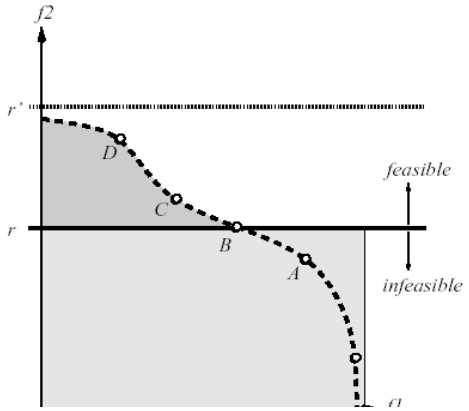

Figure 7: ϵ-Constraint method we notice that the Pareto front is obtained by moving the line f_1=r upward and finding the finding the minimum of f_2[3].

by inquiring knowledge. This knowledge is embodied in their chromosomes. In terms of optimization genetic algorithms are stochastic methods that can be perceived as a search through a potential set of points. These points are seen as individuals (genotypes), and the whole set is called population. There are many ways to represent individuals depending on the problem. One way is through the binary representation the other way is through floating points. The general components of a genetic algorithm are:

-Initialization: Where we have to initialize the first generation.

-Evaluation: Where we have to measure the fitness or the rank for each individual. The fitness, in absence of constraints, is generally just the value of the objective function in a minimization problem. It is the inverse of the objective function in a maximization problem.

-Selection: After initializing the first generation we need to produce the next generation. We do that by using genetic operators. First, we select the parents $Parent_t$ from the actual generation t. There are many selection methods. We could mention here tournament selection, proportionate selection and ranking selection. In the first one two solutions are randomly chosen from $Parent_t$ and then they are compared to each other. The best one is placed in the mating pole. Individuals could compete more than once so that we make sure that we have the best of the individuals in $Parent_t$. Proportionate selection makes an analogy with the roulette game. A roulette wheel is divided according to the average fitness value of the individuals. The bigger this value is the greater portion it takes on the roulette wheel. A randomly chosen number is generated between 0 and 1.

-Crossover: It is performed after selection. Selected individuals mate

to produce the next generation or the offspring. There are many types of crossover, the most common ones are single point crossover, SBX crossover.

-Mutation: The final stage of a GA is mutation. It changes the individuals slightly. When mutation is used without crossover, the algorithm is called an evolution strategy.

3.7 Techniques used in MOEA

Multi-objective evolution algorithms or genetic algorithms are just an extension of the simple GAs. GAs converge to a single optimum, while MOEAs produce several non-dominated solutions (i.e. vectors on the Pareto front). MOEAs extend the simple EAs by improving the selection of the individuals and keeping a more diverse population. The selection in a simple EA is based on only one objective. On the other hand, so we consider all the objectives in the selection in case of MOEA. The most common way is based on the Pareto dominance concept. The first step is to assign a rank of one to the non-dominated individuals and then find the individuals that are dominated only by individuals in the previous non-dominated set and assign the rank of two to the second layer. We obtain several layers. Individuals in the same layer have the same rank.

The diversity among individuals can be achieved by applying many techniques, which can be divided into niching and non-niching techniques. The main idea behind the niching techniques is to allow only parents with a certain distance to mate. Cavichio [1] suggested a non-niching method of preserving diversity by applying a preselecting operator: When an offspring is created its fitness is calculated and then compared with that of its parents. It is replaced with the worst parental fitness, thus in this way we allow multiple important solutions into the population.

Goldberg and Richardson [1] suggested another evolutionary niching technique: fitness sharing. The main idea behind this technique is that the fitness value of each individual is computed with respect to each point in the neighborhood interval. If a point is located in a crowded region (there are many vectors located in a small interval in the objective function space) it has a low fitness value. Among the parameter that this method uses is the niche shared σ_{share}. It usually represents the radius of the neighborhood. If we consider a problem where we want to get q solutions in a multi-objective problem we assume the $q \ll M$ where M is the population size. The sharing function is computed according to:

$$sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha}, & \text{if } d \leq \sigma_{share}, \\ 0, & \text{otherwise.} \end{cases}$$

$$d_{ij} = \sqrt{\sum_{k=1}^{|front_1|} (\frac{x_k^j - x_k^i}{x_k^max - x_k^min})^2}. \quad (9)$$

The shared fitness is calculated by the below formula where $d(i,j)$ represents the Euclidian distance between two individuals in the objective space. The indices i and j represent arbitrary individuals:

$$s_i = \frac{f_i}{\sum_j sh(d(i,j))}. \quad (10)$$

Another niching technique was proposed by DeJong [1]. He used a crowding method within the EA. This model tries to balance the population in the objective space. Basically, only a proportion of the population is permitted to reproduce. The resulting offsprings replace individuals that are similar to them.

