STUDY OF HYBRID STRATEGIES FOR MULTI-OBJECTIVE OPTIMIZATION USING GRADIENT BASED METHODS AND EVOLUTIONARY ALGORITHMS

by

Diego Fernando Páez Bautista

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Mechanical Engineering

Charlotte

2013

Approved by:

_____
Dr. Howie Fang

_____
Dr. Russell Keanini

_____
Dr. Nilabh Srivastava

_____
Dr. David Young

_____
Dr. David C. Weggel

©2013
Diego Fernando Páez Bautista
ALL RIGHTS RESERVED

ABSTRACT

DIEGO FERNANDO PAEZ BAUTISTA. Study of hybrid strategies for multi-objective optimization using gradient-based methods and evolutionary algorithms. (Under the direction of DR. HOWIE FANG).

Most of the optimization problems encountered in engineering have conflicting objectives. In order to solve these problems, genetic algorithms (GAs) and gradient-based methods are widely used. GAs are relatively easy to implement, because these algorithms only require first-order information of the objectives and constraints. On the other hand, GAs do not have a standard termination condition and therefore they may not converge to the exact solutions. Gradient-based methods, on the other hand, are based on first- and higher-order information of the objectives and constraints. These algorithms converge faster to the exact solutions in solving single-objective optimization problems, but are inefficient for multi-objective optimization problems (MOOPs) and unable to solve those with non-convex objective spaces.

The work in this dissertation focuses on developing a hybrid strategy for solving MOOPs based on feasible sequential quadratic programming (FSQP) and nondominated sorting genetic algorithm II (NSGA-II). The hybrid algorithms developed in this dissertation are tested using benchmark problems and evaluated based on solution distribution, solution accuracy, and execution time. Based on these performance factors, the best hybrid strategy is determined and found to be generally efficient with good solution distributions in most of the cases studied. The best hybrid algorithm is applied to the design of a crushing tube and is shown to have relatively well-distributed solutions and good efficiency compared to solutions obtained by NSGA-II and FSQP alone.

TABLE OF CONTENTS

LIST OF ABREVIATIONS

BFGS	Boyden-Fletcher-Goldfarb-Shanno
CSD	Constrained steepest descent
DFP	Davidson-Fletcher-Powell
FSQP	Feasible sequential quadratic programming
GA	Genetic algorithm
MOGA	Multi-objective genetic algorithm
MOOP	Multi-objective optimization problem
NSGA	Nondominated sequential genetic algorithm
NCGA	Neighborhood cultivation genetic algorithm
PF	Piling Factor
QP	Quadratic programming
RMSE	Root mean square error
SQP	Sequential quadratic programming

CHAPTER 1: INTRODUCTION

Optimization is the process of obtaining the best elements from a set of available alternatives (called design variables) to achieve the optimal, i.e., minimum or maximum, value of an objective function while satisfying some constraints. In mathematics, an optimization problem is typically solved by systematically choosing the variables' values from the allowed ranges in order to minimize or maximize the objective function. A large number of methods have been developed in the past and they can be categorized into two major types: direct search methods based on gradient information such as the conjugate gradient method, and heuristic methods such as genetic algorithms (GAs).

Unlike conventional designs that rely heavily on the designer's experience in order to make design changes, an optimum design process is mathematically formulated and involves explicit identification of design variables within the parameter ranges of the problem in order to obtain the optimum values of the objectives while satisfying the constraints. This formulation process, as illustrated in Figure 1-1, also helps the designer to obtain a better understanding of the problem, and ensures that there is an explicit process to follow.

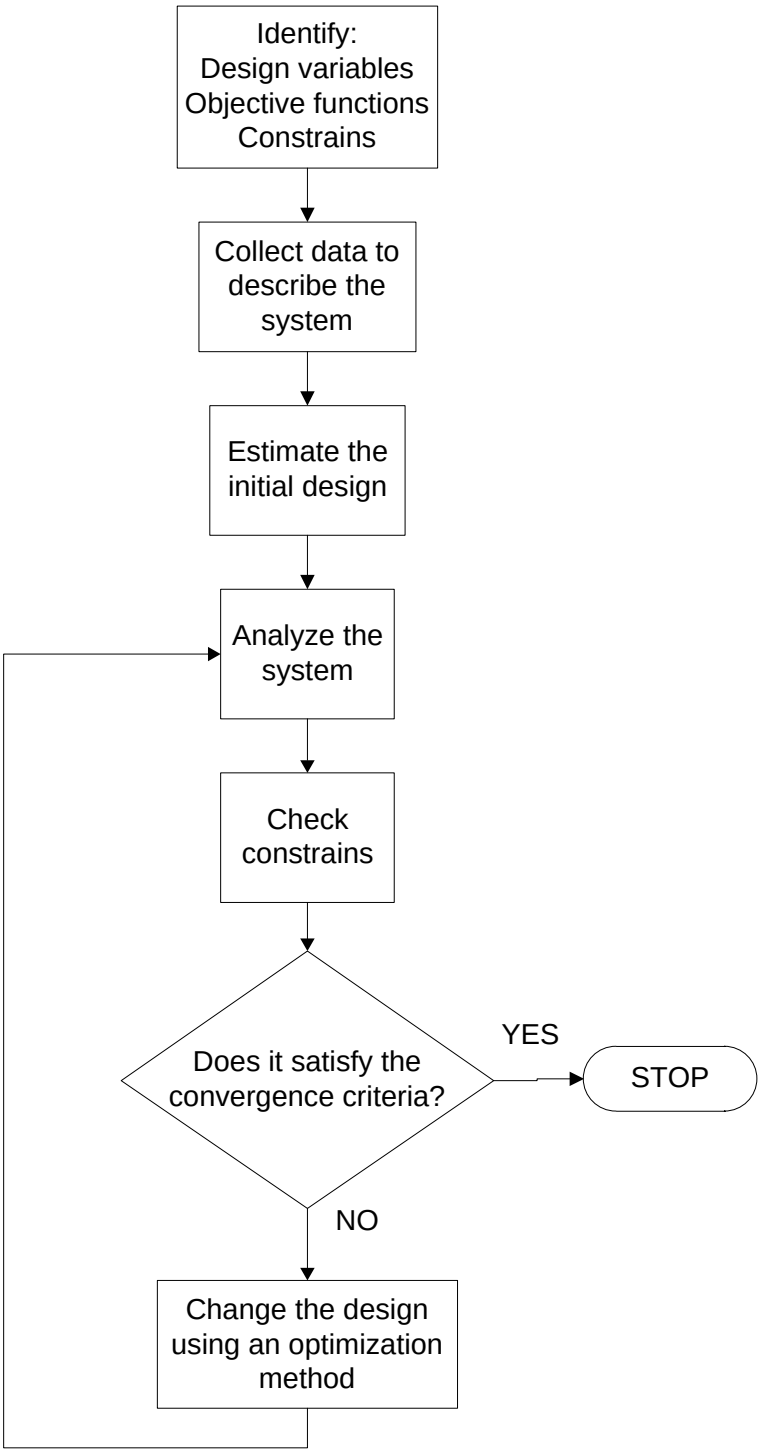

Figure 1.1: Optimum design process

Gradient-based methods can be used to solve continuous, convex optimization problems. The continuous condition ensures the existence of gradients at the current point, and the convex condition is related to the existence of the descent direction of the objective function. These methods were developed to solve nonlinear programming problems that cannot be solved using analytical methods, e.g., the Lagrange Multipliers. In gradient-based methods, the search operation is performed iteratively along the descent directions that are determined using the gradient information so as to approach the minimum of the objective function. Gradient-based methods are bounded to the Karush-Khun-Tuker (KKT) necessary and sufficient conditions, which require a convex solution space and therefore guarantees that the solution to the problem is unique.

Gradient-based methods are convenient for solving single-objective optimization problems, with or without constraints. However, there is no direct application to multi-objective optimization problems (MOOPs) that have multiple, conflicting objectives and a set of non-dominant solutions (i.e., the Pareto Front, which means that no single solution has the smallest values for all the objective functions in a minimization problem). In order to solve multi-objective optimization problems using gradient-based methods, one strategy is to convert all but one objective into constraint functions. This approach results in a single solution that may not satisfy the design needs. In addition, setting the proper constraint for the objective functions may not be easily achieved. Another commonly used strategy, the weighted sum formulation (WSF), is to combine all the objectives into a single objective that is the sum of all of the original objectives each multiplied by a weight coefficient. Each WSF can be solved as a single-objective problem that yields a single solution. Multiple solutions can be obtained by varying the

weight coefficients among the objective functions to obtain different WSFs. Some issues were identified with this method such as the inability to obtain solutions on non-convex regions of the Pareto Front (Das et al. 1997; Messac et al. 2000). Another difficulty of the WSF method is that there is no guarantee that any given combination of the weight coefficients will obtain an acceptable solution in the solution space.

Another group of methods used to solve MOOPs are evolutionary methods, which are heuristic searches based on Charles Darwin's theory of evolution. These methods suit MOOPs particularly well in that they can directly obtain a set of nondominated solutions. The main evolutionary methods are evolution strategy (ES), genetic programming (GP), and genetic algorithms (GAs).

Genetic algorithms use operations such as reproduction, mutation and fitness evaluation, the same terms used in the biological scenario. The GAs are capable of solving optimization problems using only the objective and constraint functions without the need for gradient or high-order information. In addition, GAs can be used to solve optimization problems regardless of their convexity or continuity. In a GA, each trial solution is called an individual, which is a set of variables analogous to the chromosome of a human individual. The phenotypes are the actual values of the objective functions to be optimized.

The main operations of a GA are: fitness evaluation, reproduction, and mutation. In fitness evaluation, a scalar value of fitness is given to each of the individuals in the current population, with a greater value representing a better fit of the individual. After assigning fitness values to all of the individuals, the better fit individuals are selected to reproduce individuals for the next generation. Finally, the mutation operation, which is

introduced in order to maintain genetic diversity, is performed to alter one or more values of the chromosome of individuals that account for a small percentage of the new population. This operation, like biological mutation, has a low chance of occurring.

Evolution strategy consists of mutating the individuals of the population and keeping the ones that have the best fitness values. This strategy relies heavily on the mutation operation in order to obtain better solutions which make the algorithm converge slowly. Genetic programming adds a crossover operation to the procedure of the ES. This allows the algorithm to obtain a new set of individuals that are later mutated on a small proportion (like the biological case).

One of the drawbacks of GAs is the slow convergence, as a consequence of using only zero-order information. Another drawback is that the solution accuracy cannot be controlled when approaching to the Pareto front due to oscillations caused by the genetic operations. Additionally, there is no standard termination condition for the algorithm; a maximum number of generations is usually specified as the termination condition, which implies that the algorithm may be over or under iterated and likely makes the final solution not ideal. Furthermore, there is no mathematical proof of the convergence of the GAs, though the effectiveness of GAs has been shown by empirical results and is widely accepted by researchers and engineers.

Hybrid optimization is the combination of various optimization methods in order to take advantage of the strengths of the individual methods. In the past two decades, researchers have developed various hybrid methods in which evolutionary algorithms, gradient-based methods, particle swarm optimization (PSO) and nested partitions are hybridized to create new algorithms.

In the work of Michalewicz (1994), a hybrid algorithm named Genocop II was proposed by combining a generic GA with simulated annealing (SA) and using the GA as a loop inside the SA. This algorithm was tested using nonlinear programming problems with nonlinear constraints and was shown to have satisfactory solutions compared to the true solutions of the benchmark problems used.

Shi et al. (1999) proposed a hybrid method named Hybrid NP/GA algorithm that was based on a GA and Nested Partitions (NP). The NP method divided the solution space into sub-regions and used a random scheme to select a promising sub-region based on the objective values of a point in the region. Using a backtracking rule, a new region that was larger than the initial sub-region was selected to begin a new iteration. The backtracking rule was used to avoid local optima. In the NP/GA method, the GA was used inside the NP routine to obtain an overall fitness of the members of the population in a partitioned region. This algorithm was tested on a product design problem whose objective was to design a single product to maximize the market share of the producer with 10, 20, 30 and 50 attributes. The results of the hybrid algorithm was found to increase the function value by 1%, 2%, 2% and 3%, respectively, compared to a GA. The computational time was increased by ten and eight times for the first two cases, and 44% for the 30-attribute problem. The computational time for the 50-attribute problem was the same as for the hybrid algorithm and the GA.

Two hybrid algorithms were proposed by Hu et al. (2003): one combining the Strength Pareto Evolutionary Algorithm (SPEA) and Sequential Quadratic Programming (SQP), and the other combining the Nondominated Sorting Genetic Algorithm (NSGA) and SQP. The two hybrid algorithms were initially tested using benchmark problems

ZDT-1 to ZDT-4 and ZDT-6. The results showed that the hybrid algorithms required less CPU times compared to SPEA and NSGA up to a factor of 10 and 17, respectively. The algorithms were also tested using benchmark problems HU1 to HU5, where the hybrid algorithm of NSGA and SQP performed similarly to NSGA and the hybrid algorithm of SPEA and SQP was shown to have better convergence and diversity.

The algorithm proposed by Xia et al. (2004) was named Hybrid Particle Swarm Optimization (HPSO) for the job-shop scheduling problem, which was an NP-hard problem. This algorithm was based on the PSO and SA and is divided into two stages: the PSO was first used to obtain a set of optimal solutions, and the SA was then used to diversify the solutions and get out of local optimum points. In the job-shop problem, the HPSO was able to obtain the best known solution 74% of the times the algorithm was run. The best result of the algorithm was compared to the best result of SB-GA by Dondorf, GA3 by Mattfeld, and TS-SB by Kolonko where the HPSO's result was found to have 5% lower objective value than the other algorithms.

Wang et al. (2005) proposed a hybrid algorithm named RQGA using Quantum Genetic Algorithm (QGA) and a GA. The QGA uses principles of quantum computing such as Q-bits and linear superposition. The RQGA used one step of QGA followed by one step of GA to enhance and balance the exploration of the solution space while avoiding being trapped to local optima and avoiding premature convergence. This RQGA was tested on two minimization problems, both with a single objective and two variables. On the first problem, the RQGA solution was 0.2% smaller and had 10% more computational time than that by the QGA alone. On the second problem, the RQGA solution was 0.1% smaller and had 13% more time than that by the GA alone.

A hybrid algorithm was proposed by Karen et al. (2006) based on Taguchi's method and a GA. Taguchi's method was used in cascade before using the GA. A two-bar truss problem was used to test the algorithm and the solution by the hybrid algorithm was compared with a generalized center method, GA, game theory, and weighting method. The hybrid algorithm was shown to have similar results (less than 1% difference) to other methods. The hybrid algorithm was also tested using the design of a vehicle part and was found to give similar results to the designs obtained by a GA and a CAD design software.

In the work by Vaz et al. (2006), a hybrid algorithm for global minimization was proposed. This hybrid algorithm was based on PSO as a global optimization method, and pattern search for local minimization. The hybrid algorithm was tested using several benchmark problems including the lj1, lj2, lj3 and hm problems where the hybrid method reached competitive solutions (10^{-2} or less difference compared to the best know solution) to the ones obtained by DIviding RECTangles (Finkel, 2003), adaptive simulated annealing, multilevel coordinate search and parallel genetic algorithm.

Yan et al. (2006) proposed a hybrid optimization algorithm that involves a Support Vector Machine (SVM), which was a classification technique, and a GA. The objective of this algorithm was to classify incoming brain waves read by electroencephalogram. The SVM was a pattern recognizer which classified data sets given their input features, and found an optimal hyperplane by maximizing the margin between classes. This hybrid algorithm calculated the fitness of a given solution that was evaluated using the average classification accuracy of the SVM, and then continued as a

regular GA. The solutions of the hybrid method were found to be 3% to 6% more accurate than those using the GA or SVM alone.

A hybrid algorithm was proposed by Henz et al. (2007) to use a global-local approach in which a GA was used as a global search and a continuous sensitivity equation (CSE) was used in cascade with the GA to perform a local search. This algorithm was initially tested by minimizing the infusion time by changing the injection gate locations in a liquid composite modeling process. In this problem the algorithm was compared to a GA and CSE by themselves, where the CSE gave a solution six times larger than that of the hybrid algorithm, and the GA gave a similar solution to the hybrid algorithm.

In the work by Hiwa et al. (2007), a hybrid optimization algorithm was developed using a stacked approach to the hybridization process. The hybrid algorithm used DIRECT (Finkel, 2003) as a first stage for a uniform global search and a GA as a second stage to improve the search in promising areas and the accuracy of the solutions. Finally, the SQP was used to fine tune the optimal solutions. This hybrid algorithm was tested using single objective minimization problems including the Rosenbrock function, Rastrigin function, and Schwefel function, with ten design variables. For the Rosenbrock function, the average optimum value obtained by the hybrid algorithm was 72% smaller than that by the GA alone, and the number of function evaluations was 44% less than the GA. For the Rastrigin function, the solution by the hybrid algorithm was 50% larger than the GA alone, while the number of function evaluations was reduced by 33%. Finally, for the Schwefel function, the hybrid algorithm produced a solution that was $11x10^9$ times smaller than the GA along and the number of evaluations was reduced by 55%.

Kim et al. (2007) proposed a hybrid algorithm combining a GA and a bacterial foraging algorithm. The hybrid algorithm was tested and the results were compared to those of a GA. The first problem used was a three dimensional paraboloid in which the solution by the hybrid algorithm was one order of magnitude smaller than that by the GA alone. The second test problem was a two dimensional paraboloid in which the results of the two algorithms were similar. Finally, a PID controller for an automatic voltage regulator was designed, in which the results of the two algorithms once again were at the same magnitude.

The hybrid algorithm proposed by Ngo et al. (2007) used a staged continuous Tabu search (SCTS) algorithm as a global search, and the quasi-Newton method as a local search to obtain more accurate solutions. The SCTS strategy was adopted to divide the problem in three stages of Tabu Searches: first it randomly generated a trial solution and performed a generalized search through the solution space to find a promising solution; then, it focused on the search so as to get closer to the optimum based on information from the first stage; and lastly it focused on the search to obtain a solution as accurate as possible to the real optimum. The hybrid algorithm was used in the optimization of band pass filters using fiber bragg gratings. This problem was a single objective optimization problem and the designed filter was compared to the standard sine-apodized filter. As a result, the filter design by the SCTS had a 25-dB lower response in the rejection band decay than the standard filter.

In the work by Tahk et al. (2007), a hybrid algorithm was proposed to combine an EA and a gradient-based method for optimization problems with continuous parameters. The main structure of the hybrid method was similar to that of a regular EA except that a

special individual called the ‘gradient individual’ was introduced and EA individuals were located symmetrically around it. The gradient individual was obtained from the gradient-based search after the EA section in the hybrid algorithm. The hybrid algorithm was tested using two benchmark problems, the Rosenbrock and Griewank functions, and an aircraft control design problem in which the hybrid algorithm was compared to the ES. The hybrid algorithm was found to have faster convergence and higher successful rates than the ES when running multiple times.

Blasa-Canto et al. (2008) developed a two-stage hybrid method in which the first stage used a GA to perform a global search and the second stage adopted a local multiple-shooting approach. In this hybrid method, the GA was used to obtain an approximation of the real optimum solution of a minimization problem. The local multiple-shooting was used to divide an interval into subintervals such that each one contained at least one individual from the GA. The hybrid method was tested using two single-objective optimization problems: the STAT5 signaling pathway problem and the Goodwin’s model for a feedback control system showing a Hopf bifurcation. For the STAT5 problem, the computational time of the hybrid method was 4% of the multiple-shooting approach alone. For the Godwin’s problem, the time consumed by the hybrid algorithm was 2% of the time taken by the multiple-shooting approach alone.

The algorithm proposed by Kalwij et al. (2008) was a non-adaptive hybrid genetic algorithm (AGCT) that integrated a GA and a comprehensive Tabu search. The Tabu search was used as part of the GA to replace the usual mutation operation. Another algorithm presented was global converge, which includes the same algorithms as AGCT plus feedback-driven auto-adaptive control that dynamically changes the Tabu search

parameters during run-time. The global convergence algorithm was compared to AGCT and an advanced GA and was shown to have 18.79% better convergence than AGCT and 31.9% better convergence than the GA.

In the work by Wang et al. (2008), a hybrid optimization method was proposed based on the fusion of the clonal selection algorithm (CSA) and harmony search (HS) technique. The CSA was employed to improve the members of the harmony memory in the HS method. The hybrid optimization algorithm was further used to optimize a fuzzy classification system for the Fisher Iris data classification where the classification rate achieved was 99.2% which was similar to other methods used for the same purposes.

Zahara et al. (2008) proposed a hybrid algorithm that combined a PSO and the Nether-Mead simplex method (NM-PSO). This algorithm was tested with 13 benchmark problems that were studied before by Becerra and Coello (2006) and included minimization and maximization of single objective problems. Compared to the results of previous studies, the NM-PSO was shown in some cases to produce comparative solutions to, if not better than, the cultural differential evolution method, filter SA and GA.

The hybrid algorithm proposed by Zhang et al. (2008) was a method that used an SA and GA. This algorithm was used to overlay two CT/MR and MR images taken at different times to adequately compare them since these images were taken with different parameters such as image resolution, rotation and translation along the three axes. In the hybrid algorithm, the SA was used as an elitism mechanism to maintain search width and to prevent possible premature convergence in the GA. This was a single-objective optimization problem with six design variables.

In the work by Chen et al. (2009), a hybrid optimization algorithm named AHRGA was proposed. This algorithm was a coupled real-coded GA with the steepest descent method as a local search strategy. The hybrid algorithm used GA to find a "suboptimal solution" that was subsequently used as an initial point of the local search method. This algorithm was tested using a dynamic traffic demand problem and compared to an adaptive real-coded GA (ARGA) and a simple GA (SRGA). The solutions of the AHRGA were found to be 5% better on average than those of ARGA and 20% better than those of SRGA. The computing time of the AHRGA was over three times more than the ARGA and SRGA; this indicated that the hybrid algorithm was not very efficient.

Cui et al. (2009) proposed an approach that combined co-evolution, Nash equilibrium and ε-disturbance technique to form a new hybrid approach called HCPSO. This algorithm was tested using six benchmark problems: ZDT1, ZDT2, ZDT3, ZDT4, DTLZ2, and DTLZ4. The HCPSO performed well in the test cases used; its convergence took 60% of the computational time of NSGA-II. The HCPSO was also used in a case study of a cargo ship design in which 2,625 different designs were obtained in the design space including 615 infeasible deigns. The results showed that the best design optimized by HCSO could hold 100 kg more than the best design from NSGA-II.

The hybrid algorithm proposed by Dumas et al. (2009) was an adaptive hybrid method (AHM), which used a GA as a global search and the steepest descent method with a backtracking line search strategy. The hybrid algorithm used the GA in the first stage as a global search, and later when there are clustered partial solutions it used the steepest descent method. The AHM was tested using the *Rast*1 and *Rast*10 problems in

which it performed better than the GA. This algorithm was also applied to the Rastrigin function and the solutions were shown to be comparable to results available in literature.

In the work by Hedayat et al. (2009), a hybrid algorithm composed of NSGA-II and a fast fitness evaluation system based on the cascade feed forward artificial neural networks was proposed. This algorithm was tested in the optimization of the core configuration for research reactors in which the result was a uniform and dense Pareto front over a wide range of function values.

Kampf et al. (2009) proposed a hybrid method combining the covariance matrix adaptation evolution strategy (CMA-ES) and hybrid differential evolution (HDE) algorithms coupled with an efficient backwards ray tracing technique. It was tested using standard benchmarks, including the Ackley function and Rastrigin function, as well as a solar optimization problem. The proposed algorithm outperformed both the standalone CMA-ES and HDE algorithms in both benchmark problems and the solar optimization problem in which the hybrid algorithm reduced the total computational time from 65 to 49 hours.

Nicholson et al. (2009) proposed a hybrid algorithm that combined a binary encoded GA for global search and a SQP for local search. This algorithm was tested by finding the coefficients of a matrix in order for the matrix to have certain zeros, a problem with eight variables. The hybrid algorithm showed significant improvement in performance; the number of iterations needed by the hybrid algorithm was 21% of the ones needed by the conventional GA.

In the work by Padilha et al. (2009), two hybrid algorithms, HA-1 and HA-2, were proposed. The HA-1 algorithm combined a GA with the Boyden-Fletcher-Goldfarb-

Shanno (BFGS) method as a local search. The HA-2 used BFGS after each iteration of the GA on the individuals that were the best fit, i.e. the individuals with the lowest values of the objective function. These individuals were modified using BFGS and reintroduced in the population to be used in the next generation of the GA. These two algorithms were tested using the CGAM problem with a single-objective function and the HA-1 was found to give a better objective value than HA-2 while the later was faster than the former. They were also tested using the CGAM multi-objective problem that was a convex MOOP with two objective functions. Both algorithms were able to obtain the true Pareto front.

Sadegheih (2009) proposed six hybrid algorithms, Hybrid I to VI, for the problem of network planning by combining GA, SA and Tabu search. Hybrid I was a GA that used SA to select individuals for crossover operations. Hybrid II algorithm was a GA that incorporated SA in order to select individuals for crossover and mutation. Hybrid III used Tabu search to filter the individuals before the reproduction stage to prevent unwanted strings to enter future generations. Hybrid IV combined SA and Tabu search where Tabu search was used to prevent reoccurrences of strings from previous generations. Hybrid V was similar to its predecessor by adding the creation of neighborhood solutions. Finally, Hybrid VI was a GA without seeded selection but with a Tabu checking mechanism to prevent recent solutions from re-occurring and with the SA uphill move restriction operator systems. All the algorithms were tested using a single-objective optimization problem where Hybrid VI was found to be the best algorithm.

The hybrid algorithm proposed by Tahk et al. (2009) combined BFGS, a gradient-based technique, with an EA. The hybrid algorithm used the EA to estimate the Hessian

matrix as well as the gradient vector. The hybrid algorithm was compared to RCGA-DHX developed by Herrera et al. (2003) using multiple single-objective optimization problems. The hybrid algorithm was found to outperform RCGA-DHX by six orders of magnitude.

In the work by Vaz et al. (2009), a hybrid solver for linearly constrained global derivative-free optimization called PSwarm was developed. This algorithm was an extension of that by Vaz et al. (2006) and could solve problems with linear constraints. To ensure that the individuals were in the feasible region, the objective values were not calculated unless the constraints were satisfied. This algorithm was compared to ASA, DIRECT, and NOMADm on their performance using problems introduced by Parpas (2006). The PSwarm converged up to 20% faster than the other algorithms.

Yıldız et al. (2009) proposed a hybrid algorithm based on the PSO and the receptor editing property of the immune system. Receptor editing allowed the immune system to identify foreign or own cells. In this algorithm, the receptor editing was used to avoid premature convergence of the PSO. The hybrid algorithm was tested using a tension spring problem and a pressure vessel design problem. In both test cases, the new algorithm produced comparable results to those found in literature, with 66% less function evaluations than algorithms by Coello (2004), Arora (1989) and Belugdu (1992).

The hybrid algorithm proposed by Zhang et al. (2009) combined a PSO and a Tabu search to solve the flexible job-shop scheduling problem, which was an extension to the classical job-shop scheduling problem. This algorithm was compared with the AL-CGA and PSO-SA from literature. All of these algorithms performed comparably in the case of the flexible job-shop scheduling problem.

In the work by Dengiz et al. (2010), a hybrid approach based on ant colony optimization (ACO) and SA, called ACO_SA, was proposed for the design of communication networks. The ACO was used as the global search algorithm and the SA was used as a second stage in order to conduct local search and prevent local optima. In the design of a small sized network, the hybrid algorithm was compared to the SA, ACO, NGA and LS/NGA and was found to outperform the others by finding a solution using less than half of the times by the other algorithms in some cases.

Duan et al. (2010) proposed a hybrid algorithm composed of the artificial bee colony (ABC) and quantum evolutionary algorithm (QEA). In this algorithm, the ABC was used to increase the local search capabilities in order to avoid premature convergence. The hybrid algorithm was tested using the Rosenbrock function and compared to the classical QEA algorithm. The comparison showed that the new algorithm was more practical and effective.