3.8 EA algorithms used in multi-objective optimization

3.8.1 Vector evaluated GA

The first multi-objective EA was suggested by David Schaffer in 1984. It is called the Vector Evaluated Genetic Algorithm (VEGA). As the name suggests, this algorithm evaluates an objective function vector instead of a scalar [1]. In fact, it is just an extension of a simple genetic algorithm, the only difference lies in the way the selection operator is constructed. At every generation the population is randomly divided into M equal subpopulations. Each subpopulation is assigned a fitness value based on one objective function. Then the selection operator is applied and the best solutions are placed in a mating pool ready for crossover. Thus in this way each subpopulation emphasizes one objective function. We could summarize the whole algorithm as follows:

Step 1: Set an objective counter and define $q = n/M$.

Step 2: For all the solutions $j = 1 + (i-1) * q$, assign fitness as the value of the objective function f_i.

Step 3: Perform proportionate selection on all q solutions.

Step 4: If $i = M$, go to Step 5, otherwise increment i by one and go to Step 2.

Step 5: Combine all mating pools, perform crossover and mutation on the population P to create a new set of individuals.

The main strength of this algorithm is its simplicity. VEGA however has a tendency to find extreme values for each objective function; the whole pop-

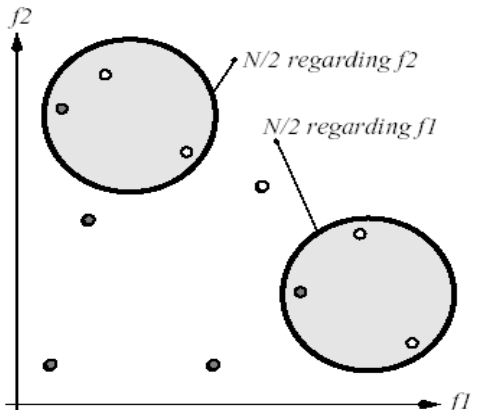

Figure 8: Vector Evaluated Genetic Algorithm procedure[3].

ulation may converge to the extreme points of each objective, which affects the diversity of the solutions.

3.8.2 Non-dominated sorted genetic algorithm

This method was implemented by Deb and Srinvas in 1994 [1]. It is based on the non-dominated sorting presented by Goldberg. This algorithm sorts the population according to their level of dominance. It produces many layers corresponding to the level of dominance. The individuals in the first front are the best, so they are assigned the rank of one. The individuals that are in the first layer are assigned a fitness of the population size M, and for the second layer we assign $M - 1$ and so on. To preserve diversity in NSGA, the fitness of individuals that are in a crowded region is degraded by lowering their fitness value.

First the distance between the decision variables is calculated, and then the sharing function is calculated from equation 10. Any solution that has a distance greater than σ_{share} contribute nothing to the sharing function. The second step is to calculate the shared fitness, which will lead to the degradation of the fitness of the solution in the crowded regions. The selection operator is based on the roulette wheel selection method.

4 NSGA-II

The Non-dominated sorting II algorithm is a multi-objective evolution strategy. It was developed by Deb Kalyanmory and his student in 2002 [5]. NSGA-II is an improvement of the NSGA algorithm created by the same author.

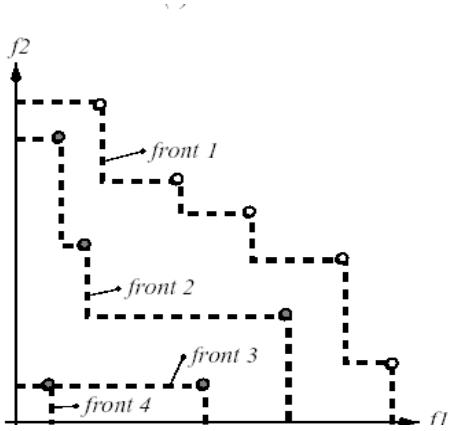

Figure 9: NSGA result in dividing the population into layers[3].

The latest method was subjected to much criticism such as the lack of elitism and the long computation time. However, Deb claims that this method is able to find much better spread of solutions and better convergence near the true Pareto front [5]. He claims that the strength of this method relies on the diverse solution that it provides. In this chapter we will describe the different parts of this MOEA.

4.1 Elitist non-dominated sorting genetic algorithm

In addition to the general components of the genetic algorithm there are two additional functions of this algorithm. The first one is Fast Non-dominated Sorting that ranks individuals. The second one is the crowding distance function [5]; which is used to calculate the euclidian distance between individuals in a specific rank.