In the work by Elhossini et al. (2010), three hybrid algorithms using an efficient PSO and an EA were studied. In the first algorithm, the PSO update operator was inserted into the EA main loop to improve the performance of the algorithm by exploring the objectives' boundaries. In the second algorithm, the EA solutions, which were obtained at a finite number of iterations or until satisfying a specific criterion, were used as the initial population of the PSO. In the third algorithm, the PSO and EA were used in the opposite order as that in the second algorithm, i.e., the PSO solutions were used as the initial population of the EA. These three algorithms were tested using benchmark problems by Kurosawe (1991), Zdt-4 and Zdt-6 in which the first and the third algorithms were found to generate better solutions but run slower than the second algorithm.

In this dissertation, hybrid strategies based on a GA (i.e., NSGA-II) and a gradient-based method (i.e., FSQP) were studied to develop an efficient and effective strategy that would have the diversity of the Pareto oriented solutions by GA and the accuracy of individual solutions by the gradient-based method. The remaining portion of this dissertation is arranged as follows. The general procedure of solving optimization problems is first introduced in Chapter 2. The gradient-based methods and genetic algorithms are then presented in Chapters 3 and 4, respectively, along with discussions of the pros and cons of these methods. In Chapter 5, hybrid optimization strategies are proposed for multi-objective optimization problems and are evaluated using three benchmark problems. In chapter 6 the best hybrid algorithm proposed in the previous chapter is applied to other benchmark problems to identify the strengths and drawbacks of this strategy. The new hybrid method is then applied to a real world application problem and the results and analysis are presented in Chapter 7, followed by the concluding remarks given in Chapter 8.

CHAPTER 2: DESIGN OPTIMIZATION

In mathematics, optimization refers to choosing the best element from a set of feasible alternatives. This means solving problems by minimizing or maximizing a given function or a set of functions while satisfying the constraints of some preset conditions. In engineering, optimization is used to improve engineering systems in a large number of fields such as materials science, thermodynamics, dynamics, controls, machine design, and structural design. This chapter will introduce the basic concepts of design optimization (or optimum design) as well as notations and the standard formulation.

2.1. Design Process

The traditional design process largely depends on the designer's experience and skills, which may lead to erroneous results in the design of complex systems. Figure 2.1 shows a flowchart for a traditional design process. The main advantage of the traditional design process is that the designer's experience can be used to make conceptual changes or to add additional specifications such as changing structural elements, geometries, or types of components. However, when it comes to designs with complex constraints and a large number of design variables, the designer may find it difficult to decide any changes that may lead to better designs and still satisfy the constraints of the particular design problem.

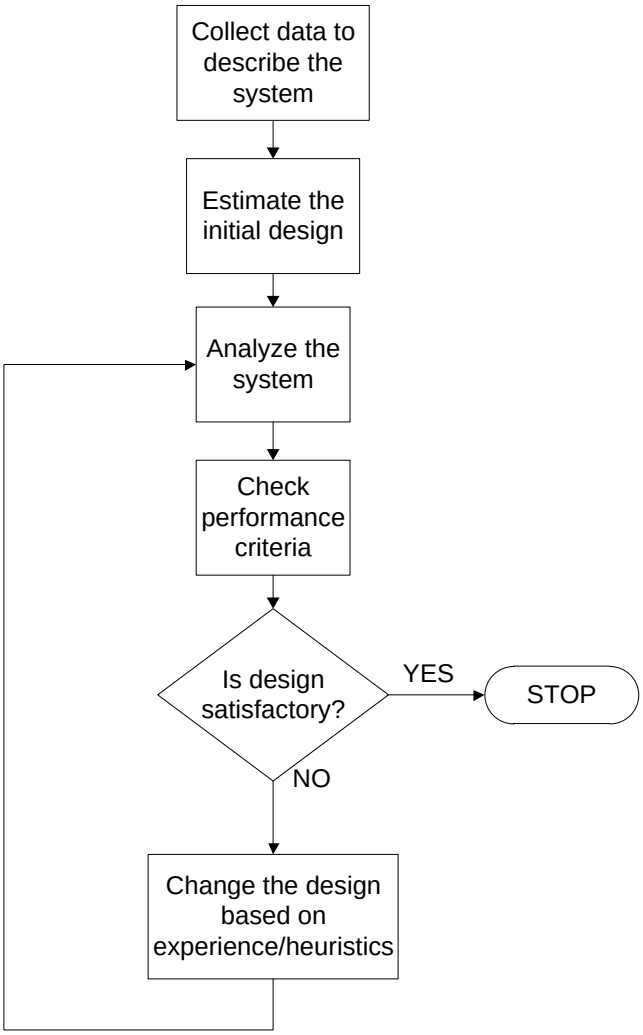

Figure 2.1: A traditional design process

In contrast to the traditional design process, an optimum design process, as shown in Figure 2.2, requires the designer to explicitly identify the set of design variables, objective functions, and constraints. By doing so, an optimum design can be obtained by using trend information to make better decisions on design changes than the traditional design. Nevertheless, the optimum design process may be greatly benefitted by the experience of the designer in formulating the problem and identifying certain constraints.

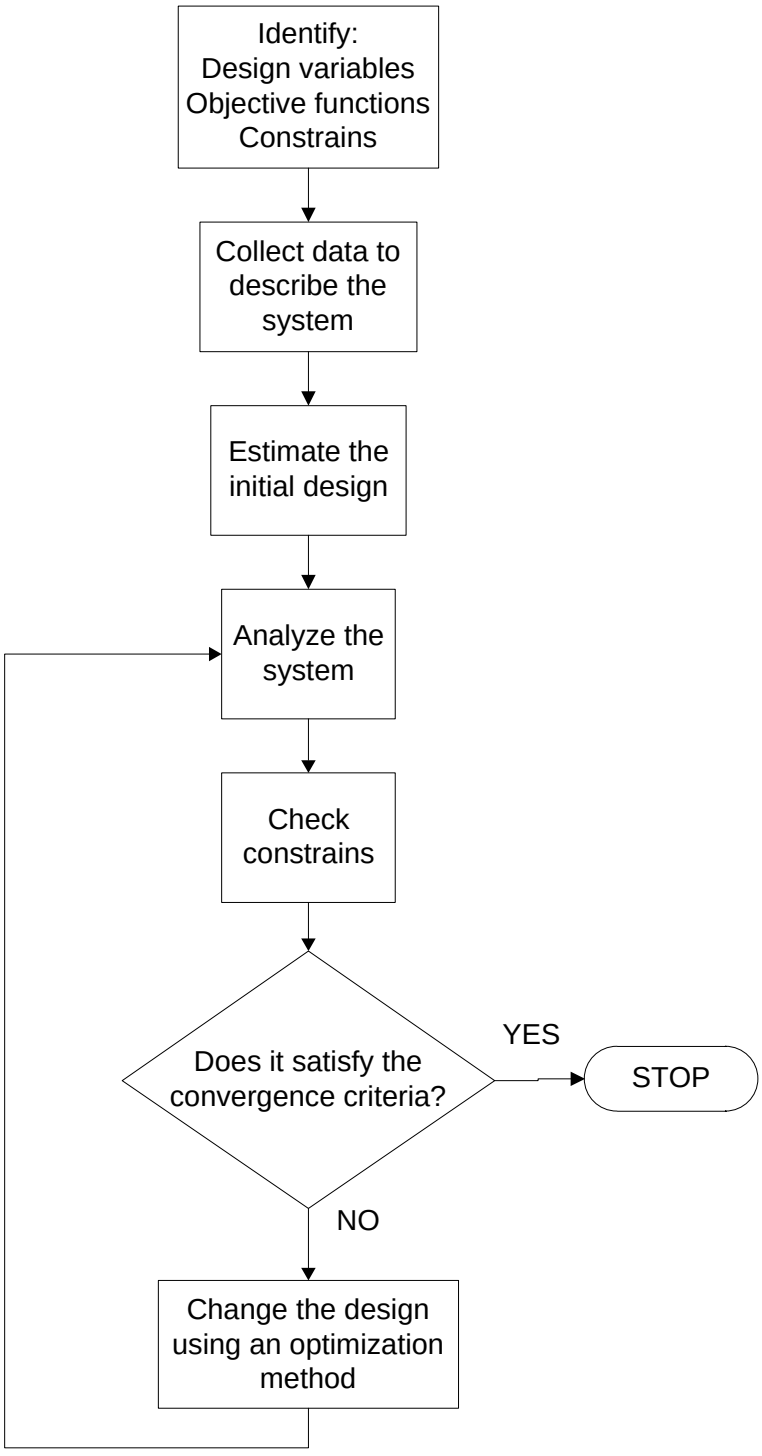

Figure 2.2: An optimum design process

2.2. Problem Formulation

An optimum solution is as good as its formulation; therefore it is important to follow well defined procedures to formulate a design optimization problem. If a critical constraint is missing in the formulation, the optimum solution will most likely be an infeasible one because the missing constraint will increase the size of the design space including infeasible designs. Also if the constraints are inconsistent or redundant, there may not be a feasible solution to the design problem. Proper formulation of an optimization problem leads to the proper solution of the problem solved by an optimization algorithm.

The formulation of an optimum design problem involves translating a descriptive statement into well-defined mathematical equations. The followings are key steps to develop a mathematical formulation for design optimization problems.

2.2.1 Problem Statement

The design problem begins by developing a linguistic statement of the problem, which should include all the objectives to be met and the associated requirements. The goals of the problem statement should be as clear as possible to the designer in order to translate the linguistic statement into mathematical equations.

2.2.2 Data and Information Collection

In order to develop a proper mathematical formulation of the design problem, data and information need to be collected on properties, performance, resource limits, materials costs, and other information relevant to the design problem. Analysis procedures and tools such as software packages should also be identified in this stage.

2.2.3 Definition of Design Variables

In this stage, design variables of the optimization problem are selected and the range values (i.e., lower and upper bounds) are identified. In order to do so, the relevant operational principles, such as mechanical, electrical, and/or chemical specifications, should be clear to the designer in order to identify the minimum amount of relevant independent variables. A set of design variables is typically denoted as $x_j, j = 1,2,3, \dots, n$ and can be represented in vector form as

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2-1}$$

2.2.4 Constraints

A restriction placed on a design is called constraint. Constraints are imposed by the particular characteristics of the design or resources at hand. In order to have a satisfactory design, all constraints must be satisfied. If at least one constraint is not satisfied, the design is considered to be unacceptable and is referred to as infeasible. There are two types of constraints: inequality constraints and equality constraints that are defined as follows.

Inequality constraints: $g_i(\vec{x}) \leq 0;\ i = 1,2, \dots, m$ (2-2)

Equality constraints: $h_i(\vec{x}) = 0;\ i = 1,2, \dots, p$ (2-3)

2.2.5 Optimization Criterion

In order to determine if a particular design is better than another one, a certain criterion is necessary for design evaluation. In optimization, a criterion is typically given in the form of a scalar function or a set of functions, whose numerical value can be obtained once a design is specified. In other words, a criterion function is a function of

the design variables $\vec{x}$. These criteria are called objective functions for optimum design problems, which need to be minimized or maximized. The objective functions must be directly influenced by the design variables; otherwise, they are not meaningful to the optimization problem.

The objective functions are designated as: $f_1(\vec{x}), f_2(\vec{x}), \dots, f_k(\vec{x})$, where k is the number of objective functions in the optimization problem. The objective functions in a vector form are represented as:

$$\vec{f}(\vec{x}) = \begin{bmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_k(\vec{x}) \end{bmatrix} \tag{2-4}$$

2.3. Standard Formulation of an Optimization Problem

The standard formulation is a general mathematical representation and description of the optimization problem. In the standard formulation, the optimization problem is always defined as the minimization of one or a set of objective functions that are subject to all inequality and equality constraints. The standard formulation is given as:

$$\text{Min.} \qquad \vec{f}(\mathbf{x}) = \begin{bmatrix} f_1(x_1 x_2, , \dots x_n) \\ f_2(x_1 x_2, , \dots x_n) \\ \vdots \\ f_k(x_1, x_2, \dots x_n) \end{bmatrix} \tag{2-5}$$

$$\text{s.t.} \qquad h_j(\mathbf{x}) = h_j(x_1, x_2, \dots x_n) = 0;\ j = 1, \dots, p \tag{2-6}$$

$$g_i(\mathbf{x}) = g_i(x_1, x_2, \dots x_n) \leq 0;\ i = 1, \dots, m \tag{2-7}$$

Note that the inequality constraints should be of the “less-than-or-equal-to” type, i.e., in the form of $g_i(\vec{x}) \leq 0$. For a maximization problem (i.e., the objective function is to be maximized), the formulation can be expressed as a minimization as

$$\text{Max. } f(\vec{x}) = \text{Min. } -f(\vec{x}) \tag{2-8}$$

where $f(x)$ is the objective function of the original maximization problem. The solutions of Eq. (2-8) are the same as those of the original maximization problem, as graphically illustrated in Figure 2.3. The plots of the two functions, $F(x)$ and $-F(x)$, show that both the maximum value of $F(x)$ and the minimum value of $-F(x)$ occur at point x^*, which is the solution of both $F(x)$ and $-F(x)$. Therefore, the maximization of $F(x)$ is equivalent to the minimization of $-F(x)$.

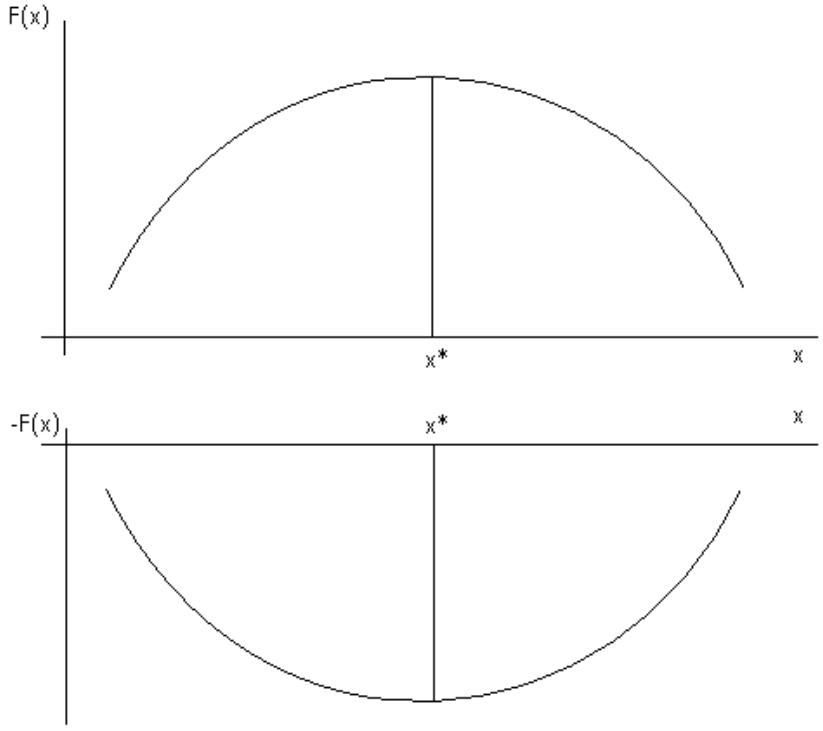

Figure 2.3: Equivalency of max $F(x)$ to min $-F(x)$

The standard optimization formulation handles only the "less-than-or-equal-to" type ($\leq$) of inequality constraints. If an optimization problem has a "greater-than-or-equal-to" constraint, i.e., $G_j(x) \geq 0$, then that constraint need be converted to the standard form by multiplying it by -1, given as follows:

$$g_j = -G_j(x) \leq 0 \qquad (2\text{-}9)$$

2.4. Solution to the Optimization Problem

The solution to an optimization problem is typically a single point for a single-objective problem and a set of points for a multi-objective problem. In both cases, the solution(s) should not violate any of the constraints.

2.4.1 Single-Objective Optimization

In the case of single-objective optimization, the solution is typically a single point in the solution space or a set of points along an active constraint. An active constraint is defined as an inequality constraint, $g_j(x) \leq 0$, satisfying the equality condition of $g_j(x^*) = 0$ at point x^*. When no constraint of the problem is violated, the solution point or points are considered a feasible solution or solutions. For example, consider the following problem:

Min. $f(x) = x^2$ (2-10)

s.t. $g_1(\vec{x}) = x - 2 \leq 0$ (2-11)

$g_2(\vec{x}) = -2 - x \leq 0$ (2-12)

Figure 2.4 shows the graphical illustration of the problem and its solution. It can be seen that the global minimum of $f(x)$ is located at $x = 0$ without considering the constraints. If the constraints are considered, then the feasible points become those that are above the red line in Figure 2.4; therefore, the solutions are x = -2 and x = 2, which make the constraints active.

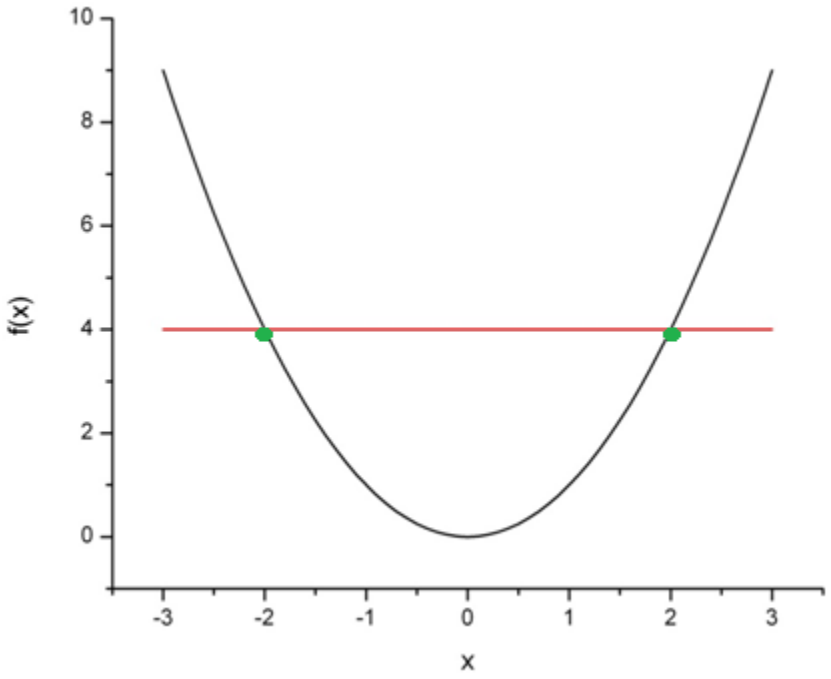

Figure 2.4: Graphical illustration of optimization problem in Eqs. (2-10) to (2-12)

2.4.2 Multi-Objective Optimization

In multi-objective optimization, the solution is typically a set of points rather than a single point. All the points in the solution set are called nondominated solutions, because there is no single solution that is better than another for all the objectives. In other words, a point $\vec{x}^*$ is nondominated in a minimization problem where it doesn't exist another point $\vec{x}$ such that $\boldsymbol{f}(\vec{x}) < \boldsymbol{f}(\vec{x}^*)$. That there is no point that improves all the objective functions simultaneously. However, there may be points that improve some of the objective functions. Consider the following problem:

Min. $$f_1(x) = (x-2)^2 \quad (2\text{-}13)$$

$$f_2(x) = (x+2)^2 \quad (2\text{-}14)$$

The graphical illustration of the problem is shown in Figure 2.5 in which there is not a clear minimum in the problem. For example, the minimum point of $f_1(x)$ corresponds to a point that is far from the minimum of $f_2(x)$.

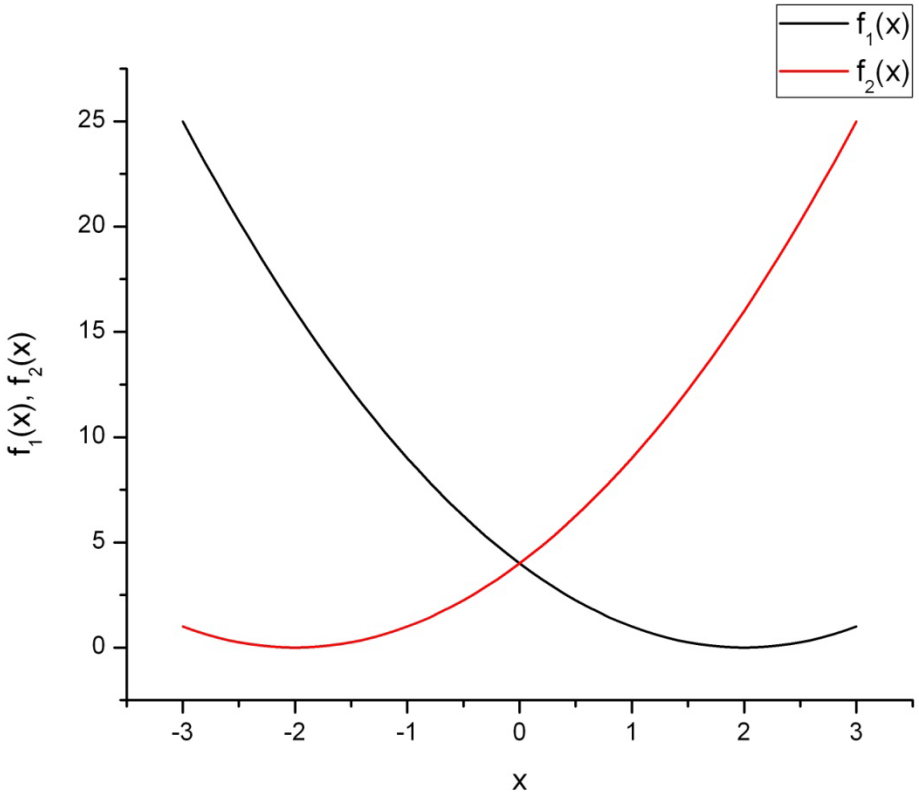

Figure 2.5: Graphical illustration of Eqs. 2-13 and 2-14

To illustrate the dominancy of the solutions, the objective or solution space is shown in Figure 2.6. For the problem given by equations (2-13) and (2-14), the solution space is a plane formed by $f_1(x)$ as the horizontal axis and $f_2(x)$ as the vertical axis.

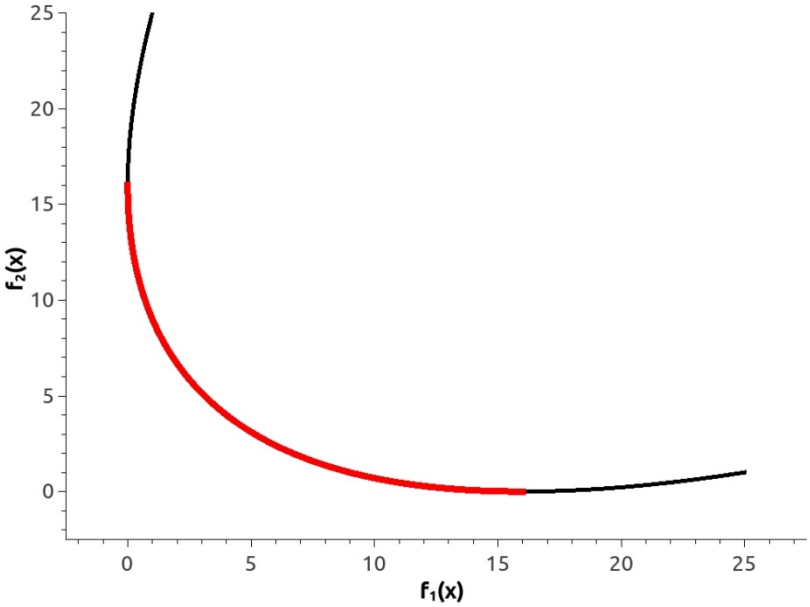

Figure 2.6: Solution space of 2-13 and 2-14

In Figure 2.6, the feasible and nondominated solutions are those on the red line, which is formally called the Pareto Front. The points on the black line are dominated by at least one point on the red line. Among the nondominated solutions, it can be seen that there is no single solution having smaller values of both $f_1(x)$ and $f_2(x)$ than any other solution. The nondominated solutions fall in the range of $[-2 \leq x \leq 2]$ in the design space, corresponding to the red line in the objective space in Figure 2.6. The solution of the problem gives the designer the liberty to choose any of the solution points in the Pareto front, depending on the designer's preference on the trade-off between the two objectives.

CHAPTER 3: GRADIENT-BASED METHODS

Gradient-based methods are iterative methods developed to solve single-objective optimization problems. These methods were developed in order to solve nonlinear optimization problems that were very difficult to solve using analytical methods. As their names indicate, these methods require the gradients of the objective function and constraint functions in order to determine the search directions and step sizes to find the optimal solution. These numerical methods are rarely done by hand; they need to be implemented into computer programs due to the complexity and amount of computational work.

Before discussing gradient-based methods, it is necessary to take a look at the line search methods. These methods are used as part of many gradient-based methods to calculate a step size by solving a one-dimensional minimization problem.

The basic idea of a line search method is to reduce an interval of uncertainty around the minimum of an objective function in each iteration. Therefore, the line search methods do not give an exact solution; instead they give a range in which the minimum of the objective function is located. Furthermore, gradient-based methods use line searches as a subsequent stage after determining the search direction, which is determined by using the first- or even higher-order information of the objective and constraint functions.

3.1. Equal Interval Search

Equal interval search is a simple method used to minimize a function $f(\alpha)$ in which an interval for the single variable, a, is determined, i.e., $\alpha_0 \leq \alpha \leq \bar{\alpha}$. The interval is then used as the solution to the problem. This method is divided in two phases and the first phase is given by the following algorithm:

Step 1: Select a small number δ, and set the iteration counter $k = 0$.

Step 2: Obtain the function values $f(k\delta + \alpha_0)$ and $f\big((k+1)\delta + \alpha_0\big)$.

Step 3: If $f(k\delta + \alpha_0) \leq f\big((k+1)\delta + \alpha_0\big)$, then go to the second phase; otherwise increase the iteration counter k by 1, and go to Step 2.

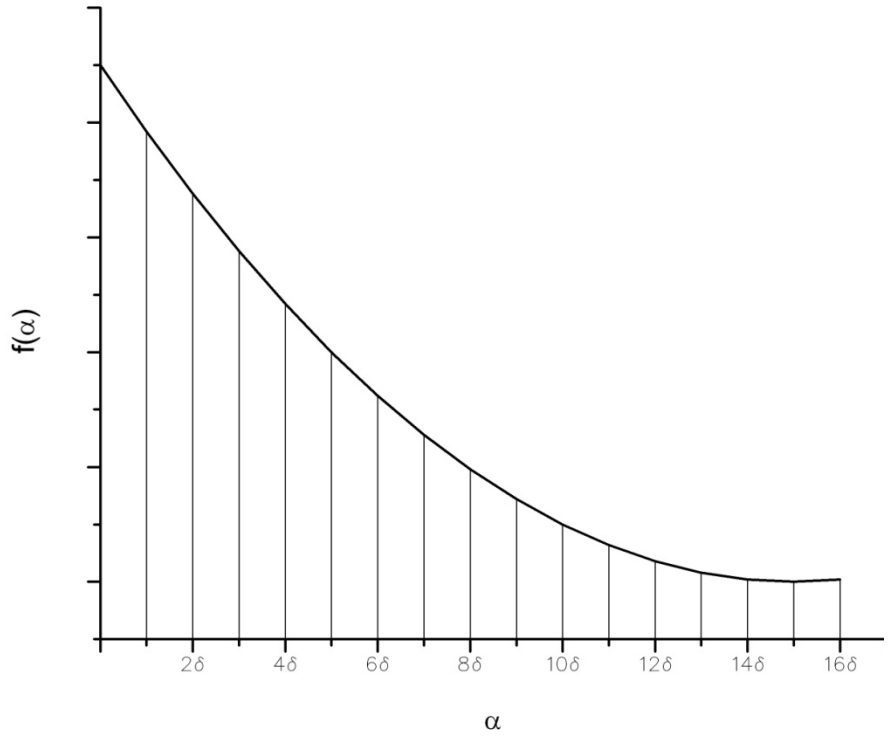

Figure 3.1: Equal interval search phase I

On a unimodal function, if the condition established in Step 3 is met, then the minimum point of the objective function is determined to reside in the range of $(k-1)\delta$ and $(k+1)\delta$. At this point the second phase is used to further reduce this interval enclosing the solution. The second phase algorithm is the same as the first one except that the second phase uses $\alpha_0 = (k-1)\delta$ and the step δ is changed to $r\delta$, where $r \ll 1$ so that the uncertainty of the solution is reduced to $2r\delta$. If the uncertainty needs to be further reduced, then the second phase can be used again and the uncertainty can be reduced to $2r^2\delta$.

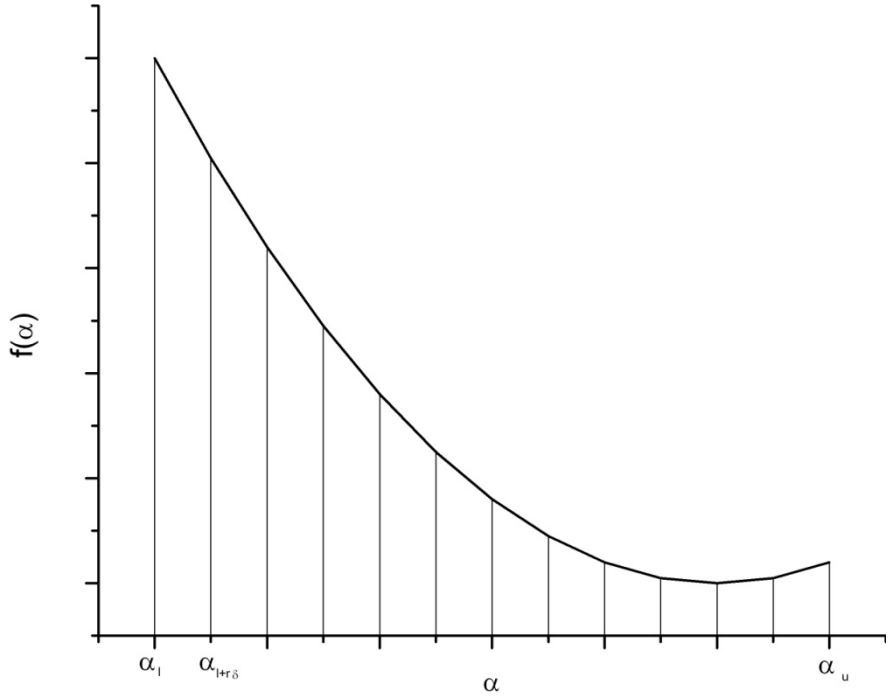

Figure 3.2: Equal interval search phase II

The efficiency of the equal interval search method depends on the number of evaluations needed to get to an acceptable solution. In other words, it depends on the required uncertainty level, which is determined by the values of the two parameters

δ and r. If δ is very small, the number of iterations in the first phase will be increased, but the second phase may require less function evaluations.

A variation of this method is to set the parameter δ dependent on the value of $\bar{\alpha}$ so as to reduce the number of parameters to be manually selected and thus improve the efficiency of the algorithm.

3.2. Golden Section Search

Golden section search is an improved method over the equal interval search and is one of the best methods for interval search. The golden section search method also has two phases. In the first phase, the increment δ is not a constant and this value changes by multiplying with the golden ratio 1.618 in each iteration to make the algorithm more efficient in locating the minimum value. In the second phase, the range located in the first phase is reduced by a value of the inverse of the golden ratio, i.e., 1/1.618 = 0.618. One of the properties of the golden ratio is:

$$\frac{1}{1.618} = 0.618 \tag{3-1}$$

The golden section search method is given by the following algorithm:

Step 1: Select a small number δ, set a convergence parameter $\varepsilon > 0$, and set the iteration counter $k = 0$.

Step 2: Evaluate the functions $f\left(\sum_{j=0}^{k} \delta(1.618)^j\right)$ and $f\left(\sum_{j=0}^{k+1} \delta(1.618)^j\right)$.

Step 3: If $f\left(\sum_{j=0}^{k} \delta(1.618)^j\right) > f\left(\sum_{j=0}^{k+1} \delta(1.618)^j\right)$, then increase the iteration counter k by 1, and go to Step 2, otherwise continue.

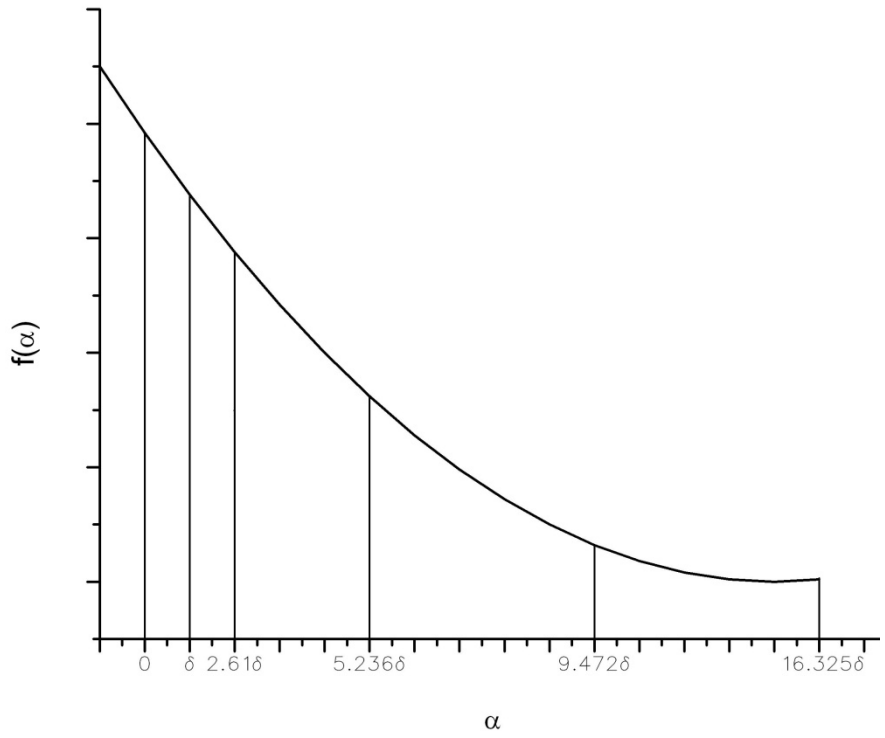

Figure 3.3: Golden section search steps 2 and 3

Step 4: Set $\alpha_l = f\left(\sum_{j=0}^{k-1} \delta(1.618)^j\right),\ \alpha_u = f\left(\sum_{j=0}^{k+1} \delta(1.618)^j\right), I = \alpha_u - \alpha_l,$

$\alpha_a = f\left(\sum_{j=0}^{k} \delta(1.618)^j\right),\ \alpha_b = \alpha_l + {}^{I}/_{1.618}$. Find $f(\alpha_b)$.

Step 5: Compare $f(a)$ and $f(b)$ and go to i, ii, or iii

i) If $f(a) < f(b)$, then the minimum point lies between α_l and α_b, set $\alpha_u = \alpha_b,\ \alpha_b = \alpha_a$. Find $\alpha_a = \alpha_l + 0.382(\alpha_u - \alpha_l)$, compute $f(\alpha_a)$ and go to Step 6.

ii) If $f(a) > f(b)$, then the minimum point lies between α_a and α_u, set $\alpha_l = \alpha_a,\ \alpha_a = \alpha_b,\ \alpha_b = \alpha_l + 0.618(\alpha_u - \alpha_l)$, compute $f(\alpha_b)$ and go to Step 6.

iii) If $f(a) = f(b)$, set $\alpha_l = a, \alpha_u = b$ and return to Step 2.

Step 6: Compute the new I, if $I < \varepsilon$, let $\alpha^* = (\alpha_b - \alpha_a)/2$ and stop. Otherwise go to Step 5.

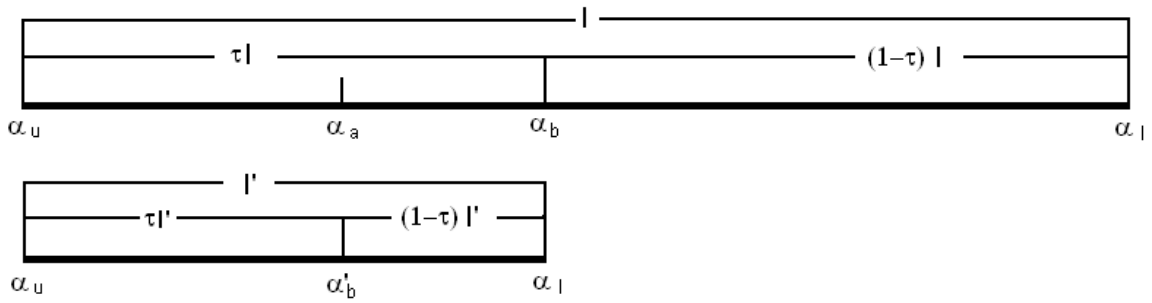

Figure 3.4: Golden section search step 5

It is important to note that both golden section search and equal interval search are line search methods that do not require determining the search direction. The following methods to be discussed require the search directions to be determined in addition to step sizes, which are typically determined by a line search method.

3.3. Steepest Descent Method

The steepest descent method is a first-order method introduced by Cauchy in 1847. It utilizes the gradient information to determine the descent direction of the objective function and find its local minimum. In each iteration of this method, the gradient of the objective function is used to find a direction d along which the cost function $f(\vec{x})$ decreases at its highest rate. A one dimensional search algorithm is then used to determine the necessary step size along the previously determined search direction. This method is a first-order method; in other words, only first-order information (or the gradient) of the cost function needs to be calculated.

The search direction is given by the negative gradient of the objective function, $-\nabla f(\vec{x})$, at the current point. Since the gradient vector gives the direction along which the function grows the most, the negative gradient thus gives the direction of function decrease, which is needed to find the minimum value of the objective.

The following gives the algorithm of the steepest descent method:

Step 1: Choose a starting point $\vec{x}^{(0)}$, choose a convergence parameter $\varepsilon > 0$ and set the iteration counter $k = 0$.

Step 2: Calculate $\boldsymbol{c}^{(k)} = \nabla f\left(\vec{x}^{(k)}\right)$

Step 3: Calculate $\left\|\boldsymbol{c}^{(k)}\right\|$. If $\left\|\boldsymbol{c}^{(k)}\right\| < \varepsilon$ then stop and $\vec{x}^{(k)}$ is the minimum point, otherwise continue.

Step 4: Calculate a step size α_k that minimizes $f\left(\boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{c}^{(k)}\right)$, using any of the one-dimensional search algorithms.

Step 5: Update the design $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k \boldsymbol{c}^{(k)}$. Increment the iteration counter k by 1, and go to Step 2.

The steepest descent method is a simple and robust method that guarantees convergence. However, it has some drawbacks such as the large number of iterations needed to reach a local minimum point. In addition, it does not use the information obtained from previous iterations, that is, each iteration is independent and orthogonal from each other, as illustrated in Figure 3-1. These orthogonal search directions do not give the best search direction in the global scheme and thus make the algorithm inefficient. Moreover, this method only uses first-order information of the objective function; this is another contributor to the slow convergence of the method. The rate of

convergence of the steepest descent method also depends on the accuracy of the one dimensional search that is used to determine the step size.

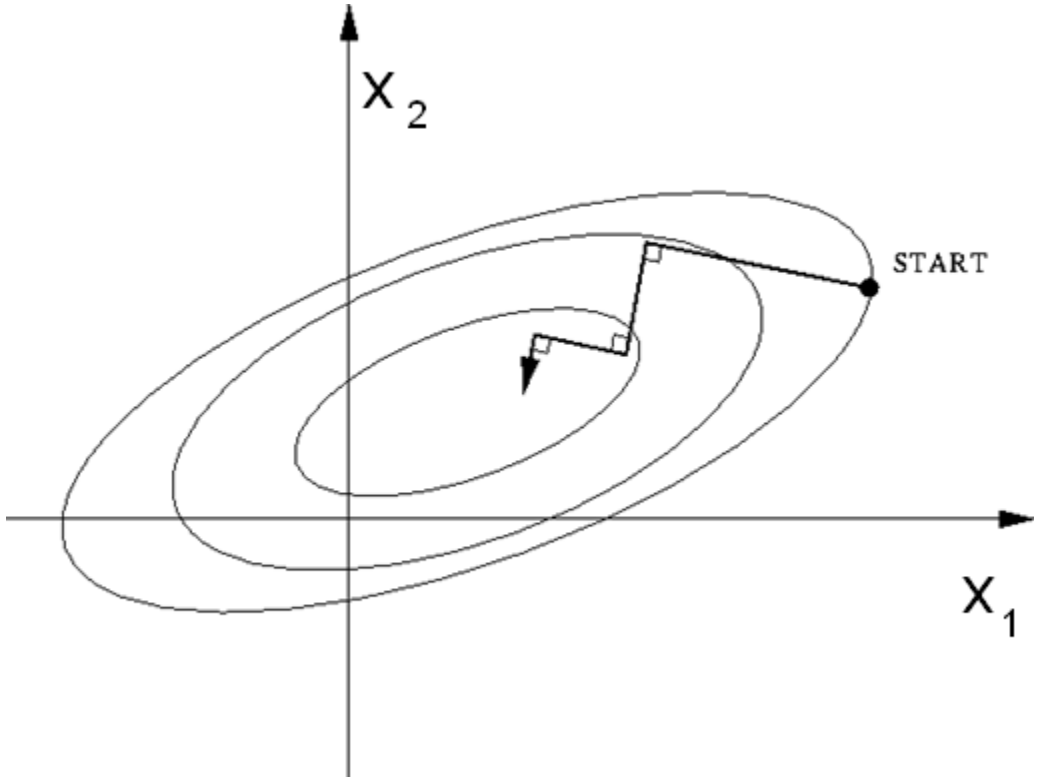

Figure 3.5: Steepest descent method approaching a local minimum

3.4. Conjugate Gradient Method

The conjugate gradient method is a simple yet effective modification of the steepest descent method. In the steepest descent method, two consecutive search directions are orthogonal to each other and thus make the algorithm converge slowly when approaching the optimum. In the conjugate gradient method, the search directions are calculated such that they are not orthogonal to each other. The search direction at the current point is determined such that it cuts diagonally through the current steepest descent directions and the search direction immediately before the current iteration. This

makes the conjugate gradient method converge much faster than its predecessor, i.e., the steepest descent method. The following gives the algorithm of the conjugate gradient method:

Step 1: Choose a starting point $\vec{x}^{(0)}$, choose a convergence parameter $\varepsilon > 0$ and set the iteration counter $k = 0$, calculate $\boldsymbol{c}^{(0)} = \nabla f\left(\vec{x}^{(0)}\right)$, if the stopping criterion $\left\|\boldsymbol{c}^{(k)}\right\| < \varepsilon$, then stop. Otherwise go to Step 5.

Step 2: Calculate $\boldsymbol{c}^{(k)} = \nabla f\left(\vec{x}^{(k)}\right)$

Step 3: Calculate $\left\|\boldsymbol{c}^{(k)}\right\|$. If $\left\|\boldsymbol{c}^{(k)}\right\| < \varepsilon$ then stop and $\vec{x}^{(k)}$ is the minimum point.

Step 4: Calculate the conjugate direction as:

$$\boldsymbol{d}^{k} = -\boldsymbol{c}^{(k)} + \beta_k \boldsymbol{d}^{(k-1)} \tag{3-2}$$

$$\beta_k = \left(\frac{\left\|\boldsymbol{c}^{(k)}\right\|}{\left\|\boldsymbol{c}^{(k-1)}\right\|}\right)^2 \tag{3-3}$$

Step 5: Calculate a step size α_k that minimizes $f\left(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}\right)$ using any one dimensional search algorithm.

Step 6: Update the design $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$. Increment the iteration counter k by 1, and go to Step 2.

The only modification from the steepest descent method is the calculation of $d^{(k)}$, which requires some additional calculations. However, it substantially improves the rate of convergence as shown in Figure 3-2. The conjugate gradient method is preferred over the steepest descent method, because the former has a capability that the later clearly does not possess.

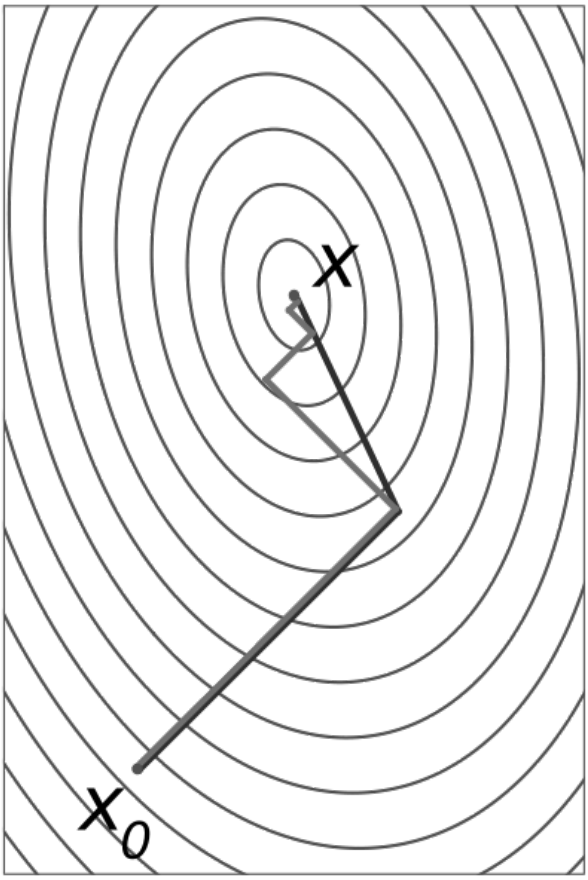

Figure 3.6: Steepest descent method vs. conjugate gradient method

3.5. Quadratic Programming (QP)

The quadratic programming (QP) method is a search direction determination algorithm in which the descent direction is calculated based on a linearized sub problem along with a line search method to calculate a step size that minimizes the objective function. The sub problem is defined as:

$$\min \bar{f} = \boldsymbol{c}^T \boldsymbol{d} + \frac{1}{2} \boldsymbol{d}\boldsymbol{d}^T \tag{3-4}$$

$$\text{s.t. } \mathbf{N}^T \boldsymbol{d} = \boldsymbol{e} \tag{3-5}$$

$$\mathbf{A}^T \boldsymbol{d} \leq \boldsymbol{b} \tag{3-6}$$

where the negative of the j^{th} equality constraint function is defined as:

$$e_j = -h_j\left(\boldsymbol{x}^{(k)}\right) \tag{3-7}$$

and the negative of the j^{th} inequality constraint function is given by:

$$b_j = -g_j\left(\boldsymbol{x}^{(k)}\right) \tag{3-8}$$

The derivative of the objective function with respect of x_i is calculated as:

$$c_i = \frac{\partial f(\boldsymbol{x}^k)}{\partial x_i} \tag{3-9}$$

The matrix of the derivative of h_j with respect of x_i is:

$$n_{ij} = \frac{\partial h_j(\boldsymbol{x}^k)}{\partial x_i} \tag{3-10}$$

The matrix derivative of g_j with respect of x_i is:

$$\alpha_{ij} = \frac{\partial g(\boldsymbol{x}^k)}{\partial x_i} \tag{3-11}$$

The design change is:

$$d_i = \Delta\left(\boldsymbol{x}_i{}^{(k)}\right) \tag{3-12}$$

A QP sub problem is strictly convex and thus it has a unique global minimum which would not be possible if the QP sub problem is non-convex.

3.6. Constrained Steepest Descent (CSD)

The CSD method introduces a penalty parameter and applied it to the objective function for constraint violations in order to ensure that the solution is feasible. This method is proven to converge to a local minimum from any starting point. The following gives the algorithm of the CSD method:

Step 1: Set k = 0 and choose a starting point $\vec{x}^{(0)}$. Select an initial value of the penalty parameter R_0 and two small numbers ε_1 and $\varepsilon_2$that define the permissible constraint violation and convergence parameter, respectively.

Step 2: Calculate the values and gradients of the cost and constraint functions at $\boldsymbol{x}^{(k)}$. Calculate the maximum constraint violation as

$V_k = \max(0; |h_1|, \dots, |h_p|, g_1, \dots, g_m)$.

Step 3: Define the QP sub problem as given in section 3.5 using the calculations from the previous step, and solve it to get the search direction $\boldsymbol{d}^{(k)}$ and the Lagrange multipliers $\boldsymbol{v}^{(k)}$ and $\boldsymbol{u}^{(k)}$.

Step 4: Check the stopping criteria $\|\boldsymbol{d}^{(k)}\| \leq \varepsilon_2$ and the maximum constraint violation $V_k \leq \varepsilon_1$. If both conditions are met then stop, otherwise continue.

Step 5: Calculate $r_k = \sum_{i=1}^{p} |v_i^{(k)}| + \sum_{i=1}^{m} u_i{}^{(k)}$ and set $R = \max(R_k, r_k)$.

Step 6: Find α_k using a line search algorithm. Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$.

Step 7: Save the penalty parameter $R_{k+1} = R$. Update the iteration counter k by 1, and go to Step 2.

For unconstrained optimization problems, the CSD method is reduced to its predecessor, the steepest descent method.

3.7. Newton's Method

Methods such as the conjugate gradient, QP, and CSD only require the calculation of first-order information of the objective and constraint functions in order to determine the search direction. In Newton's method, the second-order information is used to improve the efficiency in locating the optimal value of the objective function. With the

second-order information, i.e., the Hessian matrix, the method has a second-order convergence rate. If the objective function is quadratic and positive definite, the method converges in one iteration with a step size of one.

The basic idea of Newton's method's is to use a second-order Taylor's expansion of the change of the objective function at the current point, resulting in a quadratic expression of the objective change in terms of the design change given by

$$f(\vec{\boldsymbol{x}} + \Delta\vec{\boldsymbol{x}}) - f(\vec{\boldsymbol{x}}) = \vec{\boldsymbol{c}}^T \Delta\vec{\boldsymbol{x}} + 0.5\Delta\vec{\boldsymbol{x}}^T \mathbf{H} \Delta\vec{\boldsymbol{x}} \tag{3-13}$$

where **H** is the Hessian of the objective function f at the current point $\vec{\boldsymbol{x}}$, c is the gradient vector of f, and $\Delta\vec{\boldsymbol{x}}$ is a small change of the design variables $\vec{\boldsymbol{x}}$. If the Hessian of f is positive semi-definite, then there exists a $\Delta\vec{\boldsymbol{x}}$ that will find the global minimum of the objective. If the Hessian is strictly positive definite, the minimum is unique. The optimality condition is given by

$$\frac{\partial f}{\partial(\Delta\vec{\boldsymbol{x}})} = \vec{\mathbf{c}} + \mathbf{H}\Delta\mathbf{x} = \vec{0} \tag{3-14}$$

and $\Delta\mathbf{x}$ is solved by

$$\Delta\vec{\mathbf{x}} = -\mathbf{H}^{-1}\mathbf{c} \tag{3-15}$$

The design update is given by

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \Delta\mathbf{x}$$

Since Eq. (3-13) is an approximation for f the update may not be exact minimum point there is the need to define an iterative procedure.

The algorithm of the Newton's method is given as follows:

Step 1: Choose a starting point $\vec{\boldsymbol{x}}^{(0)}$ and set the iteration counter $k = 0$.

Step 2: Calculate $\Delta\vec{\boldsymbol{x}}^{(k)} = -\mathbf{H}^{-1(k)}\vec{\boldsymbol{c}}^{(k)}$.

Step 3: Update the design as $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \Delta x$. If $k = n(n+1)/2$, where n is the number of design variables, stop, otherwise increase k and go to Step 2.

At the end of each iteration, the Hessian matrix $\boldsymbol{H}$ and its inverse need to be recalculated; thus significant computational effort is required. It should be noted that there is no line search associated with Newton's method; this means that there is no certainty that the descent condition $f\left(\boldsymbol{x}^{(k+1)}\right) < f(\boldsymbol{x}^{k})$ is satisfied.

3.8. Modified Newton's Method

The Newton's method does not guarantee that the cost function is reduced in each iteration. To correct this deficiency, the modified Newton's method is developed by incorporating a line search that calculates a step so as to reduce the cost function's value. The following gives the algorithm for the modified Newton's method.

Step 1: Choose a starting point $\boldsymbol{x}^{(0)}$, set the iteration counter $k = 0$, and select a convergence parameter ε.

Step 2: Calculate $\boldsymbol{c}^{(k)} = \nabla f\left(\vec{x}^{(k)}\right)$. If $\|\boldsymbol{c}^{k}\| < \varepsilon$, stop. Otherwise, continue.

Step 3: Calculate $\mathbf{H}^{(k)}$.

Step 4: Calculate $\boldsymbol{d}^{(k)} = -\left[\mathbf{H}^{(k)}\right]^{-1}\boldsymbol{c}^{(k)}$.

Step 5: Calculate the step size α that minimizes $f(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)})$ using any line search algorithm.

Step 6: Update the design $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}^{(k)}$. Set $k = k + 1$, go to Step 2.

It is important to know that unless the Hessian matrix $\mathbf{H}$ is positive definite, the descent condition $f\left(x^{(k+1)}\right) < f(x^{k})$ may not be satisfied. In the case of a singular matrix $\mathbf{H}$, the descent direction $d^{(k)}$ cannot be calculated.

3.9. Quasi-Newton Methods

Both the Newton's method and modified Newton's method require the calculation of the Hessian matrix **H** in each iteration. This calculation not only significantly increases the computational cost, but also may not always result in a positive definite Hessian, which is required to ensure the reduction of the objective function. To this end, quasi-Newton methods were developed to overcome both the above mentioned deficiencies by using approximate Hessian matrices or their inverses that would always be positive definite. In this section, two commonly used quasi-Newton methods, the Davidson-Fletcher-Powell (DFP) method and the Boyden-Fletcher-Goldfarb-Shanno (BFGS) method are introduced.

3.9.1 DFP Method

The DFP method is a quasi-Newton method that builds and uses a first-order approximate matrix of the Hessian's inverse of the cost unction $f(x)$. By calculating an approximate of the Hessian's inverse, the complexity of the calculations is significantly reduced. The following gives the algorithm of the DFP method.

Step 1: Choose a starting point $\boldsymbol{x}^{(0)}$, choose a symmetric positive definite $n \times n$ matrix $\mathbf{A}^{(0)}$ (the identity matrix can be selected), set the iteration counter $k = 0$, select a convergence parameter ε, and compute the gradient vector $\boldsymbol{c}^{(0)} = \nabla f(\boldsymbol{x}^{(0)})$.

Step 2: Calculate$\|\boldsymbol{c}^k\|$, if $\|\boldsymbol{c}^k\| < \varepsilon$ stop. Otherwise, continue.

Step 3: Calculate the search direction $\boldsymbol{d}^{(k)} = -\mathbf{A}^{(k)}\boldsymbol{c}^{(k)}$.

Step 4: Calculate the step size α_k that minimizes $f(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)})$ using any line search algorithm.

Step 5: Update the design $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$.

Step 6: Update the matrix $\mathbf{A}^{(k)}$ as $\mathbf{A}^{(k+1)} = \mathbf{A}^{(k)} + \mathbf{B}^{(k)} + \mathbf{C}^{(k)}$, where

$$\mathbf{B}^{(k)} = \frac{\boldsymbol{s}^{(k)}\boldsymbol{s}^{(k)^T}}{\boldsymbol{s}^{(k)}\cdot\boldsymbol{y}^{(k)}}; \mathbf{C}^{(k)} = \frac{-\boldsymbol{z}^{(k)}\boldsymbol{z}^{(k)^T}}{\boldsymbol{y}^{(k)}\cdot\boldsymbol{z}^{(k)}} \quad (3\text{-}16)$$

$$\boldsymbol{s}^{(k)} = \alpha_k \boldsymbol{d}^{(k)}; \boldsymbol{y}^{(k)} = \boldsymbol{c}^{(k+1)} - \boldsymbol{c}^{(k)} \quad (3\text{-}17)$$

$$\boldsymbol{c}^{(k+1)} = \nabla f\left(\boldsymbol{x}^{(k+1)}\right); \boldsymbol{z}^{(k)} = \mathbf{A}^{(k)}\boldsymbol{y}^{(k)} \quad (3\text{-}18)$$

Step 7: Set $k = k + 1$ and go to Step 2.

Note that matrix $\mathbf{A}^{(k)}$ is an approximate of the inverse of the Hessian matrix and will always be positive definite. Since this method does not require the direct calculation of the Hessian and its inverse, it is much more efficient than the Newton's method.