4.2 Fast non-dominated Sorting Approach

This part of the algorithm ranks the population into different levels or layers, so that the first layer contains the non-dominated individuals and the second layer is dominated only by vectors in the first layer and so on (see Figure 9). The individuals in the population are equipped with two variables: The set of dominance S_p (the set of point dominated by the individual p) and n_p (the number of individuals that dominate p). So if a point p is not dominated by any solution, its count n_p will be equal to zero, and thus it will be ranked in the first layer. The count can not be greater than $N-1$ where N is the number of individuals. The procedure is presented below, the input is the whole population and the output is a set of layers or fronts.

```
for each p in P
    S_p := empty;
    n_p=0;
    for each q in P
        if p dominates q
            Add q to S_p;
        else
            increment the domination counter of p;
        if n_p:=0
            rank_p:=1;
            add p to F_1;
        i = 1;
while F_i is not empty
    Create an empty set Q;
    for each p in F_i
        for each q in S_p
            n_q := n_q - 1;
            if n_q := 0;
                q_rank := i+1;
                put q in Q;
        i = i + 1;
        F_i = Q;
```

Table 1: The fast non-dominated sorting algorithm.

4.3 Non-crowding distance assignment

In the traditional NSGA, diversity is maintained through the sharing function that depends strongly on σ_{share}, which is set by the user. In NSGA-II the sharing function approach is replaced by the crowding distance. The method estimates the density of points around a specific solution i in every layer. Initially the solutions in the fronts are sorted according to their objective function values, then the perimeter of the rectangle formed by the nearest neighbor is calculated. Finally $i_{distance}$ the crowding distance in each front is calculated. The extreme individuals (lower and higher objective functions values) are given the highest crowding distance i.e.infinity.

The crowding distance is used by the crowded-comparison operator, that is defined by $\leq_n$. It helps the selection operator in building a well spread and diverse population. Initially two individuals are compared with respect to their rank. If they belong to the same front, their crowding distance is compared. The solution that has bigger crowding distance is then chosen.

the $\leq_n$ is defined by:

$$j \leq_n i \text{ if}(i_{rank} \leq j_{rank}) \text{ or if } ((i_{rank} = j_{rank}) \text{ and}(j_{distance} \leq i_{distance})).$$

4.4 The Algorithm

As any GA algorithm we first generate a random initial population P_0 and assign each individual a fitness based on its level of non-dominance. The individuals are selected based on binary selection, which compares two individuals based on their level of dominance and crowding distance. The first generation Q_0 is created by mating the individuals. The procedure for the next generations differs; we usually compare the new generation with the previous one to identify the best individuals.

The Fast Non-dominated Sorting is applied in order to sort out the fronts according to the non-dominance level. If we want to construct N parents for the next generation we take all the individuals that are in the first level of non-dominance. If we still need more individuals we get them from the other front in order of non-dominance. In that way we make sure that the non-dominated solution remains in all the future generations.

---

combine parent P_t and offspring Q_t int R_t
F:=fast-non-dominated-sort(R_t);
until the parent population is filled
    crowding-distance-assignment(F_i);
    include F_i into the parent of next generation P_t+1;
    i:=i+1;
    Sort(F_i,$\leq_n$) choose the first (N-P_{t+1}) element of F_i;
    Q_{t+1}:=make-new-population(P_{t+1}); t:=t+1; go to next generation

---

Table 2: The main loop.

4.5 SBX-crossover

We chose the bounded Simulated Binary Crossover (SBX), which was proposed by Deb and Agrawal. It contains a certain degree of mutation in the crossover phase. The SBX operator introduces a factor β called the non-dimensional spread factor. If β is less than one the distance between the children will be bigger than the distance between the parents. In this case the crossover is contracting. On the other hand, if β is bigger than one,

the distance between the children is smaller than the distance between the parents. For more details the reader could refer to [?] or [?].

5 The Strength Pareto Evolutionary Algorithm

The SPEA algorithm was first proposed in 1998 by Eckart Zizler [?]. It has the following property:

-It stores the Pareto-optimal solutions found so far externally.

-It uses the concept of Pareto dominance in order to assign scalar fitness values to individuals.

-It performs clustering to reduce the number of non-dominated solutions stored without destroying the characteristics of the Pareto front.

-The fitness of the individuals depends on the solution stored in the external Pareto front.

5.1 The algorithm

The basic idea of the algorithm is to conserve all the Pareto solutions in a set called the external Pareto front so that we do not loose any Pareto point. On the other hand, we put restrictions on the size of the Pareto set. In addition the external Pareto solution serves to evaluate the fitness of the individuals outside the external Pareto set.

The fitness of an external Pareto vector is computed as the number of individuals that it dominates divided by the total individuals in the current population. The fitness of an external Pareto front is also called the strength. Diversity within the Pareto front is obtained by distributing the population uniformly in the Pareto front so that each Pareto point covers an equal amount of points [?]. The fitness of an individual outside the external Pareto set is calculated by summing up the fitness of the external Pareto vectors that cover the individual. We add one to the resulting value so that it will be always higher than the fitness in any individual in the external Pareto set. The equation used to calculate the fitness of external individuals and the other individuals is illustrated in below. S_i represents the strength while F_i represents the fitness.

$$S_i = \frac{n_i}{N+1}. \tag{11}$$

$$F_i = 1 + \sum_{i \in Pareto} S_i. \tag{12}$$

5.2 Pareto clustering

In this part of the algorithm the size of the Pareto front is reduced if it is above a certain limit. The reduction of the pareto front is necessary due to the following reasons [?]:

-Presenting all Pareto solutions found is useless.

-In the case of continuous Pareto fronts, it is not desirable to keep all generated points.

-The Pareto Front could reach the population size resulting in a reduction of the selection pressure causing to slow down of the optimization process.

-A huge Pareto front could lead to an unbalanced distribution in the population for the next generation, since the fitness of each individual depends on the Pareto point that covers it. A huge Pareto set could lead to an unbalanced distribution.

The goal of clustering is to find the best representative set which maintains the characteristics of the original set in a way that encourages diversity of the Pareto front. The general idea is to create a cluster for each point. Then a joining mechanism is used to join the clusters that have the minimum distance(we use the Euclidian distance). The scaling is done by using the maximum and minimum values for each objective in each generation. The points that are close to each other are joined. We keep doing this until the number of clusters is equal to the limit of the external Pareto set vectors. The last part of this algorithm chooses one point from each cluster; it chooses the closest point to the centroid of the cluster. We finally have the Pareto front reduced to the desired size.

$$d(1,2) = \frac{1}{\mid C_1 \mid\mid C_2 \mid} \sum_{i \in C_1, j \in C_2} d(i,j). \tag{13}$$

5.3 Main loop

The first step in SPEA is to initialize a population of size N and then find the non-dominated solutions and put them in an external Pareto front. After that, we check if the Pareto front is within the limited size by applying the second main part that is calling the Pareto clustering function. Then we calculate the fitness according to the strength of the individuals in the external Pareto set. The selection operator is used to select the mating parent in the crossover. For the crossover we used the SBX crossover again.

6 Results and Conclusion

In this section, we discuss and compare the results that we get from NSGA-II, SPEA and the weighted sum algorithms. We begin by applying our algorithms to a simple test problem that is often used in the multi-optimization literature.

6.1 Schaffer´s Test problem

A very simple test function for multi-objective optimizers is called Schaffer problem (SCH) [3]. The multi-objective optimization problem is presented as follows:

$$\begin{aligned} &\min \{x^2, (x-2)^2\}, \\ &\text{s.t } x \in [-100, 100] \ . \end{aligned}$$

We have applied three algorithms to the test problem. The running time of the three methods differ. The fastest method is actually the weighted sum method, followed by NSGA-II and finally SPEA is the slowest. The weighted sum is fastest because the algorithm does not need any fitness evaluation or any ranking mechanism that both NSGA-II and SPEA use. In addition, both objective functions are differentiable, so it is an easy task to find the minimum using any ordinary optimization method. For the test problem, both NSGA-II and SPEA give nearly the same Pareto front. The figures below show that two MOEA with the same crossover operator may lead to the same result. Indeed, the crossover is the most important operator in a GA or MOEA.

6.2 Engine optimization problem

Finally, it is time for take off. Here, we consider two objective functions that are in conflict with each other (engine weight and the fuel consumption). The two objectives are conflicting because if we have a smaller engine weight, we need more fuel in order to produce a specific thrust, while if we have a heavy engine it uses less fuel in order to produce the same thrust.