3.9.2 BFGS Method

The BFGS method is another quasi-Newton method. Unlike the DFP method in which an approximate of the inverse of the Hessian is generated and used, the BFGS uses an approximate matrix of the Hessian of the cost function. The following gives the algorithm for DFP method.

Step 1: Choose a starting point $\boldsymbol{x}^{(0)}$, choose a symmetric positive definite $n \times n$ matrix $\mathbf{H}^{(0)}$ (the identity matrix can be selected), set the iteration counter $k = 0$, select a convergence parameter ε, and compute the gradient vector $\boldsymbol{c}^{(0)} = \nabla f(\boldsymbol{x}^{(0)})$.

Step 2: Calculate $\boldsymbol{c}^{(k)} = \nabla f\left(\boldsymbol{x}^{(k)}\right)$. If $\|\boldsymbol{c}^k\| < \varepsilon$, stop. Otherwise, continue.

Step 3: Solve for the search direction $\boldsymbol{d}^{(k)}$the system $\mathbf{H}^{(k)}\boldsymbol{d}^{(k)} = -\boldsymbol{c}^{(k)}$.

Step 4: Calculate the step size α_k that minimizes $f(\boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)})$ using any line search algorithm.

Step 5: Update the design $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$.

Step 6: Update the Hessian approximation $\mathbf{H}^{(k)}$ as $\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{D}^{(k)} + \mathbf{E}^{(k)}$, where $\mathbf{D}^{(k)} = \frac{-\boldsymbol{y}^{(k)}\boldsymbol{y}^{(k)^T}}{\boldsymbol{y}^{(k)}\cdot\boldsymbol{s}^{(k)}}$; $\mathbf{E}^{(k)} = \frac{c^{(k)}c^{(k)^T}}{c^{(k)}\cdot d^{(k)}}$; $\boldsymbol{s}^{(k)} = \alpha_k \boldsymbol{d}^{(k)}$; $\boldsymbol{y}^{(k)} = \boldsymbol{c}^{(k+1)} - \boldsymbol{c}^{(k)}$

$\boldsymbol{c}^{(k+1)} = \nabla f\left(\boldsymbol{x}^{(k+1)}\right)$.

Step 7: Set $k = k + 1$ and go to Step 2.

Note that the above updating formula can keep the approximate Hessian positive definite only when an accurate step size is calculated. Due to numerical errors in the line search algorithm, the approximate Hessian may become singular or indefinite and cause the algorithm to fail. This problem can be resolved by numerical techniques such as decomposing the approximate Hessian by the Cholesky decomposition.

3.10. Sequential Quadratic Programming (SQP)

The sequential quadratic programming (SQP) method is considered as a constrained quasi-Newton method. This method is capable of handling constraints in the optimization problems. This method uses the gradient of the Lagrange function at two points to calculate an approximation of its Hessian. This method is based on CSD and has been extended to include Hessian updating and potential set strategy. Potential set strategy defines a potential constraint index which includes active non-equality constraints and all equality constraints in order to reduce the number of calculations using only active constraints.

In the SQP method, first-order information (i.e., gradient) is used to approximate the second-order information (i.e., Hessian), which in some cases cannot be calculated or is hard to calculate. The method solves an optimization problem by dividing it into a

sequence of sub-problems, each optimizing a quadratic model of the objective subject to linearized constraints similar to those in the CSD method. Before introducing the algorithm, the following vectors are defined to assist the description and discussion:

$$\boldsymbol{s}^{(k)} = \alpha_k \boldsymbol{d}^{(k)} \tag{3-19}$$

$$\boldsymbol{z}^{(k)} = \mathbf{H}^{(k)} \boldsymbol{s}^{(k)} \tag{3-20}$$

where α_k is the step size, $\boldsymbol{d}^{(k)}$ is the search direction vector, and $\mathbf{H}^{(k)}$ is the approximation to the hessian in the kth iteration.

$$y^{(k)} = \nabla L\left(\boldsymbol{x}^{(k+1)}, \boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}\right) - \nabla L\left(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}\right) \tag{3-21}$$

where $y^{(k)}$ is the difference in the gradients of the Lagrange function at two points.

$$\xi_1 = \boldsymbol{s}^{(k)} \cdot \boldsymbol{y}^{(k)} \tag{3-22}$$

$$\xi_2 = \boldsymbol{s}^{(k)} \cdot z^{(k)} \tag{3-23}$$

where $\xi_1$and ξ_2 are used to calculate the maximum constraint violation

$$\theta = 1 \text{ if } \xi_1 \geq 0.2\xi_2, \text{ otherwise } \theta = 0.8\xi_2/(\xi_2 - \xi_1) \tag{3-24}$$

$$\boldsymbol{w}^{(k)} = \theta \boldsymbol{y}^{(k)} + (1-\theta)\boldsymbol{z}^{(k)} \tag{3-25}$$

$$\xi_3 = \boldsymbol{s}^{(k)} \cdot \boldsymbol{w}^{(k)} \tag{3-26}$$

where $\boldsymbol{w}^{(k)}$ and ξ_3 are used to correct matrices $\mathbf{D}^{(k)}$ and $\mathbf{E}^{(k)}$.

The following is the algorithm for SQP method:

Step 1: set k=0 and choose a starting point $\vec{x}^{(0)}$. Select an initial value of the penalty parameter R_0 and two small numbers ε_1 and $\varepsilon_2$that define the permissible constraint violation and convergence parameter respectively. Set the initial approximate of the Hessian as $\mathbf{H}^{(0)} = \mathbf{I}$.

Step 2: At $\boldsymbol{x}^{(k)}$ calculate the objective function, constraint function values, and their gradients. Calculate the maximum constraint violation as $V_k = \max(0; |h_1|, \dots, |h_p|, g_1, \dots, g_m)$. If $k > 0$ update the Hessian of the Lagrange as $\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{D}^{(k)} - \mathbf{E}^{(k)}$, where $\mathbf{D}^{(k)} = (1/\varepsilon_3)\mathbf{w}^{(k)}\mathbf{w}^{(k)^T}$, $\mathbf{E}^{(k)} = (1/\varepsilon_2)\mathbf{z}^{(k)}\mathbf{z}^{(k)^T}$ If $k = 0$ skip updating and go to Step 3.

Step 3: Using the calculations on the previous step define the QP sub problem and solve it as seen on section 3.5, solve it to get the search direction $\boldsymbol{d}^{(k)}$and the Lagrange multipliers $\boldsymbol{v}^{(k)}$ and $\boldsymbol{u}^{(k)}$.

Step 4: Check the stopping criteria $\left\|\boldsymbol{d}^{(k)}\right\| \leq \varepsilon_2$ and the maximum constraint violation $V_k \leq \varepsilon_1$. If both conditions are met then stop, otherwise continue.

Step 5: Calculate $r_k = \sum_{i=1}^{p}\left|v_i^{(k)}\right| + \sum_{i=1}^{m} u_i^{(k)}$. Set $R = \max(R_k, r_k)$.

Step 6: Find α_k using a line search algorithm. Set $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}$.

Step 7: Save the penalty parameter $R_{k+1} = R$. Update the iteration counter k by 1, and go to Step 2.

The SQP method only uses first-order information. It is one of the most commonly used gradient-based methods and is one of the most efficient and reliable numeric algorithms for optimization.

3.11. Solving Multi-objective Optimization Problems

Gradient-based methods were mainly developed for solving single-objective optimization problems. In order to use gradient-based methods to solve multi-objective optimization problems, the problems need to be formulated as single-objective problems by either converting all but one objective functions into constraints or combining all

objectives into one function with a weight coefficient assigned to each objective. The later approach is formally called the weighed sum formulation (WSF), which can be used to obtain multiple solutions of the original problem by varying the weight coefficients of the individual objectives. The WSF is mathematically expressed as

$$F(\vec{x}) = \sum_{n=1}^{k} w_n f_n(\vec{x}) \qquad (3\text{-}27)$$

where *Wn* (*n* = 1 to *k*) is the weighting factor for the n^{th} objective and its value is between zero and one. The sum of all the weight coefficients is always equal to one. In order to obtain a Pareto front using this method, different combinations of the weight coefficients may be used to obtain multiple solutions that are filtered by the non-dominant condition.

Some drawbacks as pointed out by Athan et al. (1996), Das et al. (1997) and Messac et al. (2000) are that the WSF may obtain reliable solutions only for problems with a convex and continuous Pareto front. For problems in which the Pareto front is continuous and has both convex and concave regions, only solutions in the convex region may be obtained.

3.12. Advantages and Disadvantages

Generally speaking, gradient-based methods have the advantage of having relatively small errors and good convergence rate. Since gradient-based methods were primarily developed to solve single-objective optimization problems, strategies such as the WSF are necessary to transform a multi-objective problem into one or several single-objective problems. Another disadvantage of gradient-based methods is that they may not be able to provide adequate solutions for non-convex optimization problems since the necessary conditions for the minimization would not be satisfied. Furthermore, gradient-

based methods are not suitable for discontinuous problems (i.e., discrete optimization problems) due to the lack of continuous gradients.

CHAPTER 4: GENETIC ALGORITHMS

Genetic algorithms (GAs) are heuristic search methods that widely used for their relatively simple handling and easy adaptation to a particular problem, regardless of the numbers of variables, objectives, and constraints. GAs belong to the family of evolutionary algorithms that are based on Charles Darwin's theory of evolution, which states that the most "fit" individuals of each generation are the ones that have greater probability of survival and reproduction. In this chapter, the basic concepts of GAs, their operations, and some of the most commonly used GAs are introduced. The advantages and disadvantages are also discussed.

4.1. Basic Procedure

The GAs start by generating a random population of individuals (or candidate solutions), which are represented by strings of data called chromosomes (or design variables). A fitness value is then assigned to every individual based on evaluation using a fitness function (typically the objective function). The fitness value of an individual is proportional to the probability of the individual being selected for reproduction (i.e., producing new individuals). After reproduction, a mutation operation may be applied at a predefined, small probability emulating the biological genome changes that affects a small portion of the new generation. The "child" population is selected and combined with some of their "parents" to form a new generation and a new iteration starts. The general procedure of a GA is illustrated in Figure 4-1.

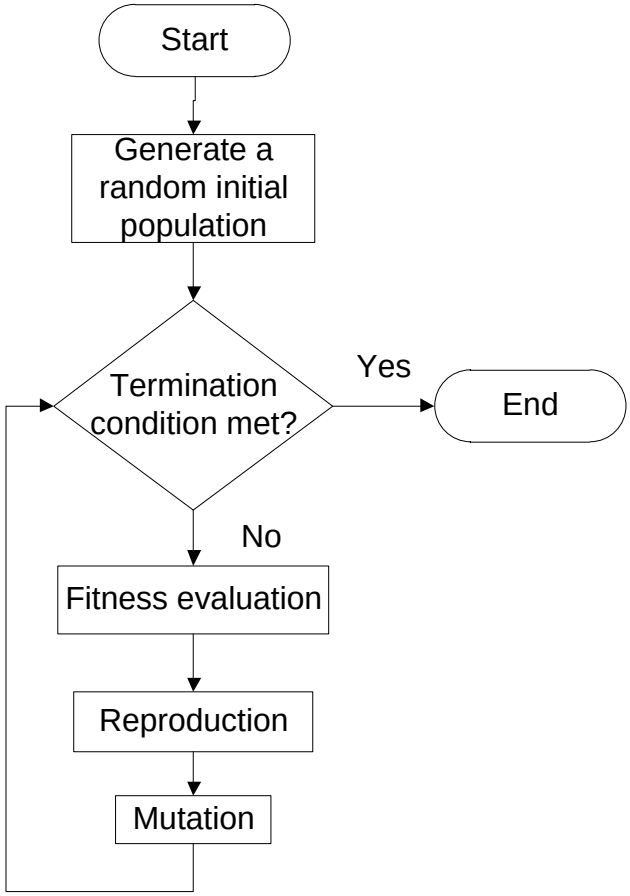

Figure 4.1: The general procedure of a GA

Depending on the application the fitness evaluation method may be changed. This gives the GA flexibility to be used in various applications.

4.1.1 Fitness Evaluation

The evaluation of fitness is a crucial process to a GA, because the individuals with higher fitness values will have a higher probability to be selected for producing new individuals. One way of assigning fitness values is based on the relative size of the objective functions; another way is based on rank of the individuals.

In the first approach of evaluating the fitness of an individual, the sum of all objective functions corresponding to this individual is first calculated after multiplying each objective by a weighting factor. The fitness value is proportional (maximization) or inversely proportional (minimization) to this sum. The weight factors of the objective functions are similar to those used in the WSF.

In the second approach of fitness evaluation, a rank value is assigned to a group of individuals formed by only non-dominant solutions. Since there may exist multiple groups of non-dominant solutions, the rank values of each group will indicate the dominancy of solutions among different groups, as illustrated in Figure 4.2. For a minimization problem, a small rank value indicates better fitness than a large rank value. Note that an individual in a lower ranked group (i.e., with a large rank value) will always be dominated by at least one individual in the immediate higher ranked group. In rank-based fitness evaluation, the fitness value is based on the dominant relationships with others and is not based on the particular objective values of this individual.

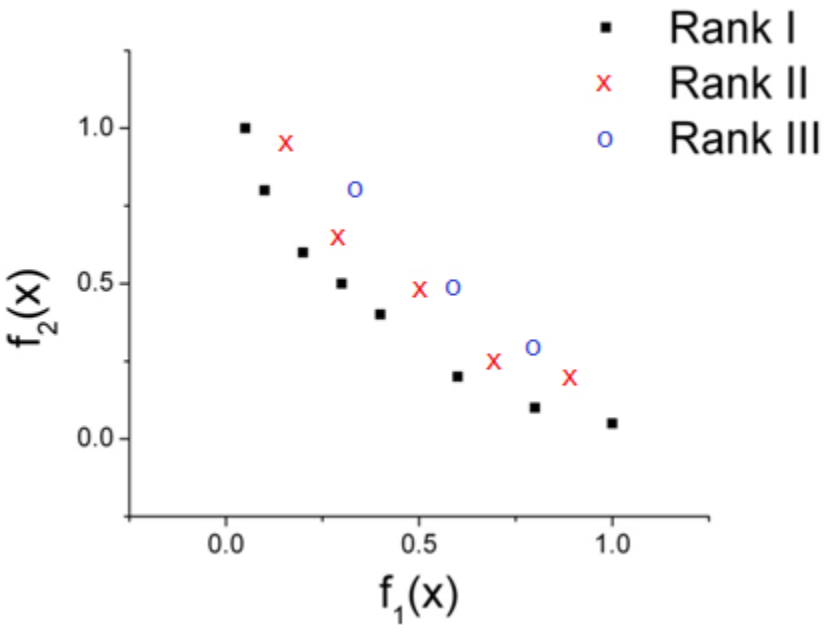

Figure 4.2: Fitness based on ranks

4.1.2 Reproduction

The reproduction process in a GA, similar to that in a biological process, is one that produces new populations with improved fitness while inheriting and maintaining good features from the old generation. The reproduction starts by selecting the parents out of the current population. This is done by selecting each individual at a proportional probability to its fitness. For a random number r, the i^{th} individual is selected if:

$$\sum_{i=0}^{n} fitness(Indiv_i) < r \leq \sum_{i=0}^{n+1} fitness(Indiv_{i+1}) \qquad (4\text{-}1)$$

After choosing the parents, a crossover point is selected where the two individuals are going to be combined to form two new individuals called “children.” This crossover point is selected at random. The above mentioned single-point crossover procedure is shown in Figure 4-3.

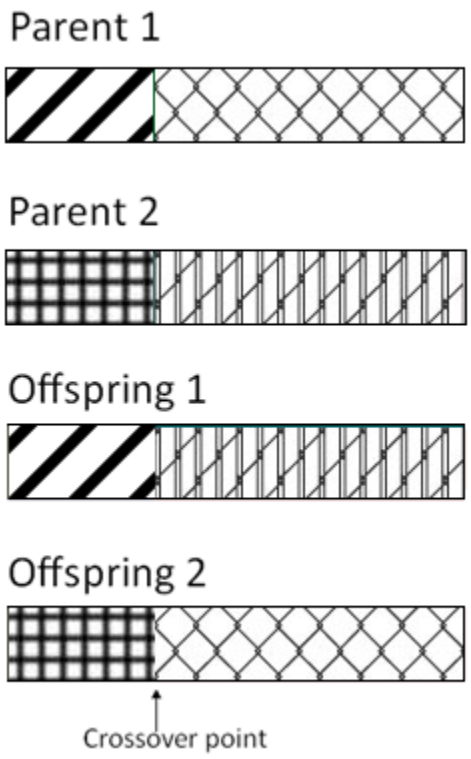

Figure 4.3: Single-point crossover

This procedure is applied to each pair of parents. It is important to note that a single individual may be selected to be a parent multiple times; it could even be selected as Parent 1 and 2 in the same reproduction operation, depending on the selection algorithm.

4.1.3 Mutation

The mutation operation is a low probability operation in which a small fraction of the individuals are subject to modification based on the predetermined mutation rate. When an individual is selected for mutation, a single bit of the binary string representing the individual is randomly selected and changed from ‘0’ to ‘1’ or from ‘1’ to ‘0’. This

operation is useful in maintaining the diversity of the solution set and discovering new solution points which may not be discovered via the reproduction operations.

4.1.4 Termination Condition

There is no standard termination condition for a GA; the iteration process is usually terminated at reaching a maximum number of generations. Other options for the termination conditions can be based on the evolution time (i.e., a maximum time is set for the algorithm execution) or based on the fitness value (i.e., the fitness of the best individual of the current population is better than a predefined threshold value).

4.2. Commonly used Genetic Algorithms

There are a large number of GAs developed to solve multi-objective optimization problems. Some of the commonly used GAs includes MOGA, NSGA, NSGA-II, MOBES and MOSES; each one of these implementations has additional features over the basic GA in order to achieve better performance.

4.2.1 MOGA

The multi-objective genetic algorithm (MOGA) was developed by Murata et al. (1995); it used a combined objective of the weighted individual objectives in the selection procedure. The weight coefficients are not constant, but randomly specified for every individual, to keep the search direction not fixed. In addition, an elite preserve strategy is introduced such that a certain number of individuals are randomly selected from each generation and preserved for the next generation in order to keep the variety of each population throughout the algorithm. Figure 4-4 shows the flowchart of MOGA.

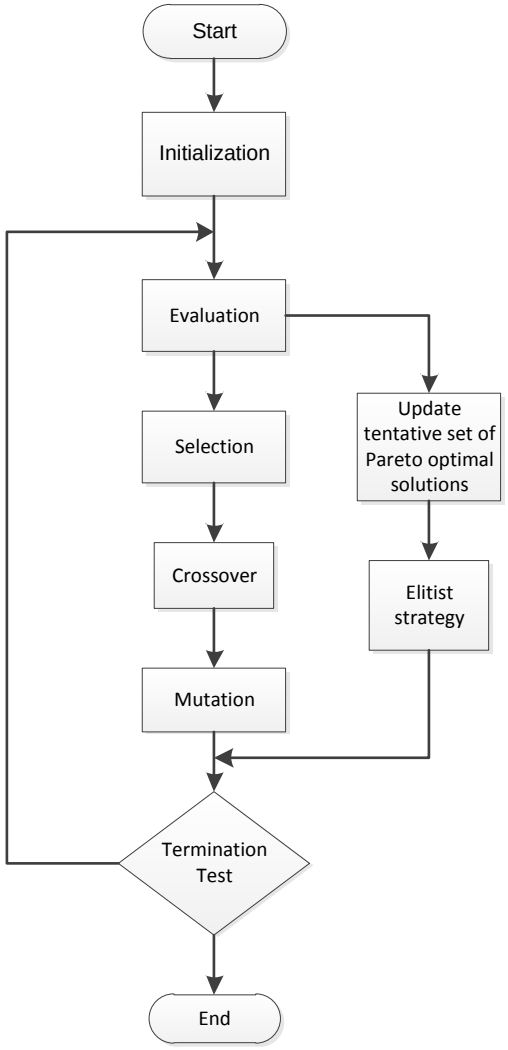

Figure 4.4: Flowchart of MOGA

Using test problems, the MOGA was shown to be effective in solving multi-objective optimization problems. The solutions by the MOGA showed that elitism was an effective way of keeping some of the “best” individuals through generations to avoid the tendency for a future generation not converging to the true Pareto front.

4.2.2 NSGA

The nondominated sorting genetic algorithm (NSGA), introduced by Srinivas et al. (1995), uses non-dominant relationships among individuals to classify and rank the individuals to determine their fitness. The individuals in the same front (i.e., a group of nondominated individuals with the same rank) are assigned the same fitness value in order to give them the same probability of being selected for reproduction. In order to obtain a more diverse population, the individuals are classified and shared with their dummy fitness values. Sharing is achieved by dividing the original fitness value by a quantity proportional to the number of individuals around it. Figure 4-5 shows the flowchart of NSGA.

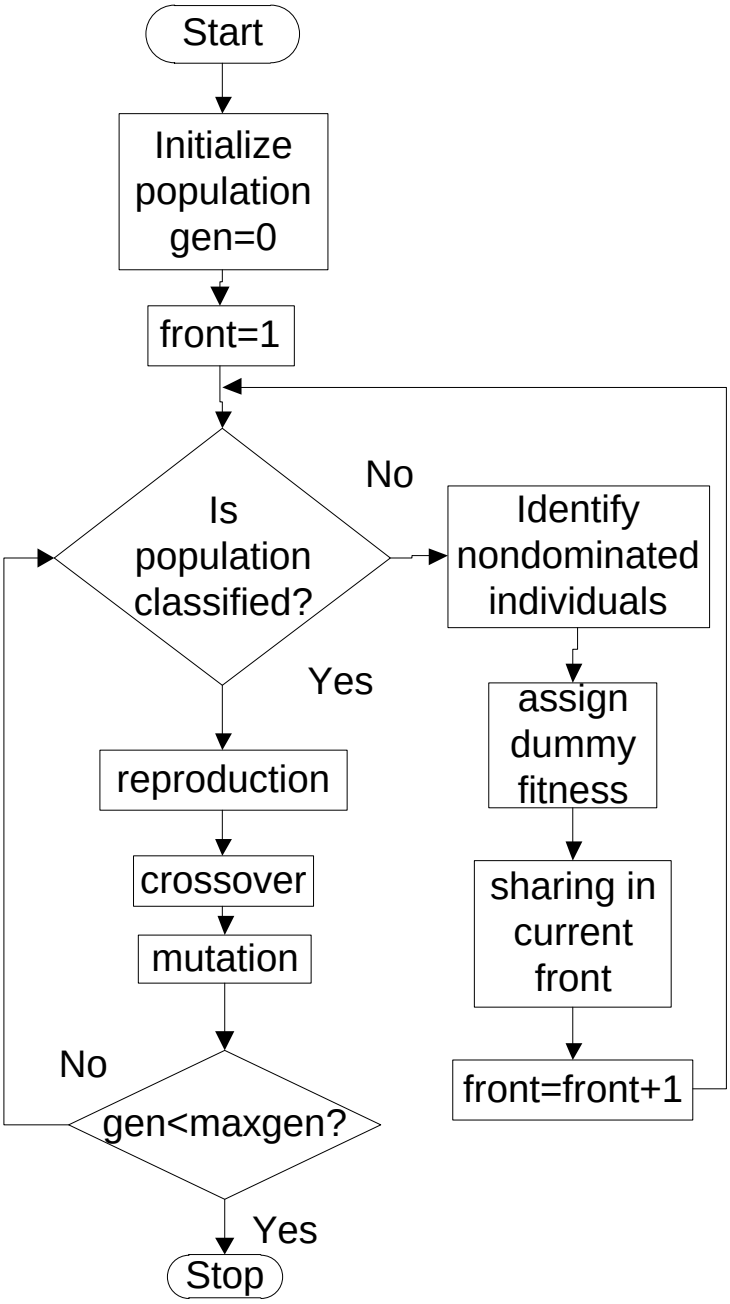

Figure 4.5: NSGA flowchart

The NSGA was shown to be able to maintain an uniform reproductive potential across nondominated individuals and obtain multiple Pareto optimal solutions, which were very useful to decision makers or designers.

4.2.3 NSGA-II

The nondominated sorting genetic algorithm II (NSGA-II) was developed by Deb (2002) to improve the previously developed NSGA based on some of the criticism on NSGA such as high computational complexity, lack of elitism, and lack of capability of specifying a sharing parameter.

The computational complexity of NSGA is $O(MN^3)$ where M is the number of objectives and N is the population size. This means that NSGA is computationally very expensive for large population sizes. In NSGA-II, the nondominated sorting alogorithm was improved by maintaining two entities for each individual: a domination count n_p that is the number of solutions dominating solution p and S_p a set of solutions dominated by solution p. NSGA-II has a complexity of $O(MN^2)$ and is given by the following algorithm.

For each $p \in P$
  $S_p = 0$
$n_p = 0$
for each $q \in P$
  If $p \prec q$ *then* If p dominates q
    $S_p = S_p \cup \{q\}$ Add q to the set of solutions dominated by p
  else if $q \prec p$ *then*
    $n_p = n_p + 1$
if $n_p = 0$ *then*
  $p_{rank} = 1$

```
        F_1 = F_1 ∪ {p}
i = 1                                   Initialize the front counter
    while F_i ≠ 0
      Q = 0                             Used to store the members of the next front
      for each p ϵ F_i
        for each q ϵ S_p
          n_q = n_q − 1
          If n_q = 0 then
            q_rank = i + 1              q belongs to the next front
            Q = Q ∪ {q}
    i = i + 1
    F_i = Q
```

NSGA uses the sharing parameter σ_{share} that is specified to maintain a good spread of solutions in the obtained set. In NSGA-II, the sharing paremeter is replaced with the crowding distance, which is a proximity measure of the distances between a solution and the rest of solutions in the same front. The crowding distance is used to indicate the spread of solutions but does not require any user-defined parameter for maintaining diversity among population members. The value of a crowding distance is equal to the normalized difference in function values of the two adjacent solutions in the same front. Figure 4-6 illustrates the computation of the crowding distance for the solution inside the rectangular box formed by the two adjacent solutions.

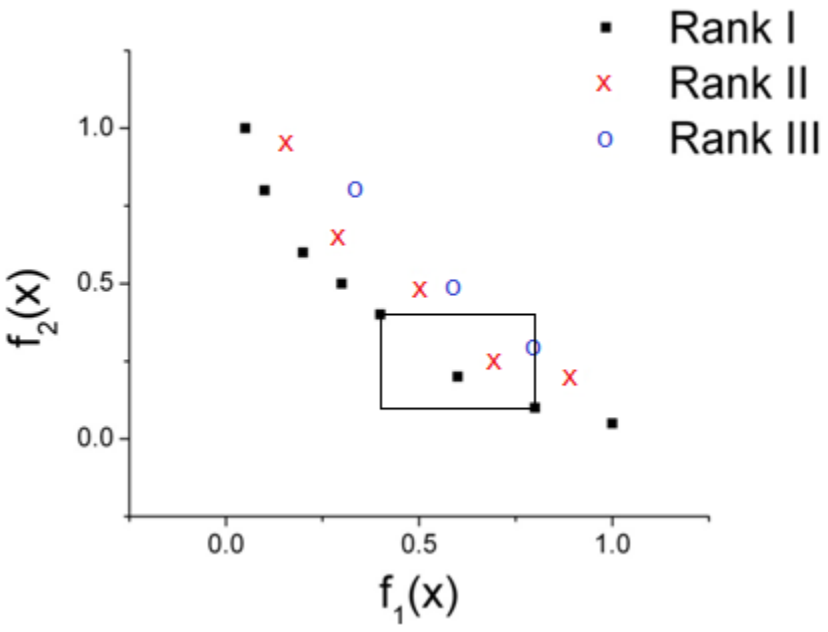

Figure 4.6: Crowding distance calculation

Fang et al. (2008) proposed a new nondominated sorting algorithm that could be used in NSGA-II. This algorithm uses a divide-and-conquer mechanism and maintains a dominance tree to reduce the number of comparisons of objective values for identifying nondominated solutions. This algorithm has a complexity of $O\left(MNlog(N)\right)$ and thus is more efficient than the algorithm in NSGA-II.