In order to solve this problem we will use the previous three algorithms. We should mention that this type of optimization is called simulation optimization because both simulation and optimization are integrated trying to find the best set of parameters in the design. Our two objectives for the

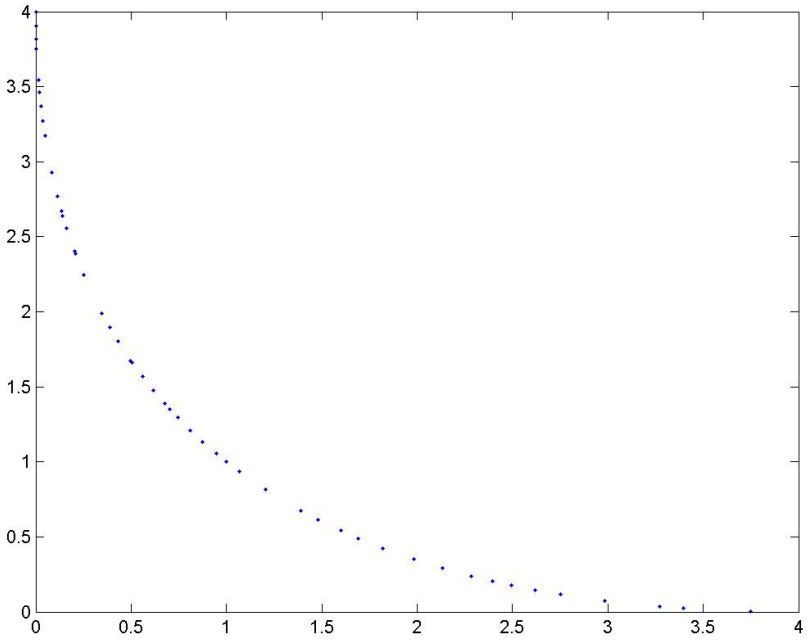

Figure 10: Pareto front resulted by applying NSGA-II.

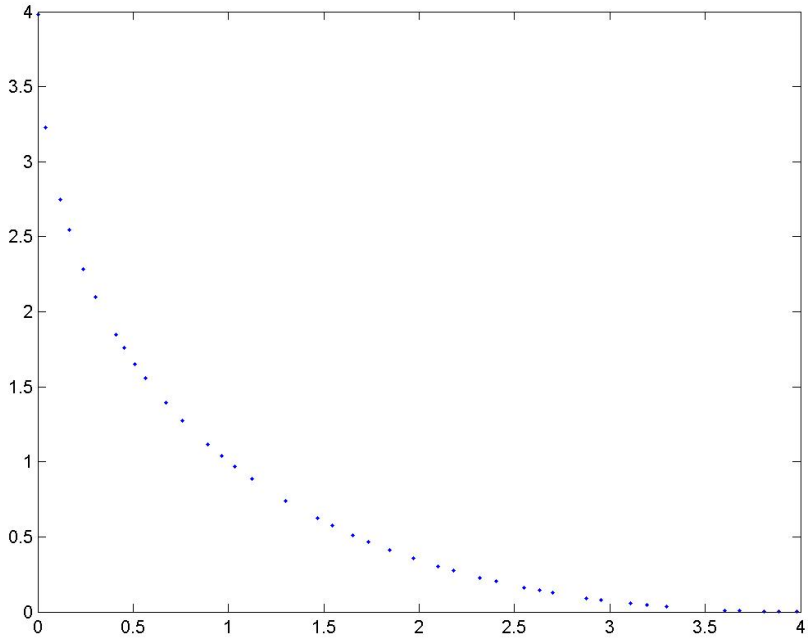

Figure 11: Pareto front resulted by applying Spea.

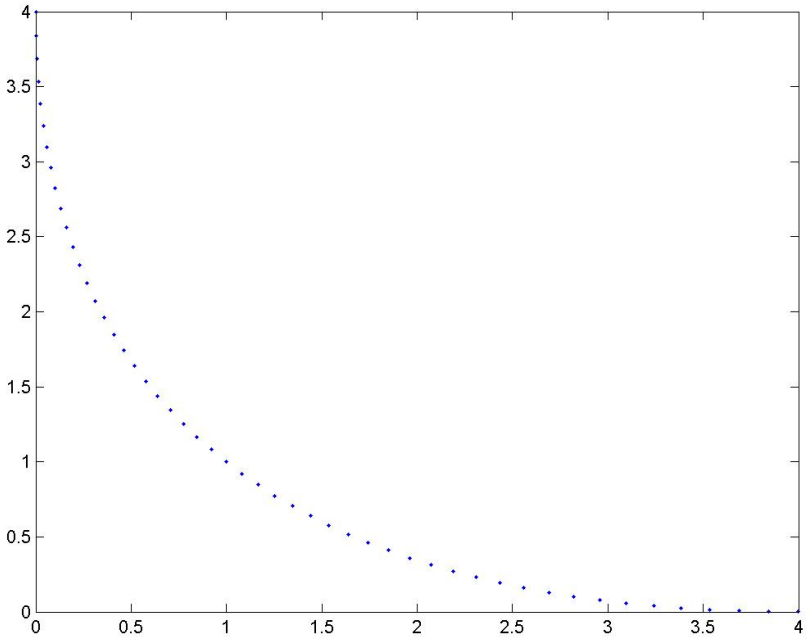

Figure 12: Pareto front using Weighted sum approch.

problem are:

$$\begin{aligned} min f_1(r_c, T_t, d_{engine}) &= \int_0^t \text{fuelflow}(r_c, T_t, d_{engine}). \\ min f_2(r_c, T_t, d_{engine}) &= \text{mass flow}(11 + 0.15\delta T_c). \\ \text{s.t } r_c &\in [3, 6]. \\ T_t &\in [1042, 1046]. \\ d_{engine} &\in [0.5, 1]. \end{aligned}$$

where r_c is the pressure ratio in the compressor, T_t is the temperature at the inlet of the turbine and d_{engine} is the diameter of the engine. In fact, we should not limit the range of T_t to only $[1042, 1046]$. The problem was that the engine model crashes if the temperature is less than 1042. But, sometimes with a high engine diameter it works. In consequence we may miss some individuals.

6.3 Results and Conclusion

Before commenting on every method,we would like to draw some conclusions concerning the turbojet engine part of this thesis. First, the simulation model is an integral part of the optimization algorithm. Second, running a simulation in Matlab made the running time for the simulation plus the optimization algorithm very long. The total run time needed is three days. As described earlier the simulation consists of an engine model that runs during the flight. The engine model was the slowest part in the simulation. Usually when several subsystems are integrated in the modelling, we should have effective tools and methods. The department of Thermo and Fluids Dynamics at Chalmers uses GESTPAN, a generalized system for the design, steady-state and transient simulation of gas turbines systems. In fact, the most complex part in this thesis is to implement the engine model since we have to consider many other subsystems. It would have been easy if we could use to GESTPAN.

The population for both MOEA's contains 50 individuals. NSGA-II is computed for 100 generations while SPEA is computed for 133 generation. The same crossover operator SBX is applied for both SPEA and NSGA-II. It has crossover probability 0.9. Mutation is induced in the crossover [?]. Finally, we had used a weight factor equal to 0.05. If we suppose that W is the weight vector, each time we find a Pareto solution we increment W_1 by 0.05 and decrement W_2 by 0.05.

The range of the Pareto front resulted from the weighted sum algorithm is greater than the range of NSGA-II and SPEA. The maximum engine weight

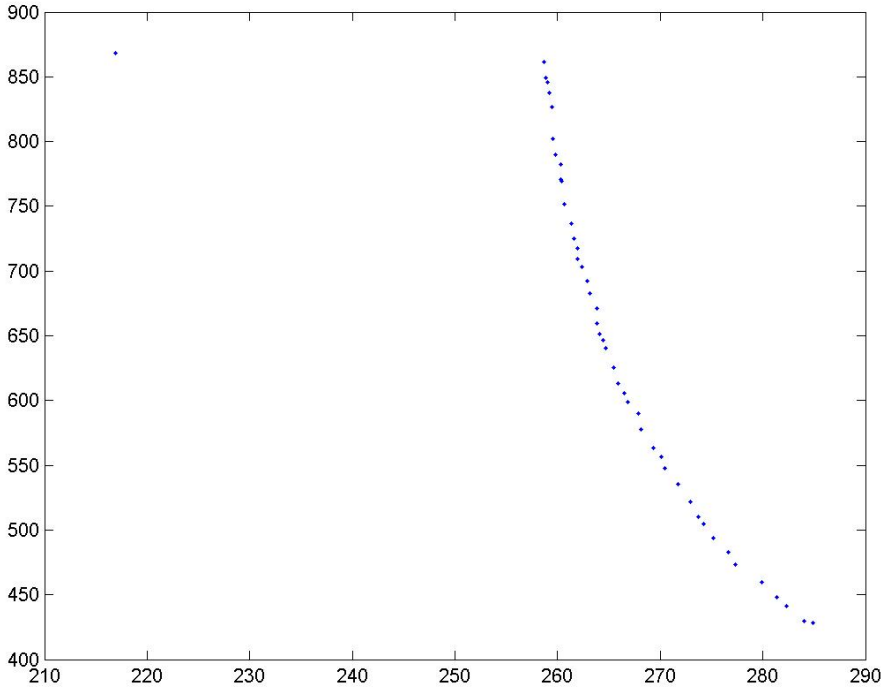

Figure 13: Pareto front resulted from NSGAII algorithm.