4.2.4 NCGA

Watanabe et al. (2002) developed the neighborhood cultivation genetic algorithm (NCGA) that used a neighborhood crossover mechanism to pair up neighboring individuals for exploring the objective space around the obtained solutions. This process makes the algorithm explore the section of the objective space around the parent individuals. The NCGA was compared to the strength of Pareto evolutionary algorithm II

(SPEA2) and NSGA-II; it was shown to possess the same capability of finding Pareto optimum solutions.

4.3. Advantages and Disadvantages of GAs

GAs are the preferred method for solving MOOPs, because they are relatively easy to implement and versatile in dealing with large numbers of objectives and variables. The downside of these algorithms is that their convergence rate decreases as the iterations number increases, since they use only zero-order information. Another issue with the GAs is that, when they converge to the true Pareto front, the solutions may start "vibrating" due to the nature of genetic operations. Furthermore, GAs do not have a standard termination condition and typically use a fixed number of generations as the stopping criterion, which may not ensure to obtain a good solution set for certain problems.

CHAPTER 5: HYBRID OPTIMIZATION ALGORITHMS

A hybrid optimization algorithm is one that is composed of different optimization algorithms in order to take the advantage of each method to have improved solution accuracy and/or computational efficiency in solving optimization problems. After a close examination of the gradient-based methods and GAs, a hybrid algorithm is proposed to combine the two types of methods and use them at different stages of the optimization process. Different hybrid strategies are investigated to achieve the best computational efficiency and solution accuracy. In this study, the nondominated sorting genetic algorithm II (NSGA-II) and the feasible sequential quadratic programming (FSQP) are selected to study different hybrid strategies.

In this chapter, the performance metrics for evaluating the hybrid methods are first defined. Three hybrid strategies are then investigated and the best one is identified. Three benchmark problems are solved: the Schaffer problem, ZDT-1 problem and ZDT-2 problem. The Schaffer problem was developed by Schaffer (1985); it has one design variable and two objective functions that form a convex Pareto front. The ZDT-1 and ZDT-2 problems were both developed by Zitzler et al. (2000). The ZDT-1 problem has thirty design variables and two objectives forming a convex Pareto front. The ZDT-2 problem is the same as the ZDT-1 problem except that the two objective functions of ZDT-2 form a non-convex (or concave) Pareto front.

5.1. Performance Metrics

The performance metrics are used to evaluate the hybrid algorithms and compare their performance to that of the non-hybrid algorithms, i.e., FSQP and NSGA-II. The root mean square error (RMSE), Piling Factor (PF), and execution time are the performance metrics of this study. The RMSE and PF are used to determine the quality of the solutions and the execution time is used to compare the computational efficiency.

5.1.1 Root Mean Square Error (RMSE)

The RMSE is a measure of the differences between the solutions of an optimization algorithm and the true Pareto front. The RMSE values can only be calculated when the true Pareto front is known and thus it is only used for benchmark problems. Since there is no standard definition of the distance between a point (e.g., a solution point in the objective space) and a curve (e.g., the true Pareto front), a distance needs to be defined in order to calculate the RMSE values. As a first step, the points on the Pareto front are located using the objective values of the solution point, as illustrated in Figure 5-1.

After the points on the Pareto front are determined, a line is traced and its slope is calculated. A perpendicular line is drawn through the solution point and this direction is used to find an interception with the Pareto front. The points $[f(\vec{x}_1)^*, f(\vec{x}_2)]$ and $[f(\vec{x}_1), f(\vec{x}_2)^*]$ are calculated using the equitation of the Pareto front. After calculating the slope of a perpendicular line, the interception point on the Pareto front is obtained. The distance between the solution point and the point on the Pareto front is defined as the difference between the solution point and the corresponding true optimal solution on the Pareto front (see Figure 5-2).

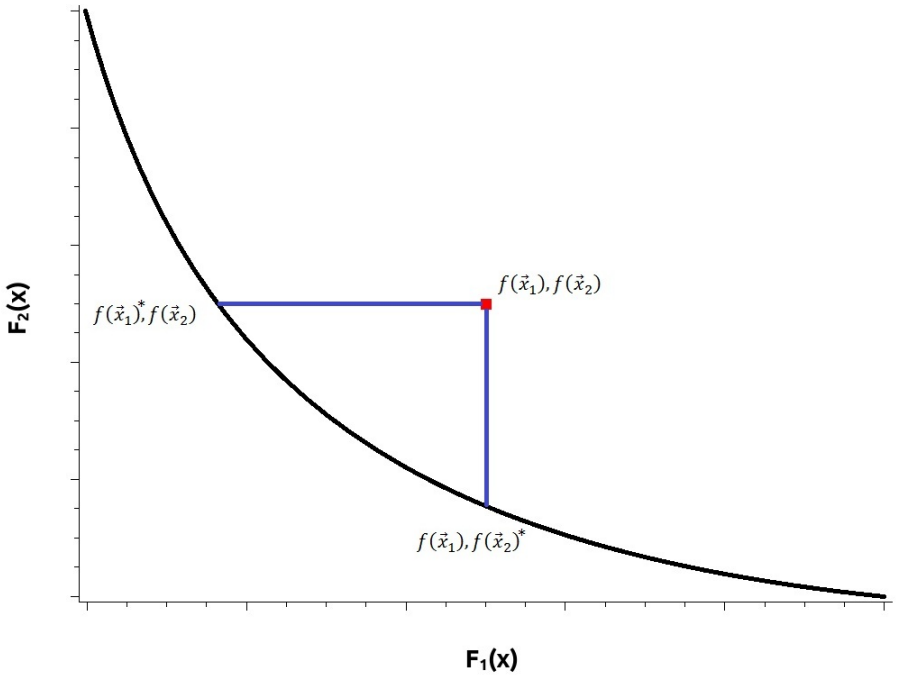

Figure 5.1 Points on the Pareto front corresponding to a solution point

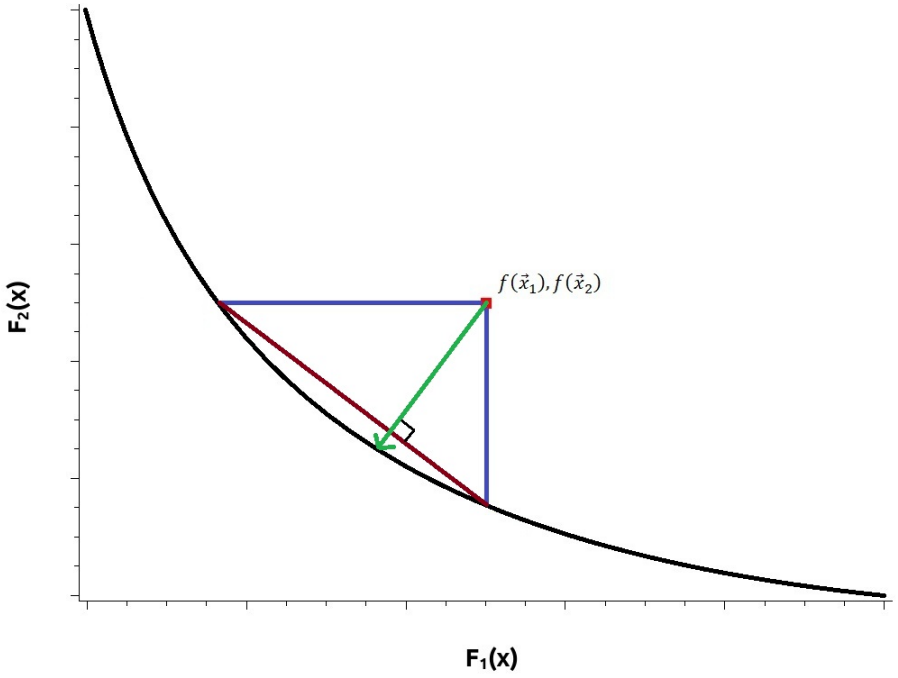

Figure 5.2: Definition of the distance between a solution point to the Pareto front

If the MOOP has more than two objectives, the line shown in Figure 5-2 becomes a surface and the slope of the surface becomes the gradient of this surface. After obtaining the perpendicular direction the distance is calculated. From this equation a point $\left(f_1(\vec{x}^*), f_2(\vec{x}^*) \ldots f_n(\vec{x}^*)\right)$ is obtained and the difference between this point and the solution point is calculated. After obtaining this distance, the RSME value is calculated as follows.

$$RSME = \sqrt{\frac{\sum_{i=1}^{n} {d_i}^2}{n}} \tag{5-1}$$

5.1.2 Piling Factor (PF)

The PF is a metric used to compare the distribution of the solution points in or close to the Pareto front. The PF is defined as the maximum value of the sum of distances between a solution point and the adjacent solution points in the objective space, as given by:

$$PF = \max \langle \left\| \vec{f}(\boldsymbol{x}_n) - \vec{f}(\boldsymbol{x}_{n+1}) \right\| \Big|_{n=1}^{m} \rangle \tag{5-2}$$

where *m* is the number of solutions in the objective space. The PF reflects the distribution of a solution set in the objective space: a smaller PF value indicates a more homogeneous distribution than a larger PF value, as illustrated in Figures 5-3 and 5-4.

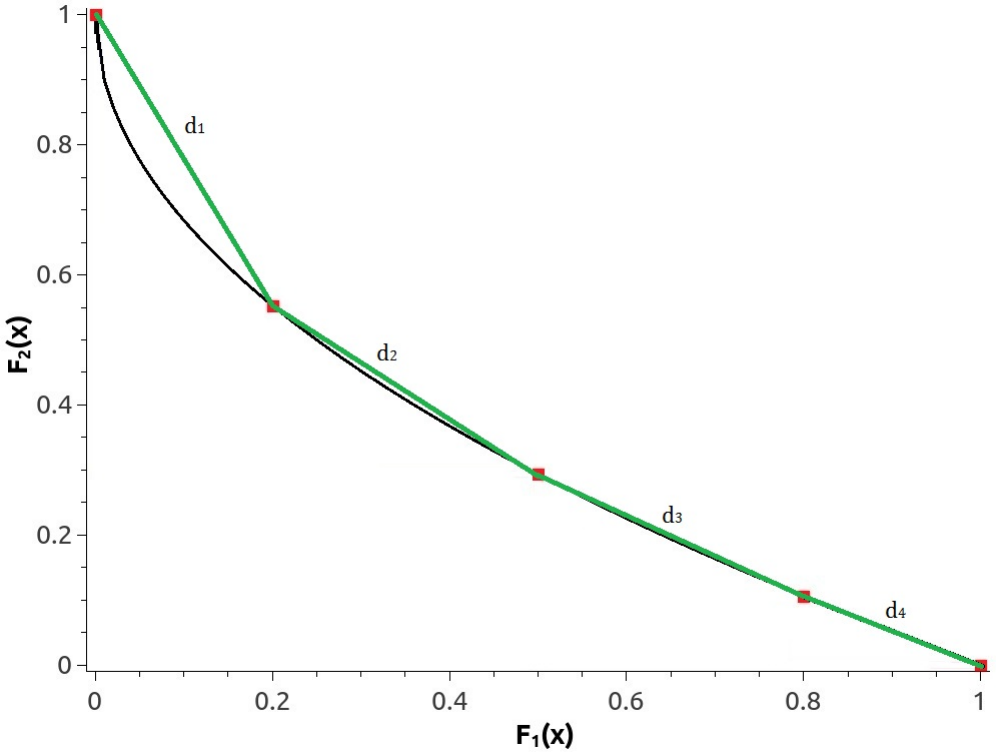

Figure 5.3: A solution set with a low of piling factor

When the distribution is more homogeneous over the Pareto front, the maximum distance between adjacent points is smaller than that when some points are clustered and others are much farther away. This is because the largest distance between two adjacent solution points determines the PF value. It is important to note that the minimum number of points needed to calculate the PF is two. The PF is considered infinity if there are two or fewer points in the solution set.

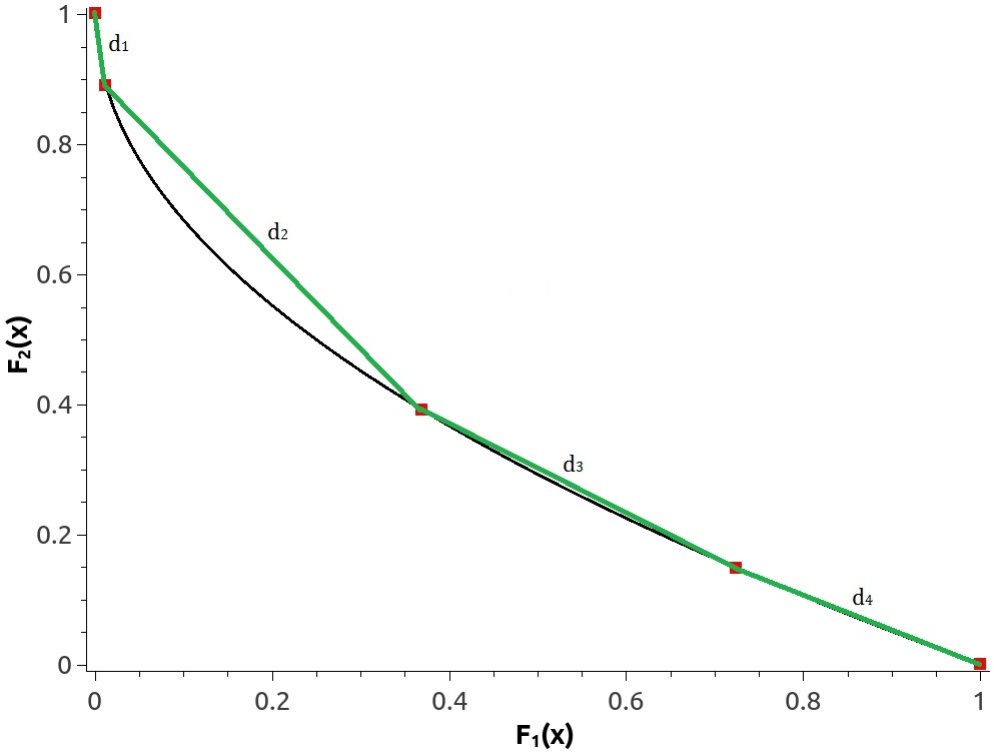

Figure 5.4: A solution set with a high piling factor

5.1.3 Execution Time

The execution time is defined as the time that takes the algorithm to be executed from beginning to obtaining the last solution to the problem. This value is given by the software with a resolution of 0.01 seconds.

5.2. The First Hybrid Stragety

The first hybrid strategy is a brute-force approach mainly used as a test algorithm in order to determine the feasibility that NSGA-II and FSQP can be combined and work together. In this strategy, the NSGA-II is first executed for a certain number of generations and the results are used as the initial points for FSQP, which uses a weighted sum formulation (WSF) on each of the initial points. The computational cost of this

algorithm is expected to be high, because it is simply a combination of the time of both individual methods.

5.2.1 Algorithm of the First Strategy

This approach uses the NSGA-II as a first layer in order to get an initial set of solutions that are closer to the Pareto front than a set of randomly generated initial points. These solution points are then used by FSQP as initial points to obtain the final solutions to the problem.

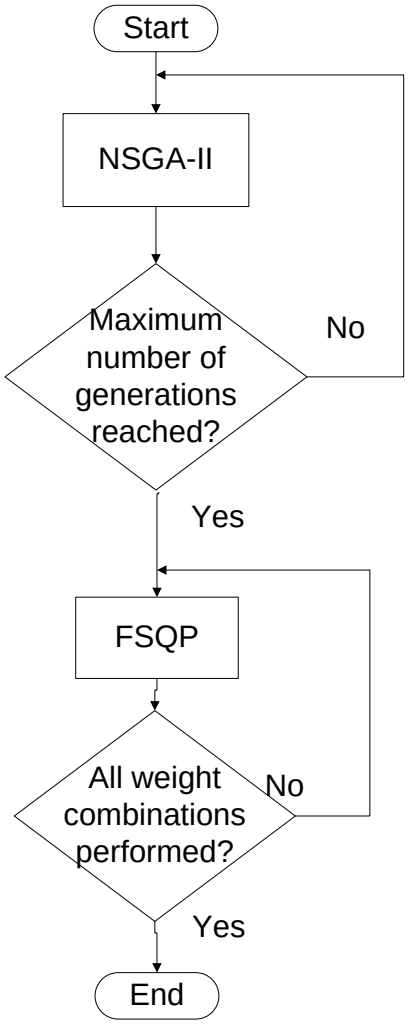

Figure 5.5: First hybrid algorithm flowchart

It is important to note that FSQP is run as many times as the points yielded by NSGA-II, i.e. if there are 20 points as an output to NSGA-II, FSQP is executed 20 times, each time with one initial point and a WSF to obtain a set of solutions.

5.2.2 Test Problems and Results of the First Strategy

In order to test the performance of the algorithm, three benchmark problems, Schaffer, ZDT-1 and ZDT-2, are chosen, because these problems can test the ability of an algorithm to identify nondominated solution points.

5.2.2.1. The Schaffer Problem

This MOOP was introduced by Schaffer (1985); it has two objective functions and one design variable, as given by:

$$\text{Min} \quad \left(f_1(x), f_2(x)\right) \tag{5-3}$$

$$f_1(x) = x^2 \tag{5-4}$$

$$f_2(x) = (x-2)^2 \tag{5-5}$$

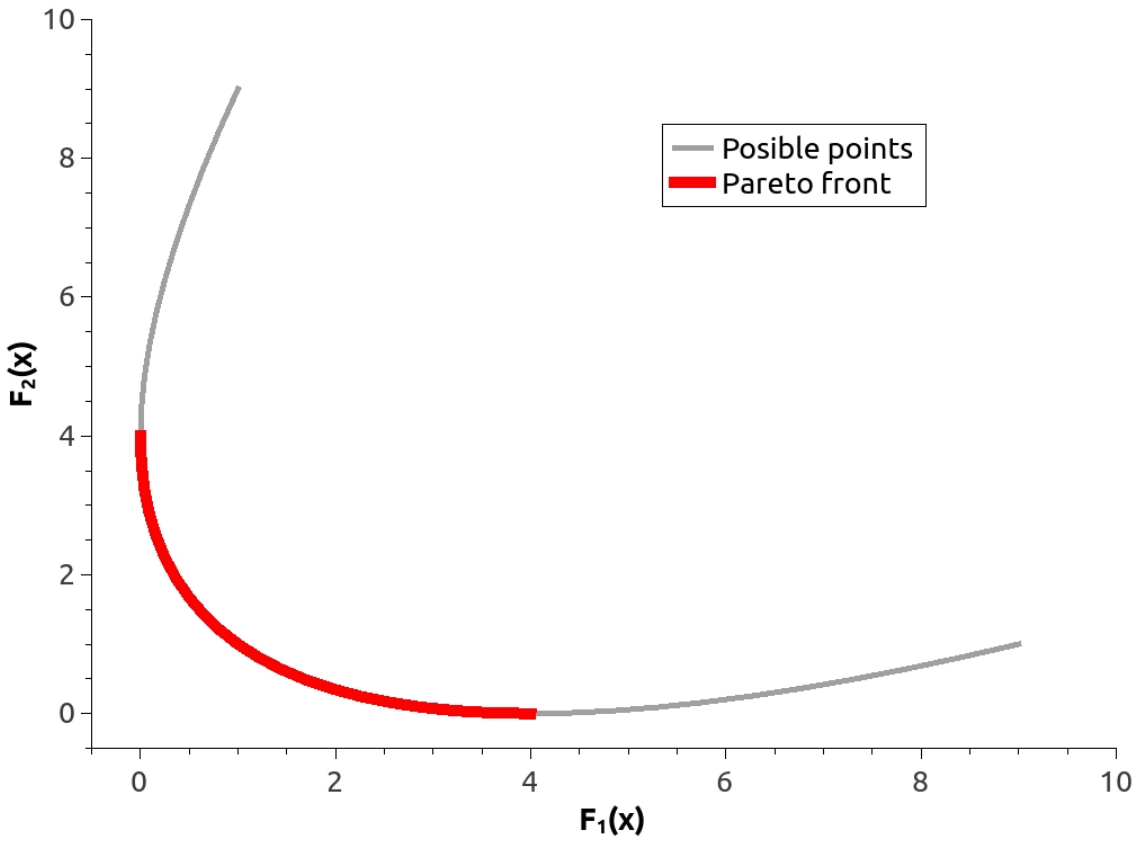

Figure 5.6: The Pareto front and feasible domain of the Schaffer problem

The Pareto front of the Schaffer problem is obtained when the design variable is in the range of $0 \leq x \leq 2$, as shown in Figure 5-6. The expected solution points for this problem is set to 50, which is set as a guideline in order to compare the different methods in a fair manner. The input parameters for FSQP are: number of initial points (50), minimum weight (0.02) for WSF, and weight increase (0.02) for WSF. The results of the FSQP are shown in Figure 5-7 and the performance metrics are given in Table 5-1. It should be noted that the formulation of this problem does not allow feasible points to be off the Pareto front, hence the RMSE value is always zero for all the three methods. Also,

the simulation times for FSQP and NSGA-II are less than the resolution that the software can record; therefore their execution times are shown as 0 second.

Table 5-1: Schaffer problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0	0
PF	0.15	0.69	0.15
Execution time	0s	0s	0.02s
Number of solutions	49	50	49

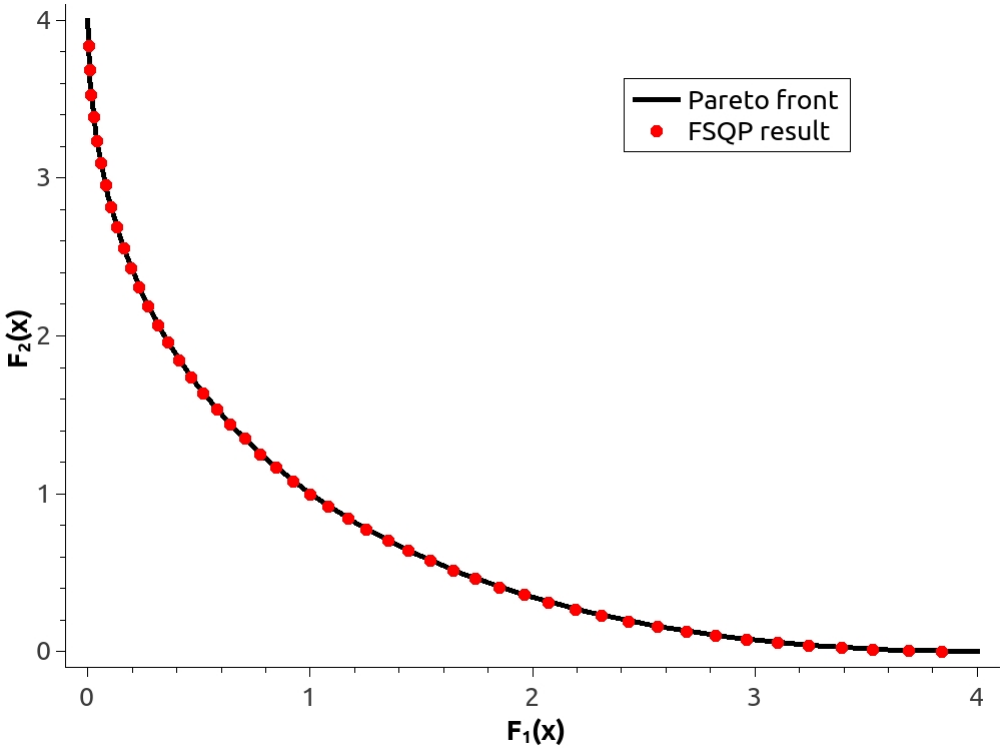

Figure 5.7: The Schaffer problem solved by FSQP

The input parameters for NSGA-II are the number of initial points (50), the number of generations (300), crossover rate (0.9), and mutation rate (0.01). For the Schaffer problem, the best PF value is obtained at 300 generations, as can be seen in Figure 5-8; therefore, 300 generations are specified for the NSGA-II. The results of NSGA-II are shown in Figure 5-9 and the performance metrics are also given in Table 5-1.

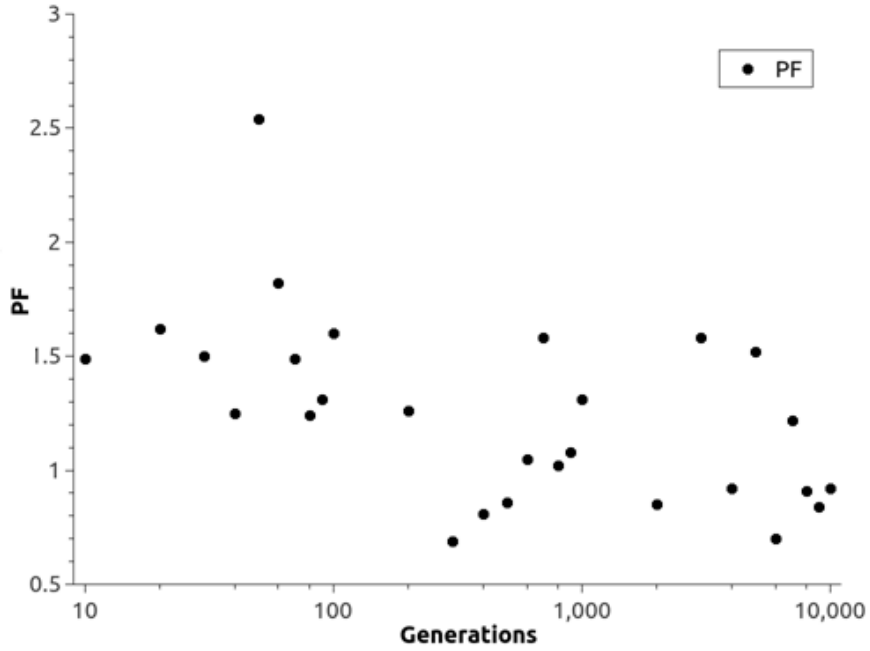

Figure 5.8: The PF vs. number of generations for Schaffer problem

.

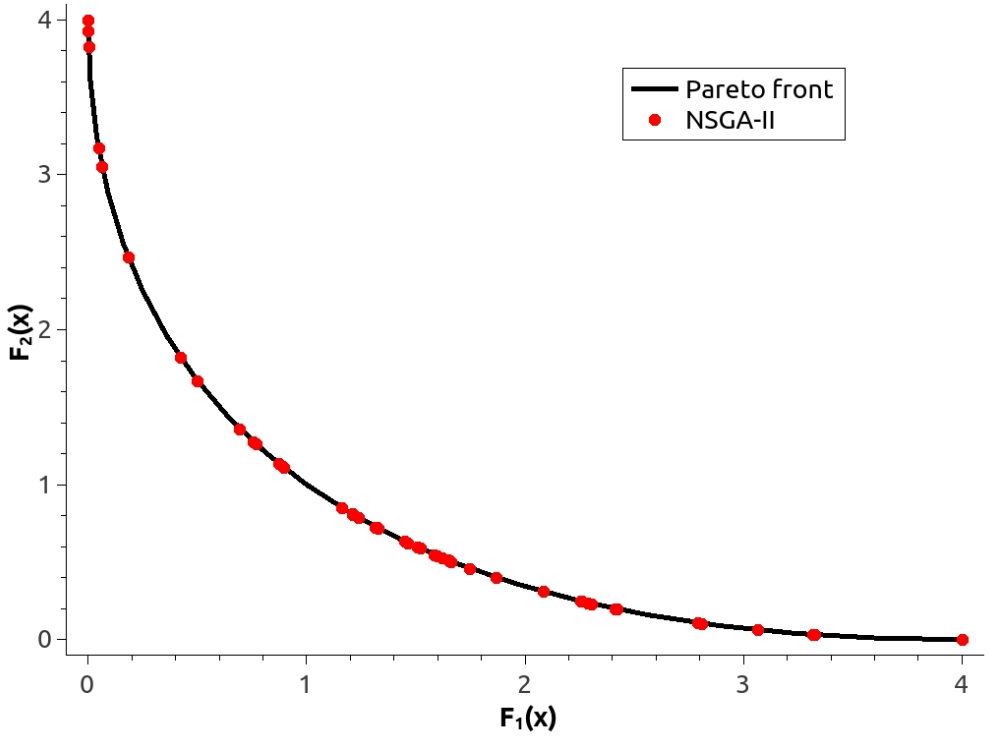

Figure 5.9: The Schaffer problem solved by NSGA-II

For the first hybrid algorithm, the input parameters are the number of initial points (50) for both the NSGA-II and FSQP, and the maximum number of generations (10) only for the NSGA-II. The results of the hybrid algorithm are shown in Figure 5-10, and the performance metrics are given in Table 5-1.