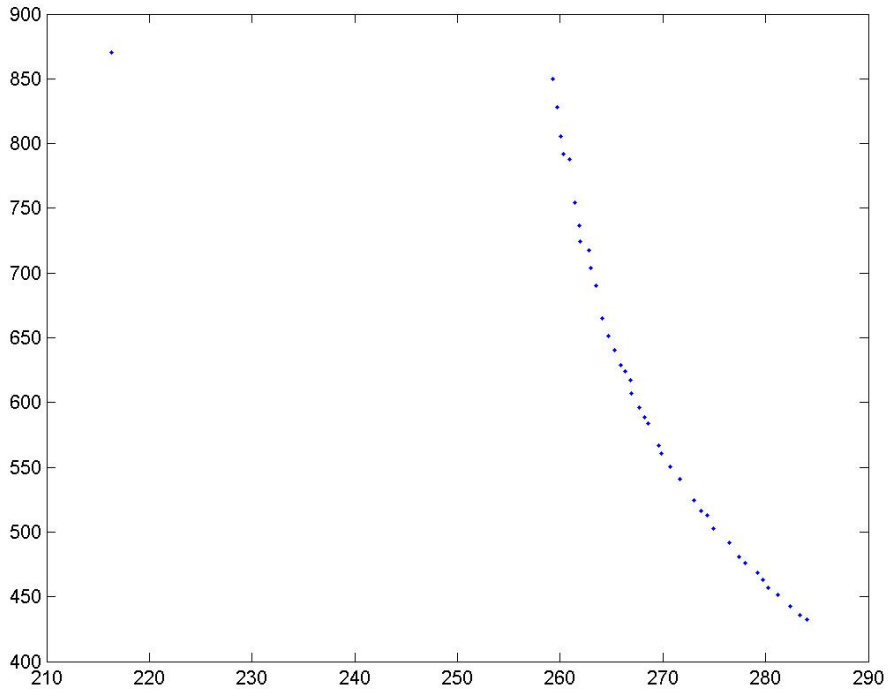

Figure 14: Pareto front resulted applying SPEA algorithm.

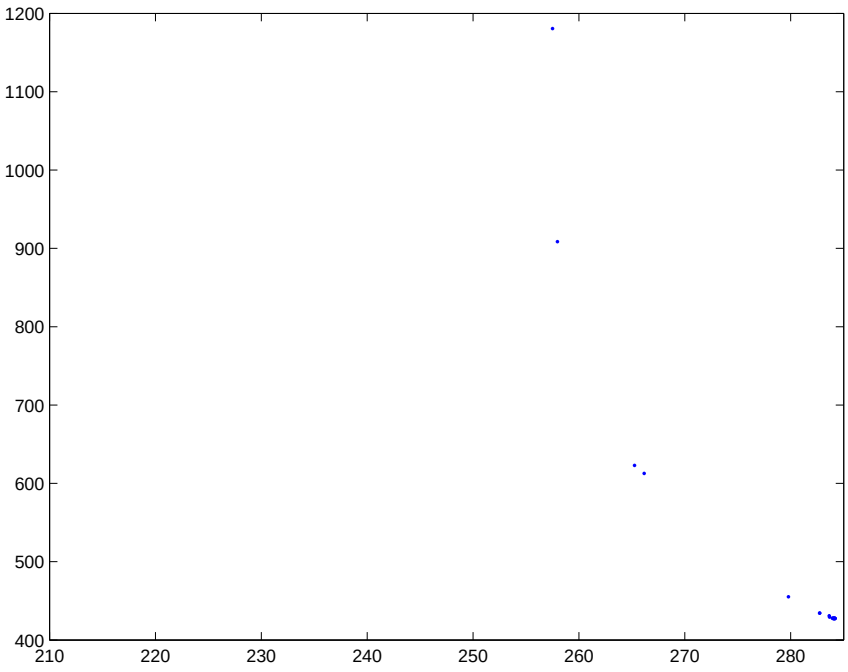

Figure 15: Pareto front resulted from the weighted sum approach

is 1180 Kg and the fuel consumption is 285 Kg. In Both Spea, and NSGA-II the range is 850 Kg for the engine weight and approximately 284 Kg in the fuel consumption. Usually, MOEAs need many generations in order for individuals to converge to the real Pareto front. I did not have time to run my MOEA more than 150 generations. On the other hand from the figures above we could see that NSGA-II and SPEA give a better spread of solutions along the Pareto front than the weighted sum approach. We could conclude that MOEA produce a front where the solution are distributed more evenly in one run.

In the weighted sum approach (Figure 15) the two extreme solutions and many of the solutions are concentrated near the maximum fuel consumption. The reason is that the fuel consumption is smaller than the engine weight, so minimizing the weighted sum of the two functions yields a solution with more fuel and less engine weight. So many points will converge in one extreme side that has the smallest objective value. So we will not have a well spread Pareto front. In fact the outcome of the weighted sum algorithm depends very much on how the objectives are aggregated. So for this case we could use a scaled weighted sum. The idea is to calculate the extreme point first, then divide the weights by the extreme values for each objective before optimizing the sum. This may lead to a better spread but on the other hand this requires extra computation time.

The other observation is that NSGA-II and SPEA yield nearly the same Pareto front. Why do we have this similarity? It is natural since we want to get a Pareto front at the end so we must get similar results. I believe that the reason for this similarity is that we used the same crossover operator for both methods. On the other hand the solutions are more spread along the Pareto front in NSGA-II better than SPEA. NSGA-II is considered to have a better diversity mechanism that is based on the crowding distance operator.

The main difference between NSGA-II and SPEA is in the computation time: NSGA-II is a faster method than SPEA. NSGA-II has a total complexity of $O(MN^2)$ for fast non-dominating sorting function, a total of $O(M(2N)log(2N))$ for crowding distance assignment and a total of $O((2N)log(2N)$ for the comparison operator $\leq_n$ where M and N represent the number of objective and population size respectively [2], while the total complexity of SPEA is $O(MN^3)$ [2].

As expected the weighted sum is a faster algorithm than the other two. We used **fmincon** in order to solve the weighted objective function. It took almost 100 function evaluations for each point. This means two generations in MOEA for only a single point. But we only need to get around 20 points which means it takes the same time as 40 generations in a MOEA. On the other hand if we wanted to generate 50 solutions, it will take us the same

time as 100 generations in NSGA-II or SPEA. But still the weighted sum will be faster because it does not use any other function to preserve diversity or to produce a new generations. We could also deduce that the weighted sum approach is less expensive in terms of computation time. In addition, MOEAs need more memory in order to store the data structure of every population.

Now the question to answer is: What is the best method for multi-objective optimization? In general our results and also those of other authors readings suggest using MOEAs in the case of engineering design. It is easier to solve a multi-objective optimization using MOEAs. They are more general for different kind of design problems in engineering, especially if we lack information about design problems. They could handle both continuous and discrete parameters although they need a lot of computation time, memory, and even produce an incomplete Pareto front. But, if we have some knowledge about the objective functions, classical methods are preferred since they generate a smooth and complete Pareto front in less time and require less storage of data. In real life problems we do not have any knowledge about the range of the Pareto front or even about the objective function themselves. In that case we should not hesitate to use MOEAs in order to find the Pareto front or at least aquire knowledge about the problem before doing any further optimization with classical methods.

In this thesis we focus our effort on solving a problem with two conflicting objectives. In other problems one might have more than three objective functions, which may have conflicting relationship. In that case the Pareto front may be difficult to visualize for the decision maker. In this kind of problems we are more concerned with getting one optimal solution rather than many (Pareto front).

In summary, we would say that Genetic algorithms for multi-objective optimization are very useful for design problems. But more research should be carried out into this field especially on the crossover operator, because individuals are generated based on special parameters, so that we reduce the number of generations and computing time. Of course, if we have a well defined problem it is better to use classical methods because they produce complete and smooth Pareto front.

References

[1] Kalyanmoy Deb, “Multi-objective Optimization Using Evolutionary Algorithms," Chichester:Wiley, 2001.

[2] Kaisa Miettinen, "Nonlinear Multiobjective Optimization," Kluwer Academic Publishers, Boston 1999.

[3] E. Zitzler. "Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications", PhD Thesis, Swiss Federal Institute of Technology (ETH) Zurich. TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, December 1999.

[4] H. I. H. Saravanamuttoo, G. F. C. Rogers, H. Cohen, "Gas turbine theory," Prentice Hall, Harlow, 2001.

[5] K. Deb, S. Agrawal, A. Pratap and T. Meyarivan," A fast and elitist multiobjective genetic algorithm: NSGA-II," Technical Report, Indian Institute of Technology, Kanpur Genetic Algorithms Laboratory, India 2001.

[6] E. Zitzler. "An evolutionary algorithm for multiobjective optimization: The strenght Pareto aproach," Swiss Federal Institute of Technology (ETH), TIK-Report 43.May 1998.

[7] C. Tong and D. Sriram, "Artificial intelligence in engineering design," Academic Press, Boston, MA, 1992.

[8] B. Rekiek. "Assembly Line Design (multiple objective grouping genetic algorithm and the balancing of mixed-model hybrid assembly line)," PhD thesis, CAD/CAM Department, Free University of Brussels, Brussels, Belgium, December 2000.

[9] M. Waline, "Engine Optimization for a Low-Signature Reconnaissance UAV," licentiate thesis Chalmers univeristy of technology Göteborg, Sweden, 2003.

[10] I. Das, John E. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," Rice university, Texas, July 1999.