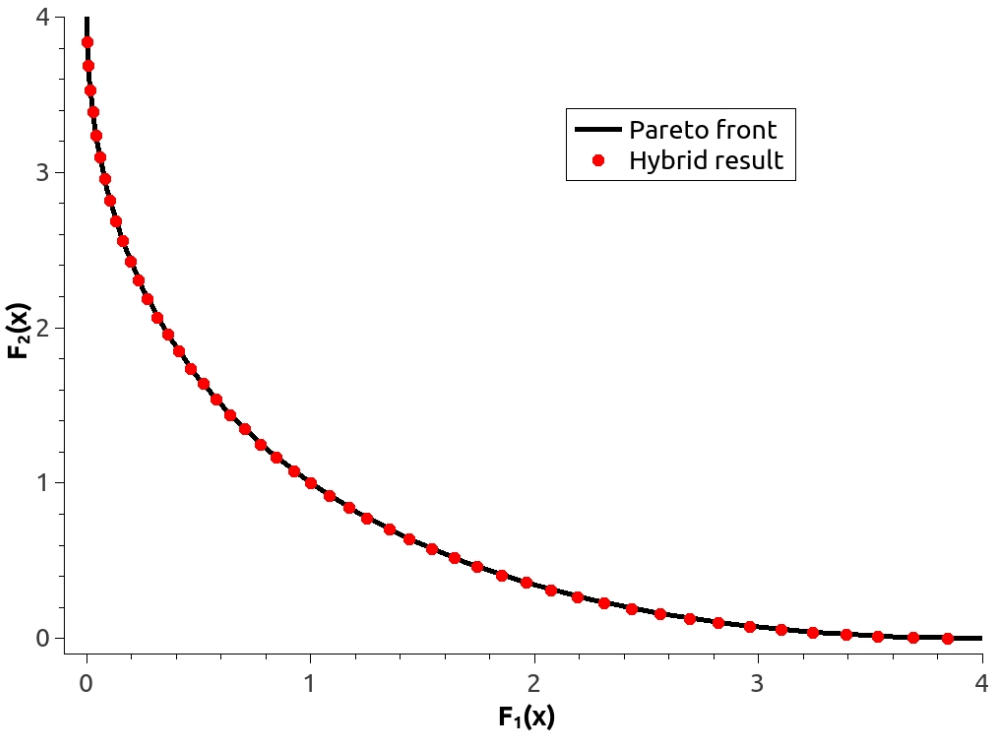

Figure 5.10: Schaffer problem solved by the first hybrid algorithm

It can be seen from Table 5-1 that the PF of NSGA-II is four times larger than that of FSQP and the hybrid algorithm due to its uneven distribution on the Pareto front. For both FSQP and the first hybrid algorithm, there is one solution point lost due to the nondominance check performed on all the final solutions. Regarding the execution time, it can be seen that FSQP and NSGA-II are executed in less than 0.01 seconds, while the first hybrid algorithm takes more than double the time of its counterparts due to the brute-force approach. Nevertheless, the hybrid algorithm is able to obtain solutions on the Pareto front that are difficult for the GA. This indicates that the hybrid strategy provides a viable means to improve the solutions of an individual algorithm (e.g., the NSGA-II in this case).

5.2.2.2. The ZDT-1 Problem

This problem was introduced by Zitzler et al. (2000) with two objective functions and thirty design variables. The ZDT-1 problem is stated as:

$$\text{Min} \quad \left(f_1(x), f_2(x)\right) \tag{5-6}$$

$$f_1(\vec{x}) = x_1 \tag{5-7}$$

$$f_2(\vec{x}) = 1 - \sqrt{x_1 / g(\vec{x})} \tag{5-8}$$

where

$$g(\vec{x}) = 1 + 9 \sum_{i=2}^{n} x_i / (n-1), \; n = 30 \tag{5-9}$$

The Pareto front of the ZDT-1 problem is obtained when $g(\vec{x}) = 1$, which gives $f_2(\vec{x}) = 1 - \sqrt{f_1(\vec{x})}$ as shown in Figure 5-11.

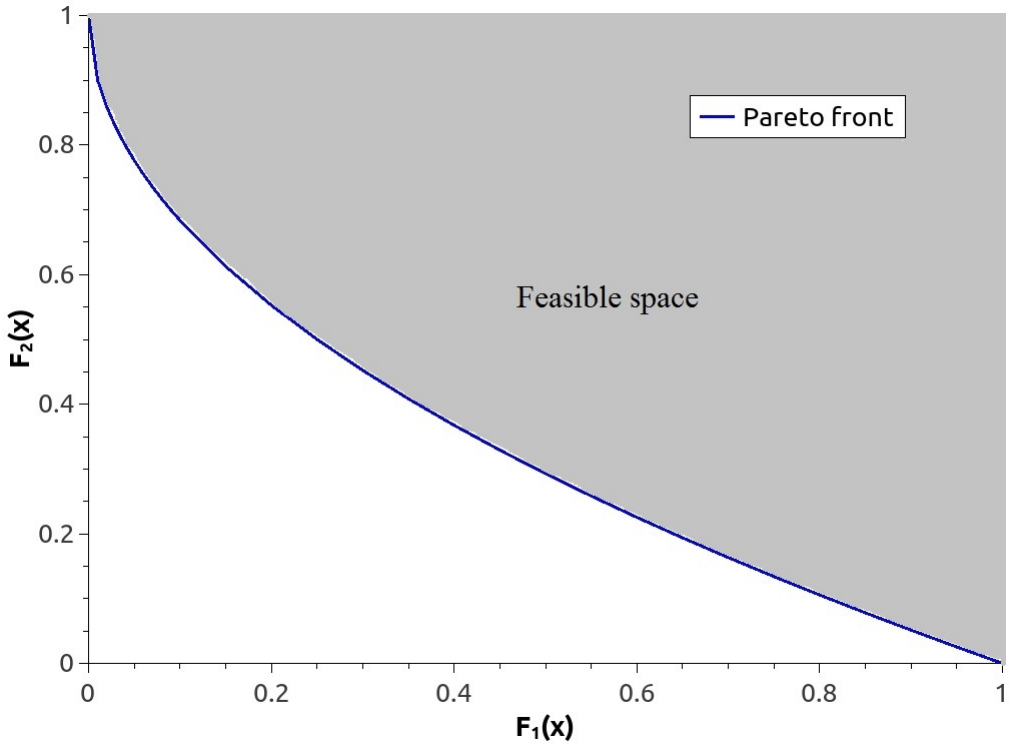

Figure 5.11: The Pareto front and feasible domain of ZDT-1 problem

Similar to the previous problem, the expected number of solutions is 50. The input parameters for FSQP are: number of initial points (50), minimum weight (0.02), and weight increase (0.02). The results of FSQP are shown in Figure 5-12 and the performance metrics are given in Table 5-2. In this case, the problem gives an RMSE value of 1.08×10^{-5}, a piling factor of 0.235, an execution time of 16 sec, and 31 final solutions. The reduction in the number of final solutions is due to the nondominance filter that is used in the algorithm of FSQP, because not all the solutions from the WSF are nondominated solutions.

Table 5-2: ZDT-1 problem results

	FSQP	NSGA-II	Hybrid
RMSE	1.93E-06	8.43E-04	2.84E-07
PF	2.95	0.16	0.2
Execution time	16s	1.25s	8.260s
Number of solutions	31	50	33

.

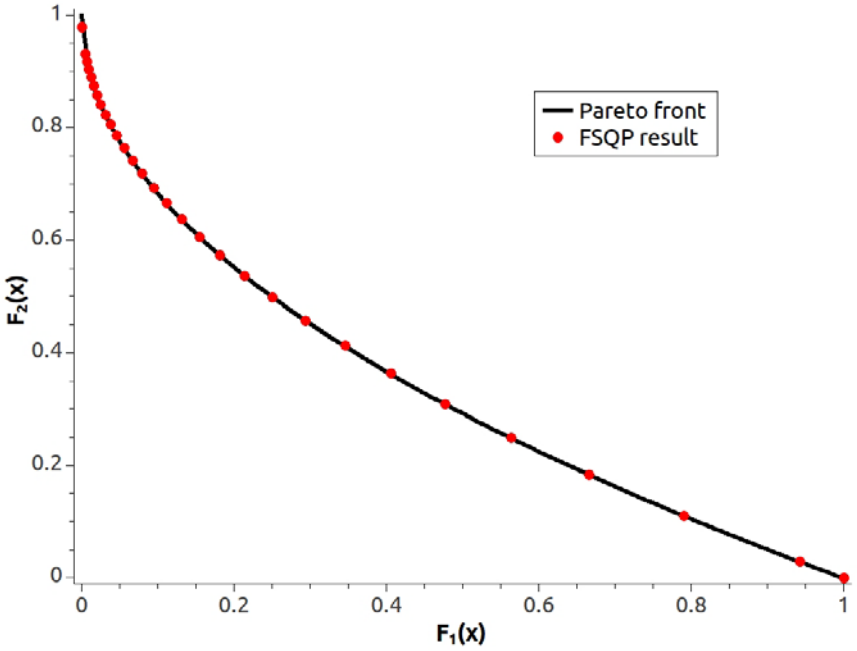

Figure 5.12: The ZDT-1 problem solved by FSQP

Using NSGA-II, the ZDT-1 problem is solved with (50) initial points, crossover rate (0.9), and mutation rate (0.01). In order to select the adequate number of generations for which the results of NSGA-II have the best PF, the algorithm is run with different numbers of generations with the PFs shown in Figure 5-13.

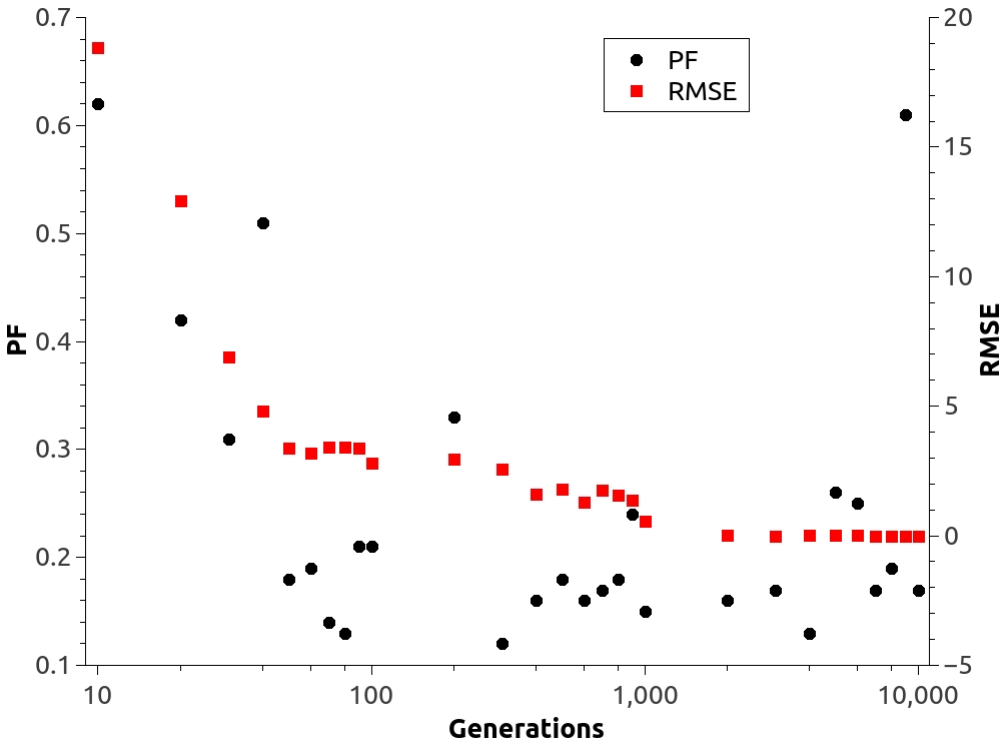

Figure 5.13: The PF and RMSE vs. number of generations

for the ZDT-1 problem solved by NSGA-II

It can be seen from Figure 5-13 that the least number of generations to have the best PF is 2000. The results of NSGA-II are shown in Figure 5-14 and the performance metrics are also summarized in Table 5-2.

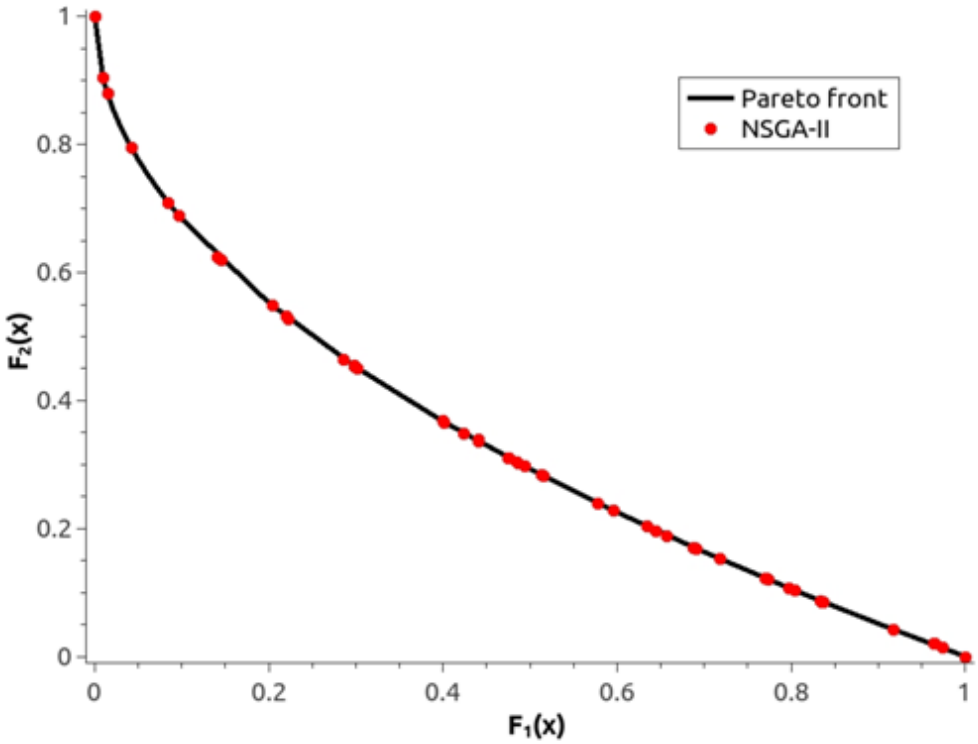

Figure 5.14: The ZDT-1 problem solved by NSGA-II

The parameters for this problem on the first hybrid are the number of initial points (50) for NSGA-II and FSQP, and the maximum number of generations (10) for the NSGA-II section. The results of the algorithm are shown in Figure 5-15, and the results are given in Table 5-2.

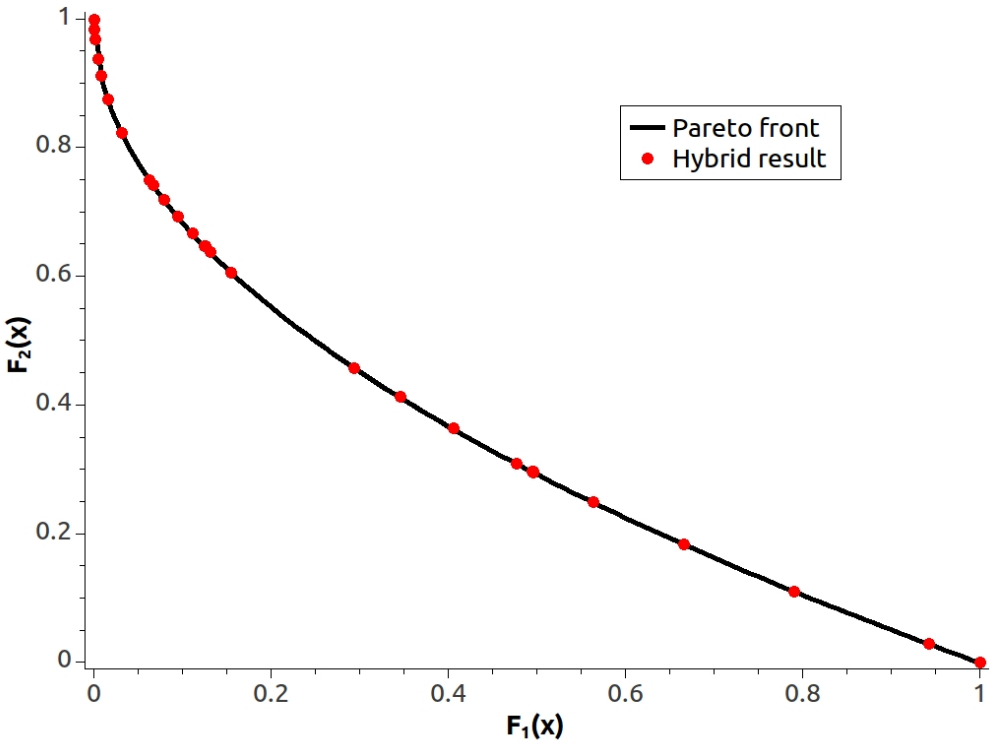

Figure 5.15: The ZDT-1 problem solved by the first hybrid algorithm

It can be seen from the results in Table 5-2 that the hybrid algorithm has the lowest RMSE value. The NSGA-II has slightly better PF than the hybrid algorithm and both have much smaller PFs than the FSQP. The fastest algorithm for the ZDT-1 problem is NSGA-II followed by the hybrid algorithm that has eight times more than the NSGA-II. The FSQP takes twice as much time as the hybrid algorithm. Finally, the number of solutions from the hybrid algorithm and FSQP are 33 and 31 solutions, respectively, while NSGA-II has 50 solutions. Similar to the case of the Schaffer problem, the reduced number of solutions in the hybrid algorithm and FSQP is due to the nondominance check on the final solutions obtained by the WSF.

5.2.2.3. The ZDT-2 Problem

This problem was introduced by Zitzler et al (2000). It has two objective functions to be minimized, and thirty design variables. The ZDT-2 problem is stated as:

Min $$\left(f_1(x), f_2(x)\right) \quad (5\text{-}10)$$

$$f_1(\vec{x}) = x_1 \quad (5\text{-}11)$$

$$f_2(\vec{x}) = 1 - \left({x_1}/{g(\vec{x})}\right)^2 \quad (5\text{-}12)$$

where $$g(\vec{x}) = 1 + 9\,{\sum_{i=2}^{n} x_i}/{(n-1)},\ n = 30 \quad (5\text{-}13)$$

The Pareto front is obtained when $g(x) = 1$, the feasible space and the Pareto front are shown in Figure 5-16.

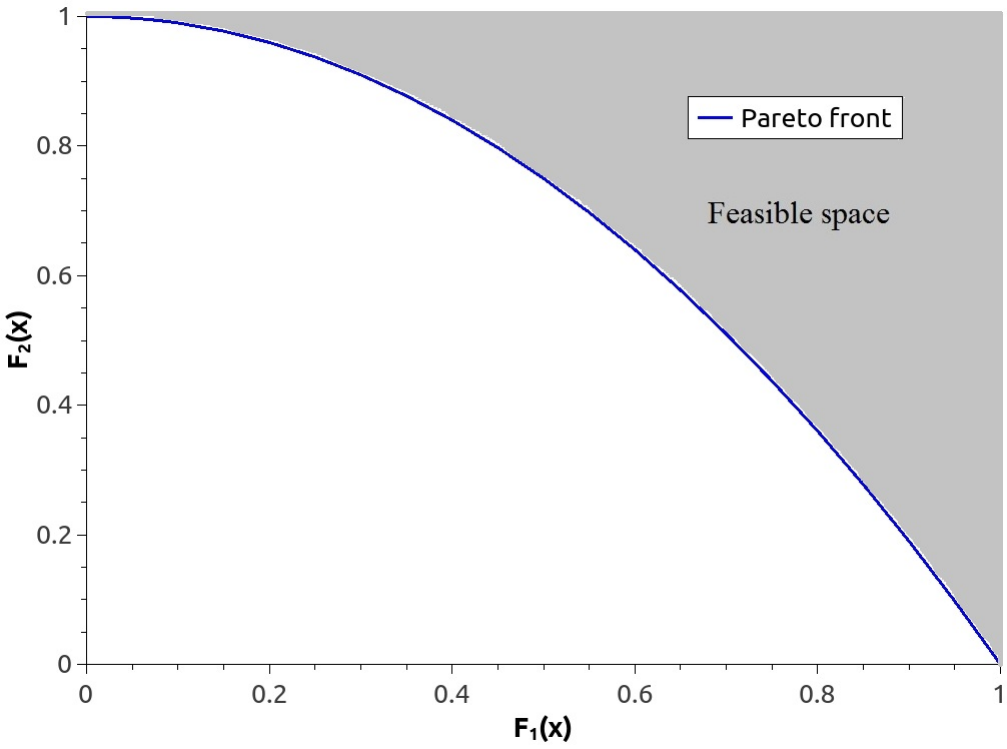

Figure 5.16 The Pareto front and feasible domain of the ZDT-2 problem

Similar the ZDT-1 problem, the expected number of solutions is 50. The settings for FSQP are: number of initial points (50), minimum weight (0.02), and weight increase (0.02). The result of the problem is shown in Figure 5-17, and the performance metrics are summarized in Table 5-3. In this case the problem gives an error of 0, this due to the fact that the solution points are exactly at the Pareto front. The piling factor infinity, since there are only two solution points in the solution set. This factor can only be calculated when there are more than three points in the solution set. The execution time is 4 sec.

Table 5-3: ZDT-2 problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0.286	0
PF	1.41	0.1	1.41
Time	4s	3.11s	2.39s
Number of solutions	2	50	3

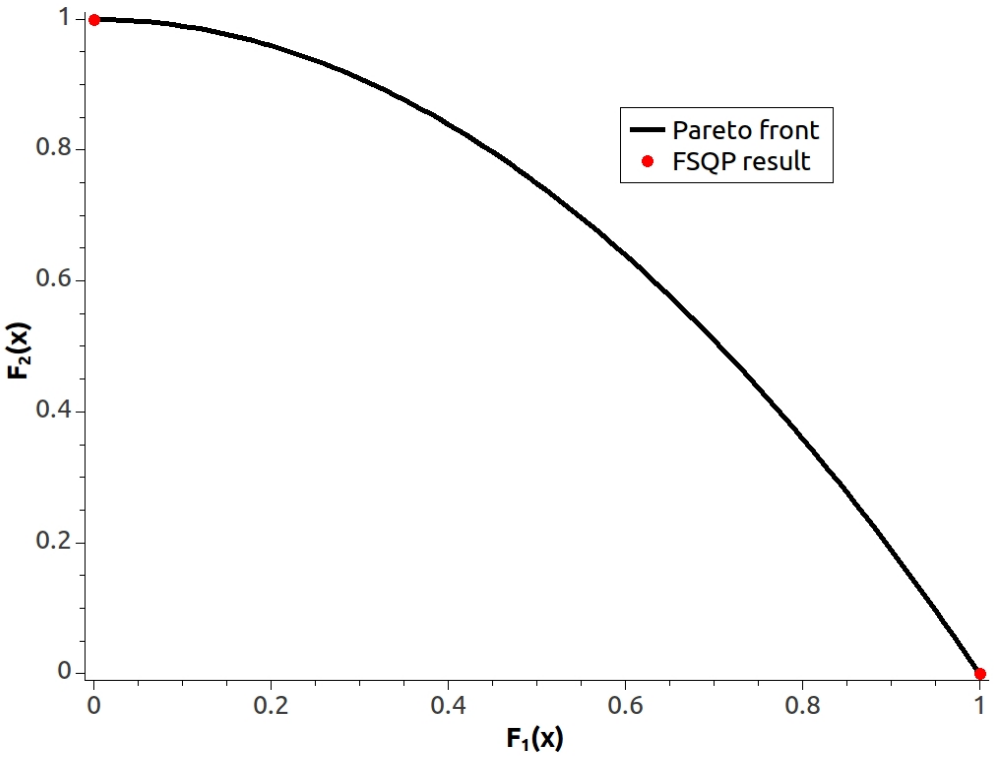

Figure 5.17: The ZDT-2 problem solved by FSQP

In this problem the settings for NSGA-II are (50) initial points, crossover rate (0.9), and mutation rate (0.01). In order to select the most adequate number of generations the problem was run with several numbers generations and shown in Figure 5-18.

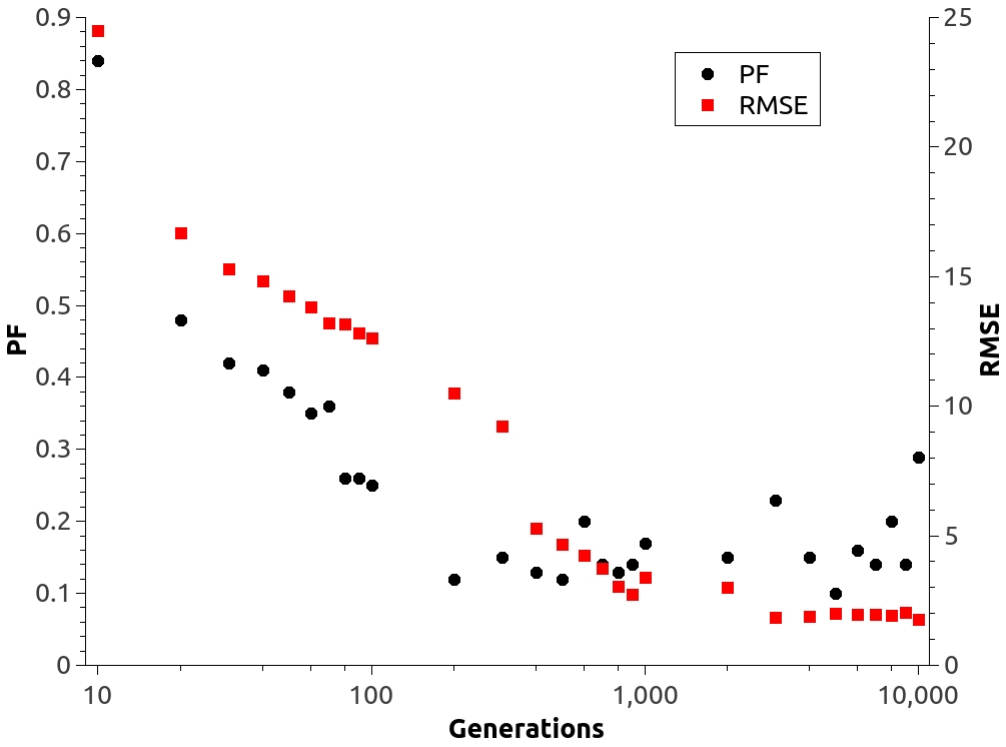

Figure 5.18 The PF and RMSE vs. number of generations

for ZDT-2 problem solved by NSGA-II

It can be seen from the above figure that 5000 generations yield the best result regarding both PF and RMSE. The results of NSGA-II are shown in Figure 5-19 and the performance metrics are given in Table 5-3.

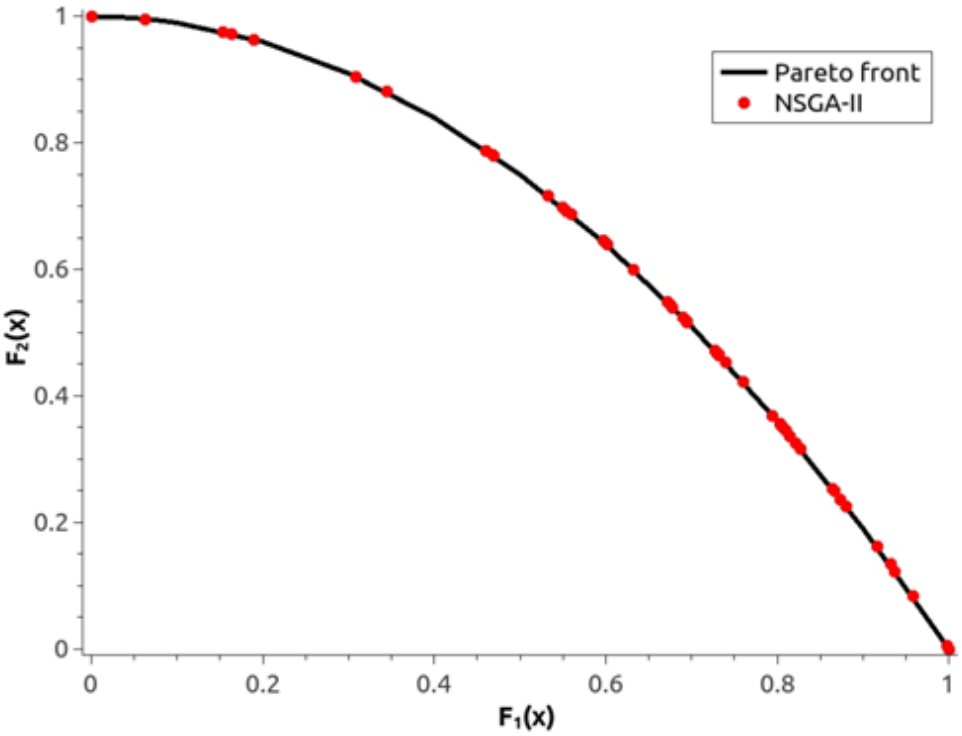

Figure 5.19: The ZDT-2 problem solved by NSGA-II

The parameters for this problem on the first hybrid algorithm are number of initial points (50) for both NSGA-II and FSQP, the maximum number of generations (5000) for the NSGA-II section. The results of the algorithm are shown in Figure 5-19, and the performance statistics are given in Table 5-3.

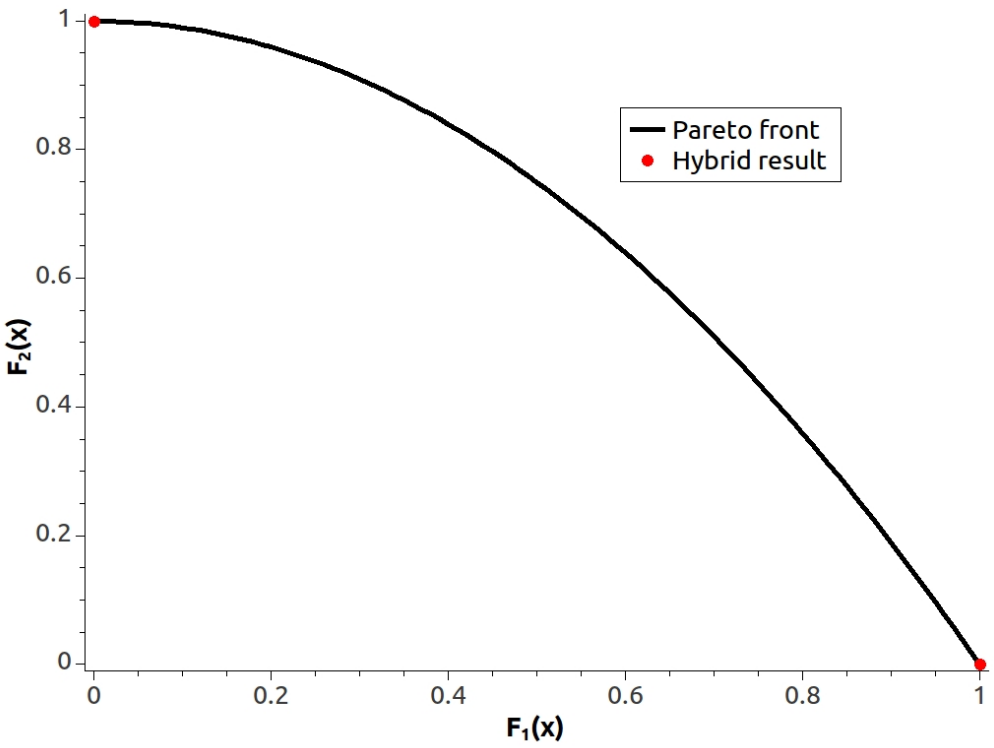

Figure 5.20: The ZDT-2 problem solved by the first hybrid algorithm

In this case, only the NSGA-II can generate an acceptable set of solutions; both the FSQP and the first hybrid algorithm are unable to properly solve this multi-objective optimization problem with a non-convex Pareto front. This is due to the WSF, which is adopted to obtain multiple solutions using a single-objective optimization method, has the inherent limitation of being suitable only to multi-objective optimization problems with convex Pareto fronts.

It can be seen from the three benchmark problems that the first hybrid algorithm is capable of solving a multi-objective optimization problem with a convex Pareto front. The accuracy and quality of solutions by the hybrid algorithm is better than both FSQP and NSGA-II when the optimization problem can be properly solved. An expected

drawback of the hybrid algorithm is that it is not capable of solving multi-objective optimization problems with non-convex Pareto fronts, as seen on the ZDT-2 problem. These results suggest that the hybrid algorithm provides a good initial approach to making a more efficient and effective hybrid algorithm.

5.3. The Second Hybrid Strategy

The second hybrid algorithm follows the same structure of the first one, except that, before running FSQP, a direction is calculated for each initial point towards the Pareto front. The basic idea is, for each initial point, a single solution is to be obtained using the single-objective optimization method without using the expensive WSF. With this strategy, the computational efficiency of the hybrid algorithm is expected to be significantly improved.

5.3.1 Algorithm of the Second Strategy

In the second hybrid algorithm, a new block called “direction calculation” is included in the flowchart shown in Figure 5-21. In this case there is no combination in the weighted sum, but a single set of weights per point are calculated and used.

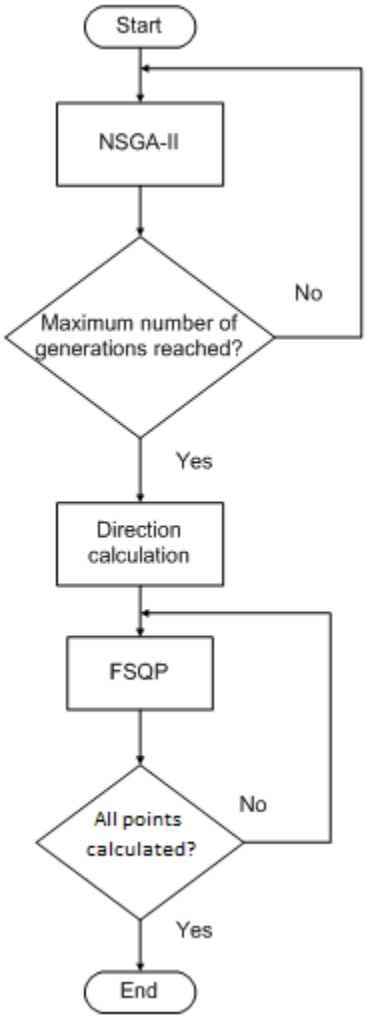

Figure 5.21: Flowchart of the second hybrid algorithm

The calculation of the weights of the individual points is based on the squared value of the direction cosines. In order to obtain evenly distributed solutions in the objective space, each of the objectives is normalized so as to make the maximum value of each direction equal to one and the minimum equal to zero. After normalizing the objectives, the square of the direction cosines are calculated as follows.

$$\cos^2(\alpha) = {x_i}/{\sum_{j=0}^{n} {x_j}^2} \qquad (5\text{-}14)$$

where i is the current objective and n is the number of objectives in the optimization problem.

Another property of the direction cosine is that the closest the point is to an axis the higher the value of its direction cosine is. If this value is used as the weight, which implies that the direction of the minimization would be more dominant in the direction of the objective of higher value. This would make the solutions of this problem cluster since all the solutions would be directed towards the middle of the Pareto front as shown in Figure 5.22.

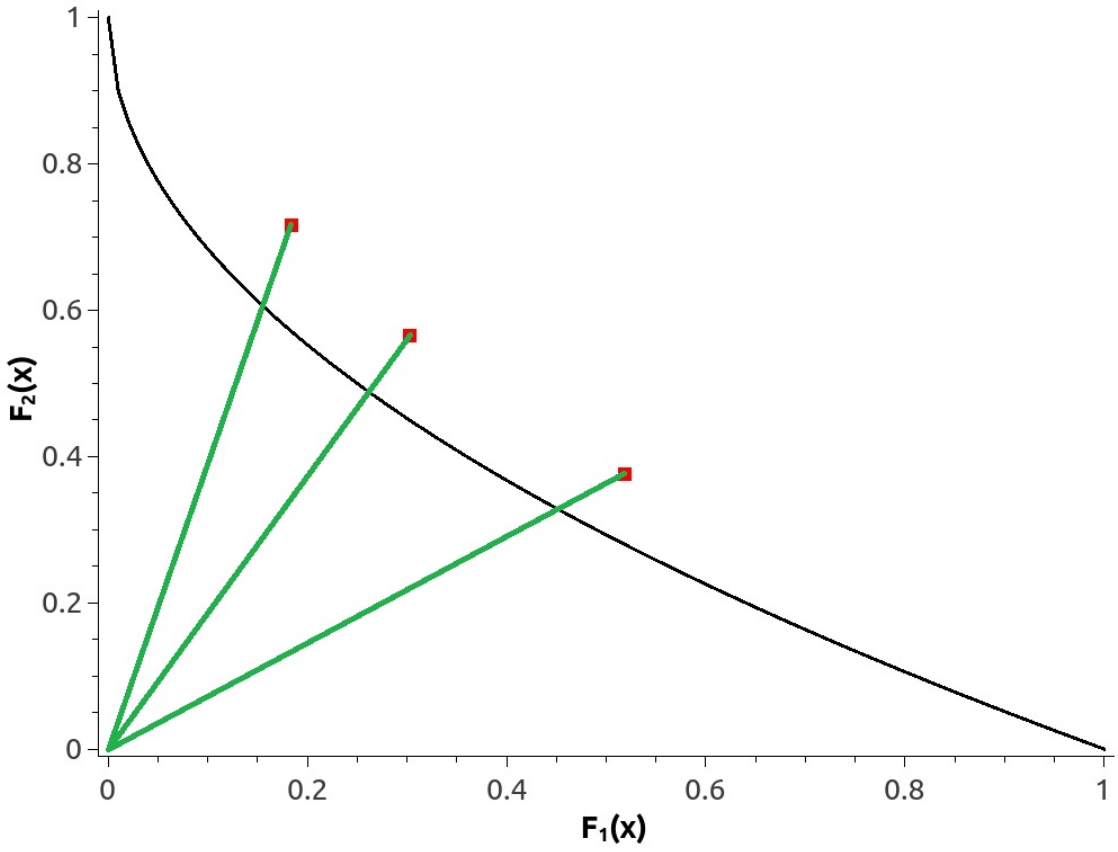

Figure 5.22: Direction calculation with direction cosines

This concern of solution clustering makes it necessary to swap the weights of solutions to spread out the solutions on the Pareto front. Specifically, the largest and the smallest weights are swapped; the second largest weight is swapped with the second smallest, and so on.

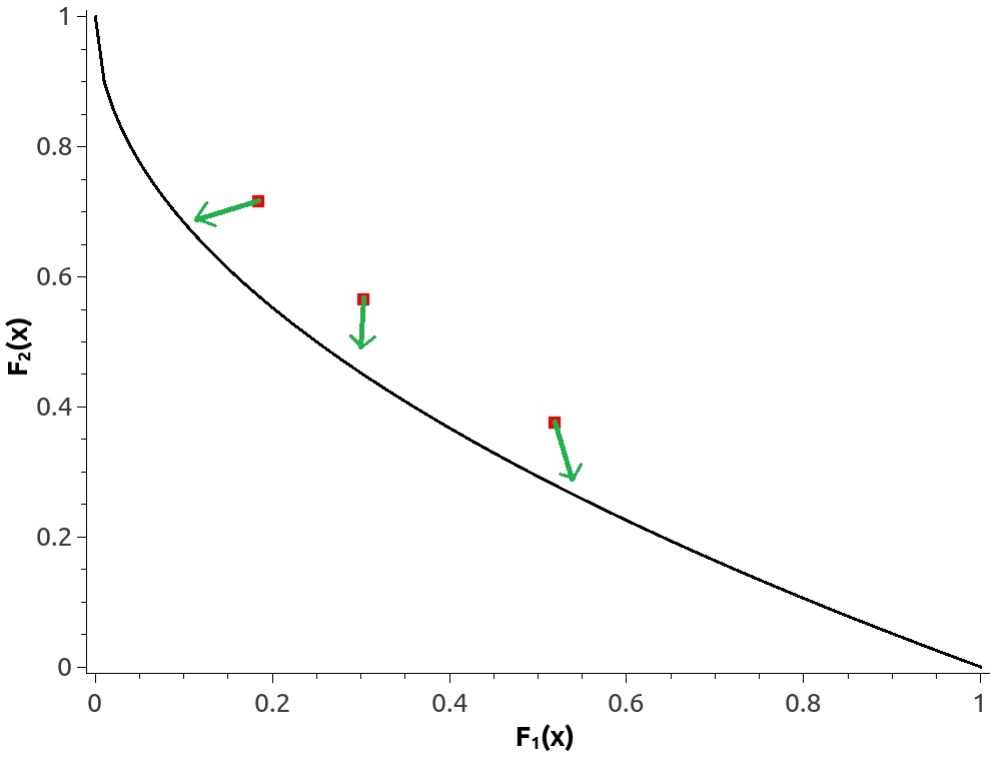

Figure 5.23: Solution direction with inverted direction cosines

Also, it is important to note that the direction cosines have the property that its sum of squares is equals to one, and the squares are non-negative. Therefore it coincides with the condition of the weighted sum combination that the sum of weights should be equal to one and they should be non-negative.

5.3.2 Results of the Second Approach

Same as the case of the first algorithm, the problems used to evaluate the performance of this algorithm are the Schaffer, ZDT-1 and ZDT-2 problems.

5.3.2.1. The Schaffer Problem

The input parameters for the second hybrid algorithm are: number of initial points for both NSGA-II and FSQP (50), and the maximum number of generations for NSGA-II (10). The performance metrics are given in Table 5-4 and the plot of the objective space is shown in Figure 5.24.

Table 5-4: Schaffer problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0	0
PF	0.15	0.69	1.25
Execution time	0s	0s	0s
Number of solutions	49	50	50

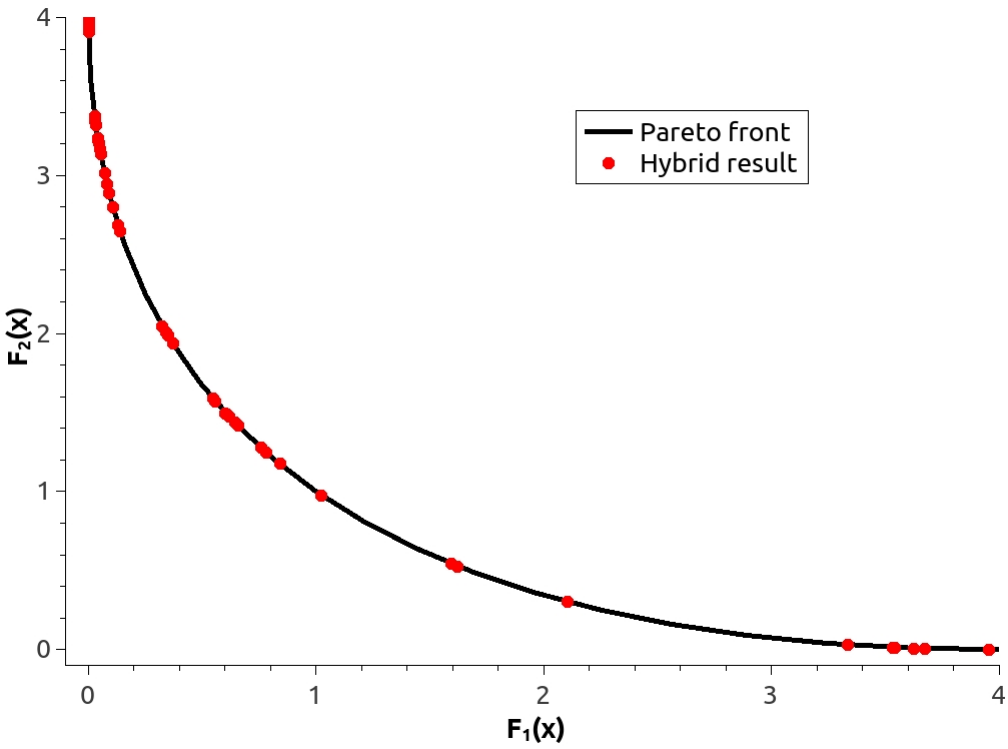

Figure 5.24: Schaffer solved by the second hybrid algorithm

In this case, the piling factor of the hybrid algorithm is slightly larger than those of both FSQP and NSGA-II, and the execution time is the same as those for FSQP and NSGA-II. This indicates that the second hybrid strategy is capable of obtaining a nondominated Pareto front with improved computational efficiency compared to the first hybrid algorithm.

5.3.2.2. The ZDT-1 Problem

For the ZDT-1 problem, the parameters for the second hybrid algorithm are the number of initial points (50) for NSGA-II and FSQP, and the maximum number of generations (10) for NSGA-II. The results of the algorithm are shown in Figure 5-25, and the performance metrics are given in Table 5-5.

Table 5-5: ZDT-1 problem results

	FSQP	NSGA-II	Hybrid
RMSE	1.93E-06	8.43E-04	2.75E-06
PF	2.95	0.16	0.33
Execution time	16s	1.25s	0.29
Number of solutions	31	50	21

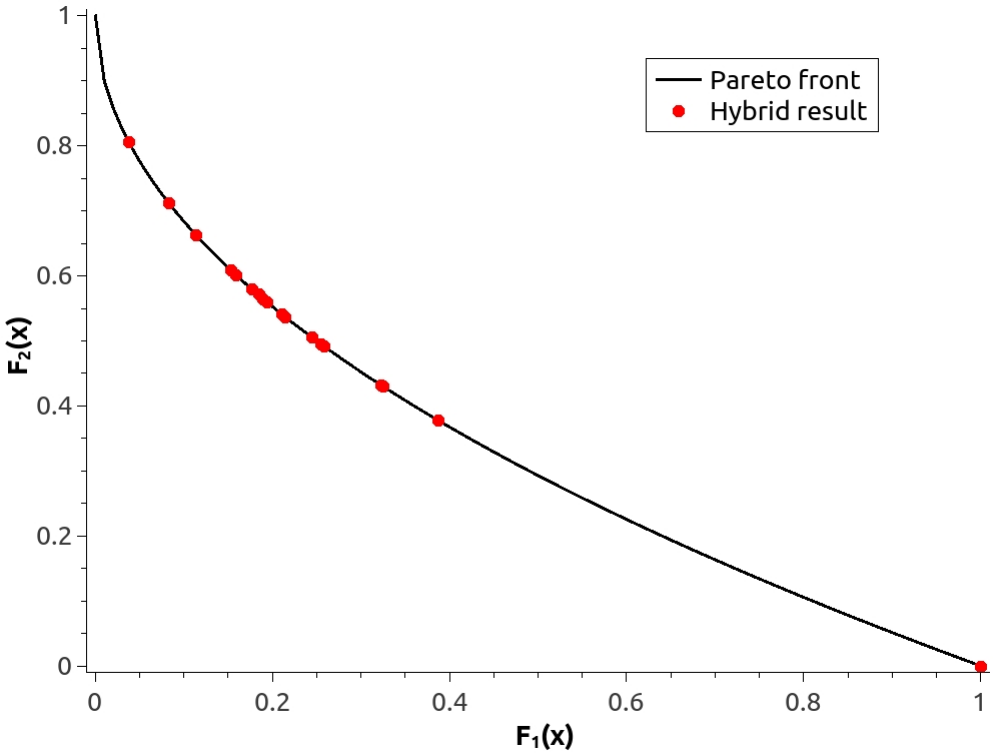

Figure 5.25: The ZDT-1 problem solved by the second hybrid algorithm

In this case, the RMSE by the hybrid algorithm is about fifty percent larger than FSQP yet the RSME of NSGA-II is 10^2 times larger than the hybrid algorithm. The PF of

the hybrid algorithm is twice as large as that of NSGA-II, which is the smallest among the three algorithms. The PF of the hybrid algorithm is more than ten times smaller than that of FSQP. Regarding the execution time the hybrid algorithm is the fastest followed by the NSGA-II and FSQP that has the highest value, which is significantly larger than both the hybrid algorithm and NSGA-II. Again, the number of solutions of FSQP and the hybrid algorithm are reduced due to nondominance check performed by the WSF.

5.3.2.3. The ZDT-2 Problem

As in the case of the ZDT-1 problem, the parameters for the ZDT-2 problem are the number of initial points for NSGA-II and FSQP (50) and the maximum number of generations (10) for NSGA-II only. The results of the hybrid algorithm are shown in Figure 5-26, and the performance metrics are given in Table 5-6.

Table 5-6: ZDT-2 problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0.286	0
PF	1.41	0.1	1.41
Time	4s	3.11s	0.7s
Number of solutions	2	50	2

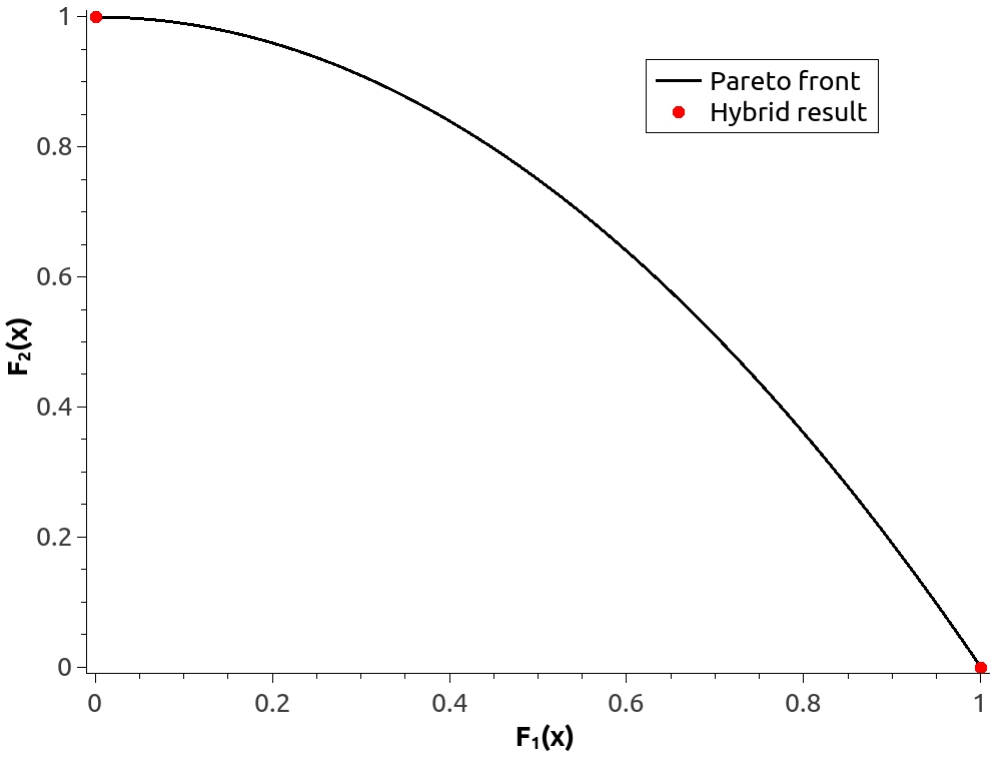

Figure 5.26: ZDT-2 problem with second hybrid algorithm

In this case, the only algorithm that is capable of obtaining an acceptable set of solutions is NSGA-II. Both the second hybrid algorithm and FSQP are shown to be unable to properly solve a multi-objective optimization problem with a non-convex Pareto front.

As seen from the results, there is a significant improvement in the execution time to obtain similar results with respect to the RMSE and PF. Despite its improvement on computational efficiency over the first hybrid algorithm, the second hybrid algorithm is still not capable of giving a satisfactory solution set for optimization problems with non-convex Pareto fronts.

5.4. The Third Hybrid Strategy

To overcome the issue of the first two hybrid algorithms in solving optimization problems with non-convex Pareto fronts, the third hybrid algorithm is proposed. In this case, a calculation is introduced in order to change the multi-objective problem into the minimization of one objective subjected to the original constraints and additional equality constraints converted from the rest of objectives in the original optimization problem. The expectation of this approach is to keep the execution time lower than or at least the same as the second hybrid algorithm, with the added capability of solving optimization problems with non-convex Pareto front.

5.4.1 Algorithm of the Third Strategy

The third hybrid algorithm includes a calculation that transforms the MOOP into several single objective optimization problems with as many equality constraints as extra objectives the original problem has; i.e. if the original problem has four objectives and no constraints, then the modified problem would have one objective and three equality constraints. The flowchart of the algorithm is shown in Figure 5-27.

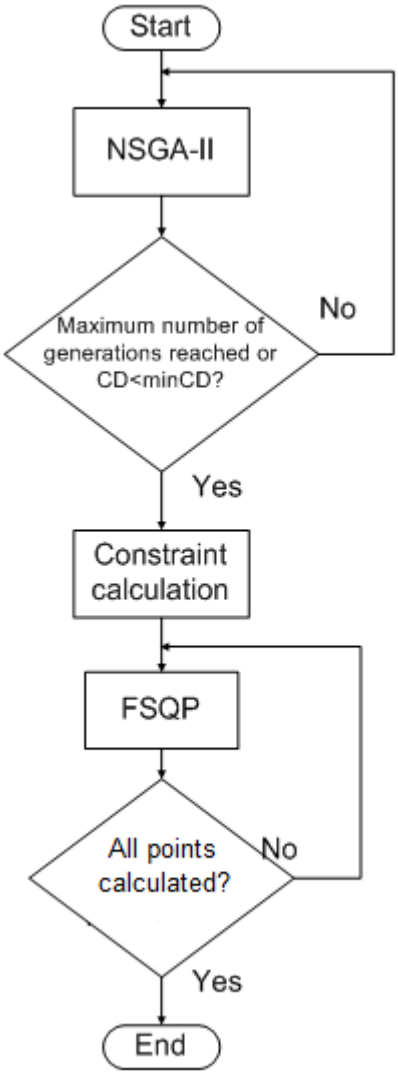

Figure 5.27: Third hybrid algorithm

In the third hybrid algorithm, the equality constraints are set based on a set of virtual points that are built by normalizing the objectives obtained from the results of the NSGA-II stage and tracing a line between the original point and its normalized image, as shown in Figure 5-28.

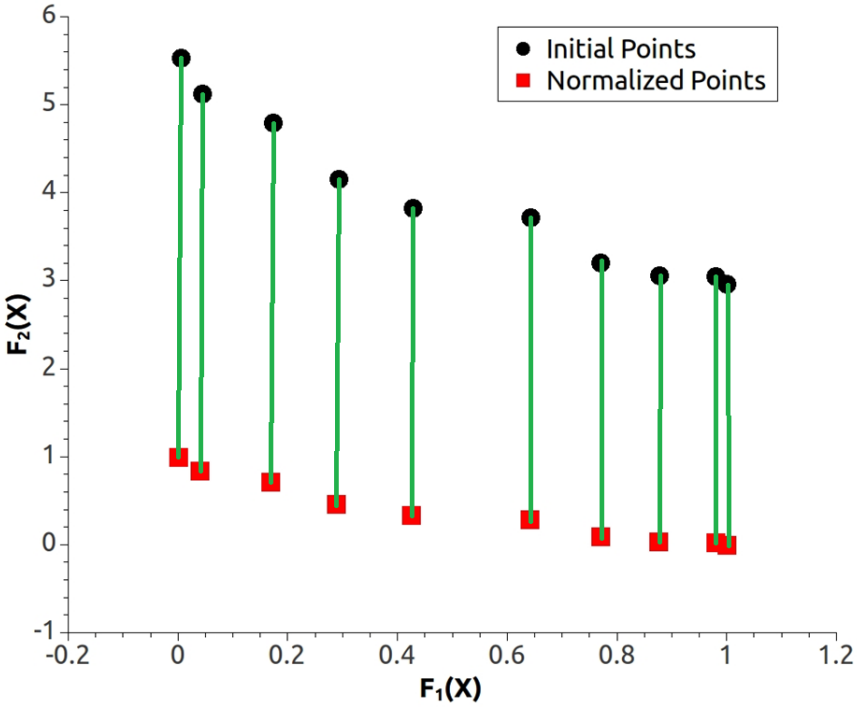

Figure 5.28: Constraint calculation in the third hybrid algorithm

The following equations give the general equation of a line in n dimensions. Equation (5-16) shows how an equation can be obtained for any two dimensions.

$$\begin{pmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_n(\vec{x}) \end{pmatrix} = \begin{pmatrix} f_1(\vec{x}_1^{\,*}) \\ f_2(\vec{x}_1^{\,*}) \\ \vdots \\ f_n(\vec{x}_1^{\,*}) \end{pmatrix} + m \begin{pmatrix} f_1(\vec{x}_2^{\,*}) \\ f_2(\vec{x}_2^{\,*}) \\ \vdots \\ f_n(\vec{x}_2^{\,*}) \end{pmatrix} \quad (5\text{-}15)$$

$$\frac{f_1(\vec{x}) - f_1(\vec{x}_1^{\,*})}{f_1(\vec{x}_2^{\,*})} = \frac{f_2(\vec{x}) - f_2(\vec{x}_1^{\,*})}{f_2(\vec{x}_2^{\,*})} \quad (5\text{-}16)$$

This algorithm also includes an additional termination condition for the NSGA-II segment of the hybrid algorithm. This termination condition is based on the crowding distance that is calculated as part of the NSGA-II algorithm, as discussed in Section 4.2.3.

This termination condition is included to improve the quality and/or efficiency of the first stage whose results will be used in the second stage to obtain more accurate solutions.

In order for the hybrid strategy to be effective, the objective functions are normalized to a range from 0 to 1 to avoid clustering. Also, it is important to note that there is a limitation regarding the quantity of equality constraints of the transformed problem. Therefore, this algorithm is suitable only for problems that, after being transformed, have less number of total equality constraints than the number of design variables. If the number of equality constraints is equal to or greater than the number of variables after the problem is transformed, the problem cannot be solved.

5.4.2 Results of the Third Approach

Same as the two previous hybrid approaches, the third hybrid algorithm is tested using the Schaffer, ZDT-1 and ZDT-2 problems. In this case, the hybrid algorithm is expected to be capable of solving optimization problems with non-convex Pareto fronts.

5.4.2.1. The Schaffer Problem

The input parameters for the third hybrid algorithm are: number of initial points for both NSGA-II and FSQP (50), the minimum crowding distance for NSGA-II (0.01), and the maximum number of generations for NSGA-II (10). The results are shown in Figure 5-29 and the performance metrics are given in Table 5-7.

Table 5-7: Schaffer problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0	0
PF	0.15	0.69	1.13
Execution time	0s	0s	0.01s
Number of solutions	49	50	50

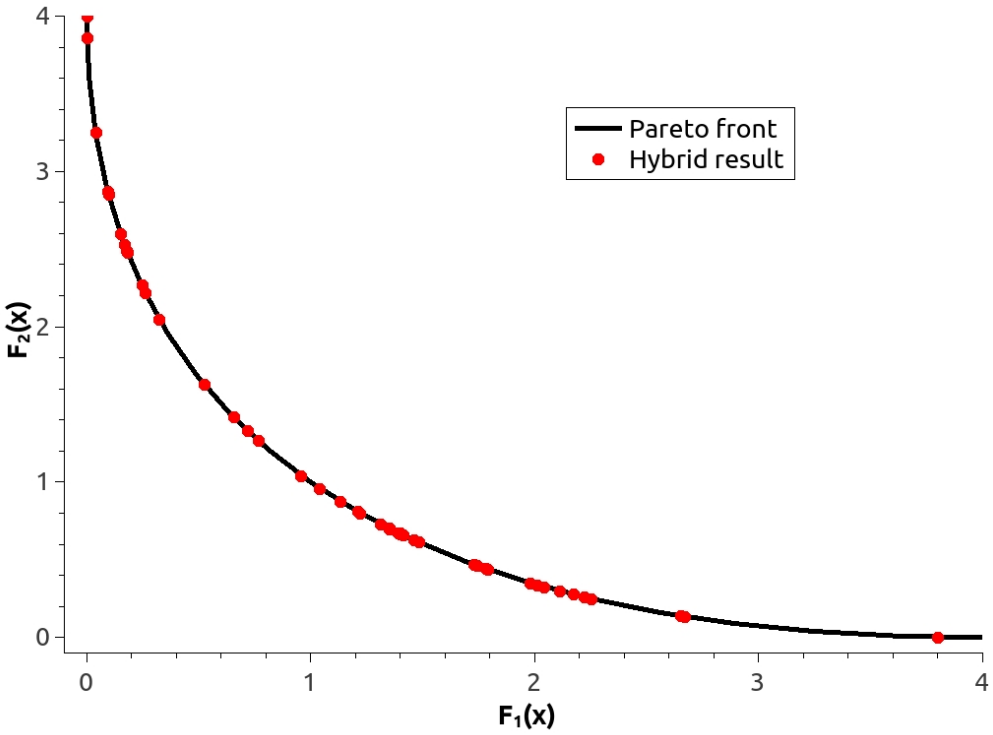

Figure 5.29: The Schaffer problem solved by the third hybrid algorithm

Similar to the previous algorithms, the third hybrid algorithm is capable of solving this problem, with the PF higher than the other algorithms and the execution time of 0.01s. In addition, the third hybrid algorithm is capable of obtaining solution points on the Pareto front.

5.4.2.2. The ZDT-1 Problem

The parameters for this problem using the third hybrid algorithm are initial points (50) for both NSGA-II and FSQP, the maximum number of generations (10) for the NSGA-II section, and the minimum crowding distance (0.01) for NSGA-II. The results of the third hybrid algorithm are shown in Figure 5-30 and the performance metrics are given in Table 5-8.

Table 5-8: ZDT-1 problem results

	FSQP	NSGA-II	Hybrid
RMSE	1.93E-06	8.43E-04	8.1E-07
PF	2.95	0.16	0.12
Time	16s	1.25s	2.02s
Number of solutions	31	50	50

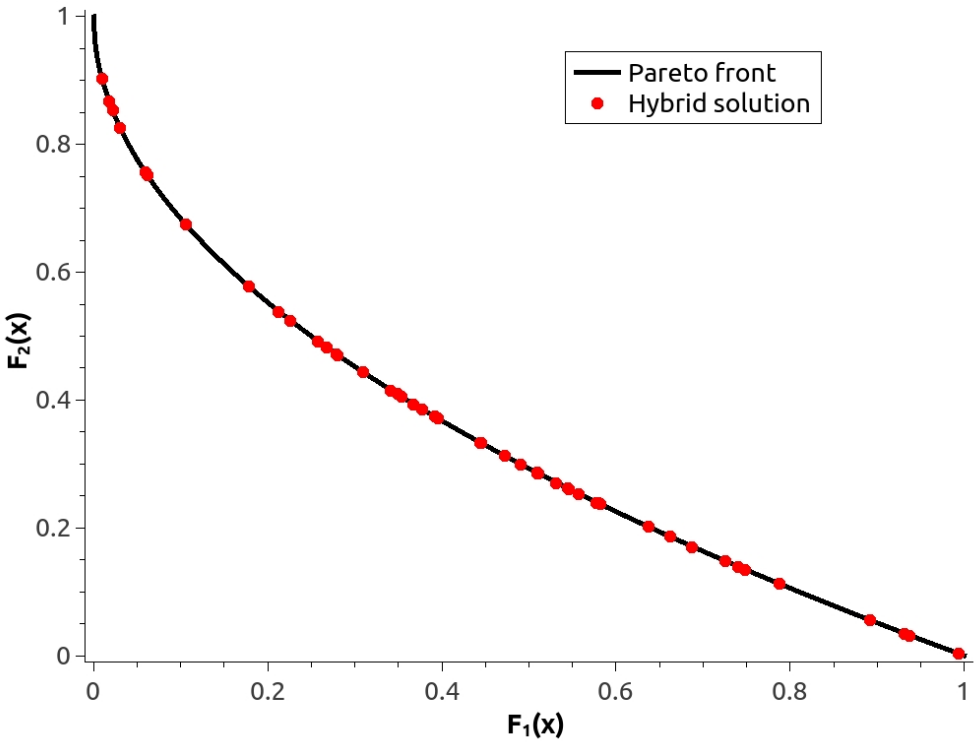

Figure 5.30: The ZDT-1 problem solved by the third hybrid algorithm

In this case, the RMSE of the hybrid algorithm is about 2.5 times smaller than that of FSQP, which in turn is 10^2 times smaller than that of NSGA-II. The PF of the hybrid algorithm is 33% larger than that of NSGA-II, which is approximately twenty times smaller than that of FSQP. Regarding the execution time, the NSGA-II is the fastest followed by the hybrid algorithm; FSQP takes significantly more time than both of them. This results show that the third hybrid strategy is capable of accurately solving multi-objective optimization problems with convex Pareto fronts.

5.4.2.3. The ZDT-2 Results

Same as the previous problems, the inputs for the ZDT-2 problem are number of initial points for NSGA-II and FSQP (50) generations in the NSGA-II stage (10) and the

minimum crowding distance (0.01) for NSGA-II. The results of the algorithm are shown in Figure 5-31 and the performance metrics are given in Table 5-9.

Table 5-9: ZDT-2 problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	2.86E-01	8.3E-07
PF	1.41	0.1	0.2
Execution time	4s	3.11s	1.6s
Number of solutions	2	50	50

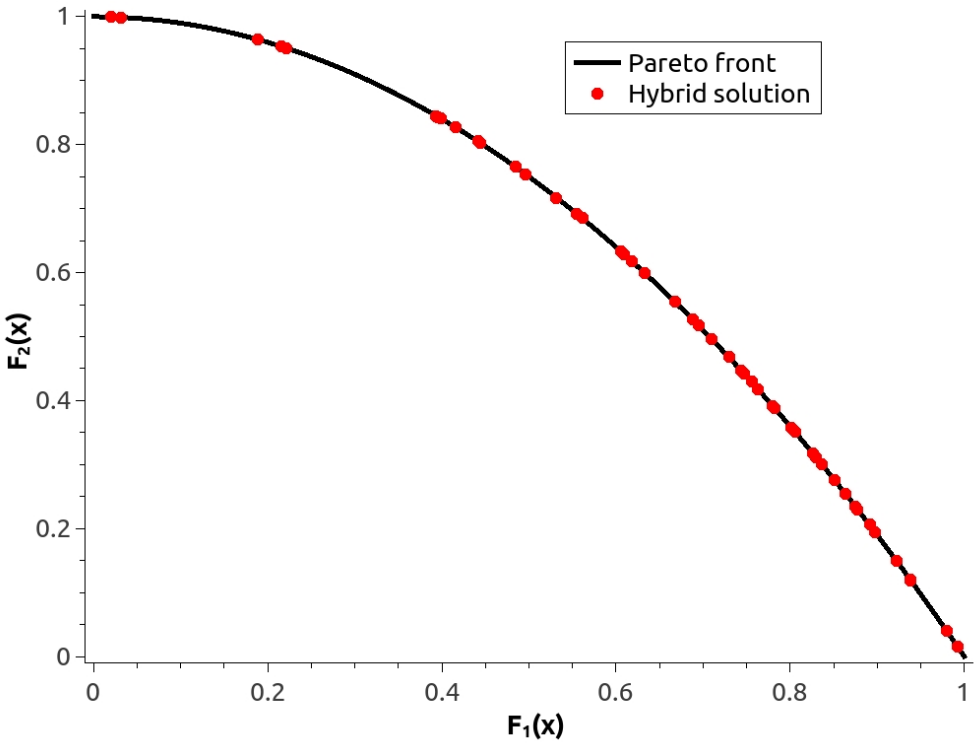

Figure 5.31: The ZDT-2 problem solved by the third hybrid algorithm

This problem can be solved properly only by this hybrid approach and NSGA-II because of limitations of the other algorithms in solving optimization problems with non-convex Pareto fronts. In this case, the RMSE of the hybrid algorithm is in the order of 10^6 times smaller than that of NSGA-II. The PF of the hybrid algorithm doubles that of the NSGA-II. The execution time of the hybrid algorithm is approximately half of that of NSGA-II. Finally, the number of solutions of each problem remained the same at 50.

The results of the third hybrid algorithm show that it is capable of solving both MOOPs with convex and non-convex Pareto fronts. Its performance is better than NSGA-II and FSQP with regards to the RMSE values, particularly in the cases of ZDT-1 and ZDT-2 problems. The hybrid algorithm also has comparable results regarding the PF; this means that the distribution of solution points on the Pareto front is as good as the other algorithms. The execution time and number of solutions of the hybrid algorithm are at the same levels as those of NSGA-II, except for the execution time on solving the ZDT-2 problem in which case the hybrid algorithm is faster than NSGA-II.

From the results shown in this chapter it can be concluded that the third hybrid algorithm gives the best approach in terms of not only keeping a good distribution of the solutions on the Pareto front, but also being capable of solving MOOPs with non-convex Pareto fronts. The third hybrid algorithm will be the only hybrid algorithm that will be tested and compared to other algorithms in the following chapters.

CHAPTER 6: BENCHMARK OPTIMIZATION PROBLEMS

In this chapter, eight benchmark optimization problems are used to test the hybrid algorithm and compare its solutions and performance with those of the NSGA-II and the FSQP.

The benchmark problems are used for the fact that their Pareto fronts are available in analytical forms. This makes it possible to calculate errors of the solutions by the hybrid algorithm. Without a known Pareto front, it is not possible to calculate the error between a solution point and the true Pareto optimum in the objective space. In addition, the convexities and ranges of the benchmark problems are known as well.

In order to solve problems in which the Pareto front's range is not between zero (0) and one (1), these problems need to be rescaled in order to be properly solved.

6.1. The Bihn Problem

Introduced by Bihn (1996), this problem has two objective functions and two design variables. The problem is stated as follows.

$$\text{Min} \quad \big(f_1(x), f_2(x)\big) \tag{6-1}$$

$$f_1(x) = x_1^2 + x_2^2 \tag{6-2}$$

$$f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \tag{6-3}$$

The Pareto front is given by:

$$f_2(x) = 2\big(\sqrt{f_1(x)/2} - 5\big)^2 \tag{6-4}$$

$$0 \le f_1(x) \le 50 \tag{6-5}$$

This problem is an unconstrained convex minimization problem as shown in Figure 6-1.

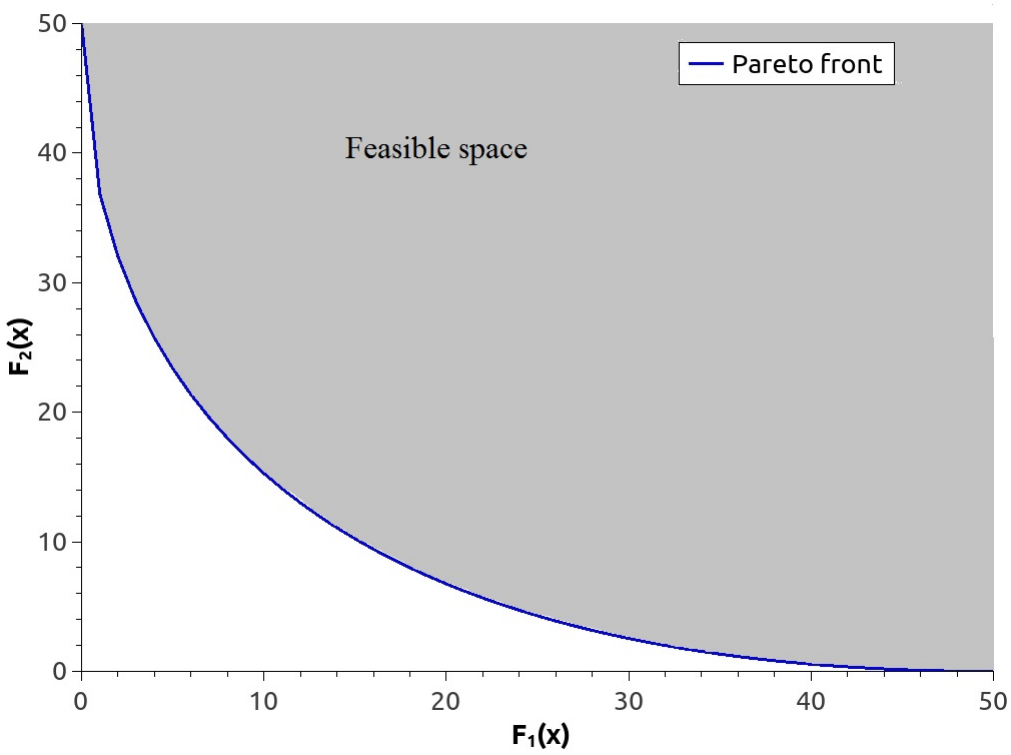

Figure 6.1: The Pareto front and feasible domain of the Bihn problem

In order to set the range of both objectives between zero and one, the original objective functions are divided by fifty. The settings to solve this problem for FSQP are the number of initial points (50), minimum weight for WSF (0.02), and weight increment (0.02). The solution is shown in Figure 6-2 and the performance metrics are given in Table 6-1.

Table 6-1: Bihn problem results

	FSQP	NSGA-II	Hybrid
RMSE	8.75E-07	7.27E-01	1.8E-05
PF	1.94	6.19	11.5
Execution time	0.03s	0.35s	0.04s
Number of solutions	49	50	50

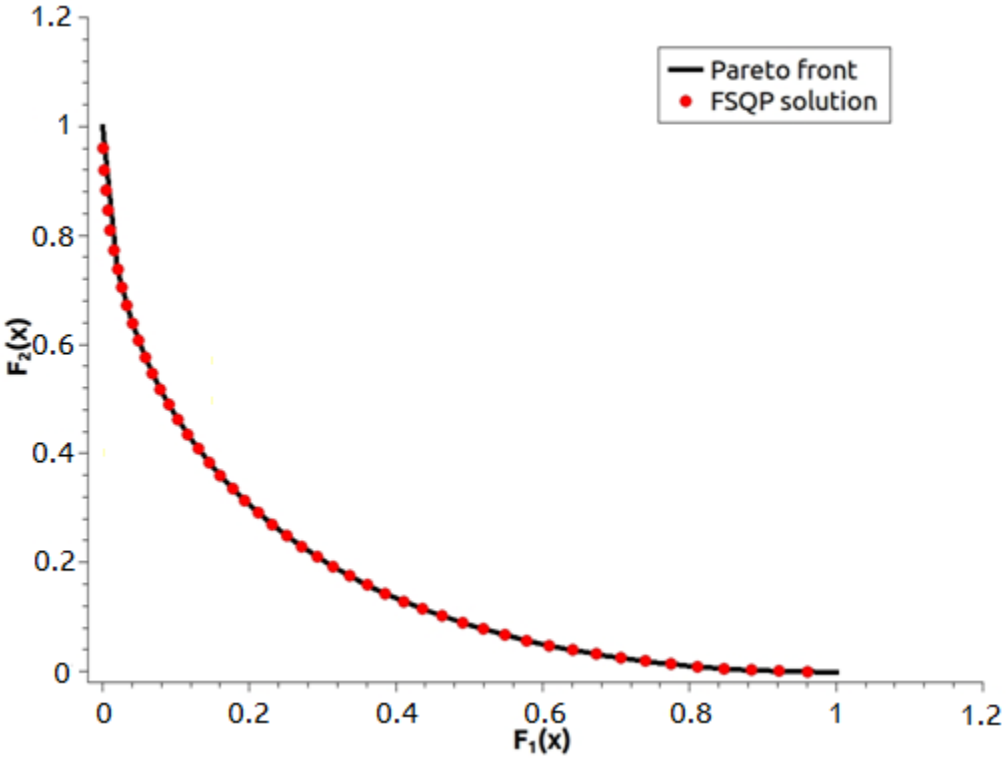

Figure 6.2: The Bihn problem solved by FSQP

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the maximum

number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-3.

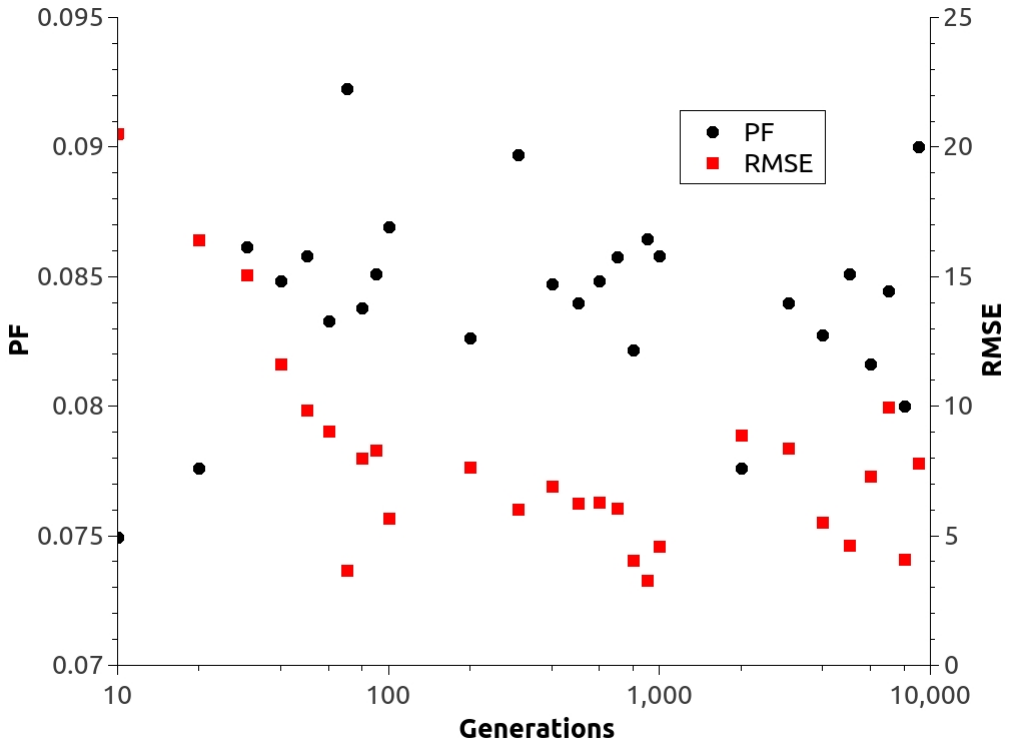

Figure 6.3: The PF and RMSE vs. number of generations for the Bihn problem solved by NSGA-II

From Figure 6.3, it is determined that 600 generations give the best results and thus are selected. The results of NSGA-II are shown in Figure 6-4 and Table 6-1 gives the performance metrics.

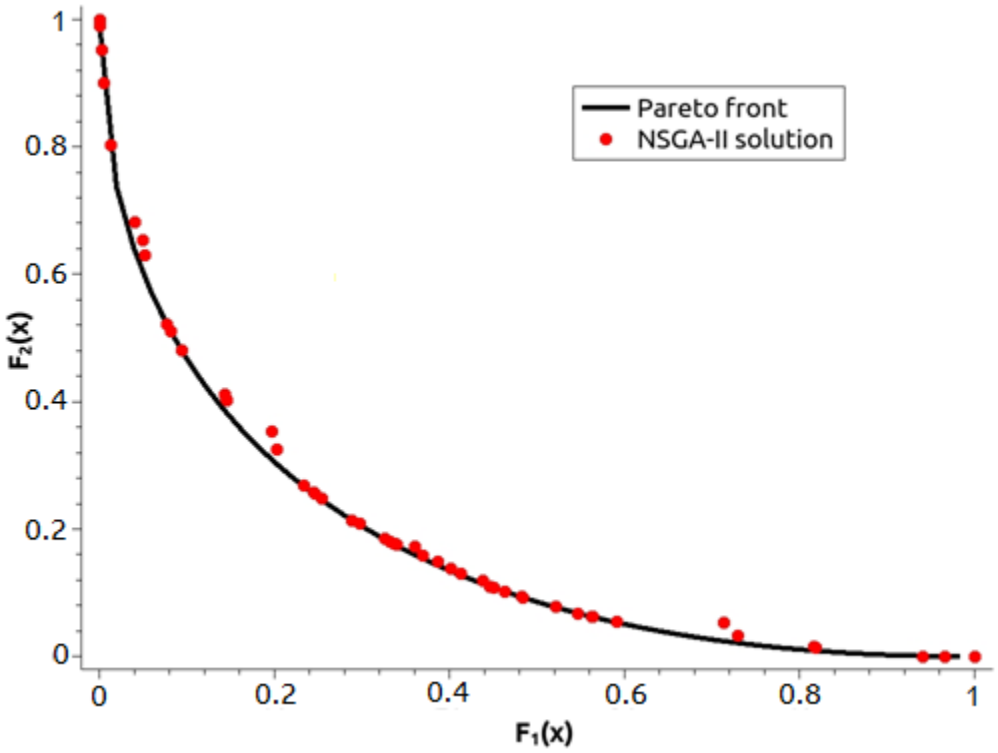

Figure 6.4: The Bihn problem solved by NSGA-II

The settings for the hybrid algorithm are the number of initial points for both NSGA-II and FSQP (50), maximum number of generations (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The results of the hybrid algorithm are shown in Figure 6-5, and the performance metrics are given in Table 6-1.

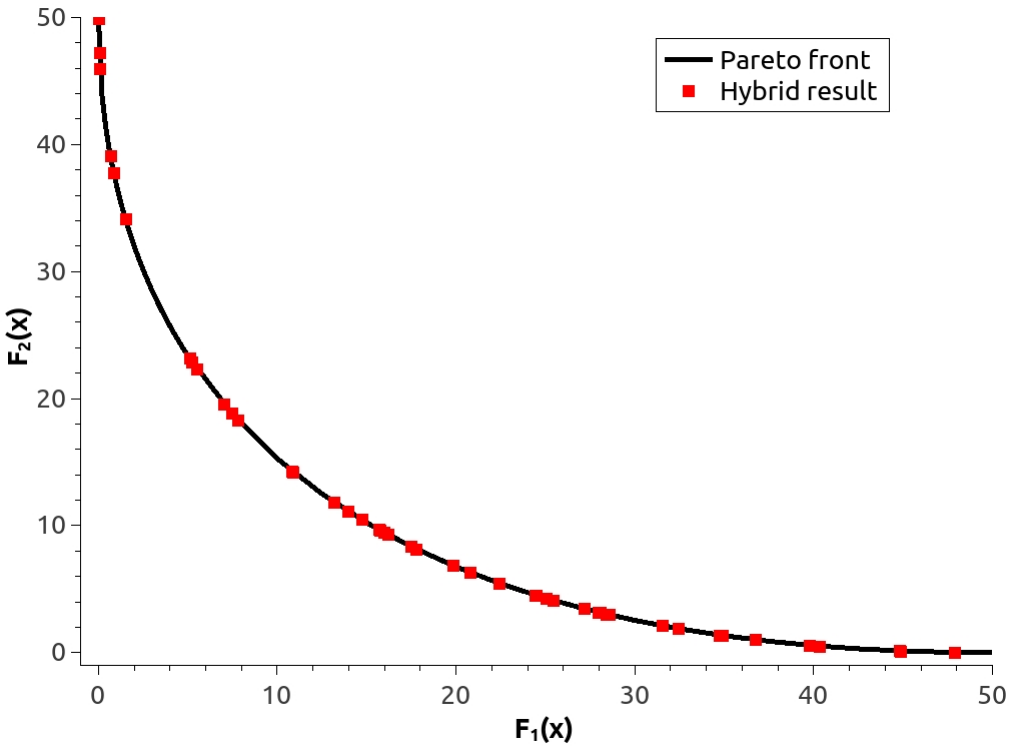

Figure 6.5: The Bihn problem solved by the hybrid algorithm

By solving the Bihn problem with the three algorithms, it can be concluded that the fastest algorithm is FSQP, followed closely by both NSGA-II and the hybrid algorithm. Regarding the piling factor, FSQP has the best distribution from the three algorithms followed by NSGA-II and the hybrid has highest value. Finally, regarding the RMSE, FSQP has the lowest value followed by the hybrid algorithm which is about 100 times larger than the hybrid algorithm, and NSGA-II has the highest value that is 10^4 times larger than the hybrid algorithm.

6.2. The Messac Problem

Introduced by Messac (2000), this problem contains a convex as well as a non-convex region on the Pareto front as seen in Figure 6-6. Also, all of the solution points

are part of the Pareto front making the RMSE zero for any of the possible solution sets. This problem is used to identify if the algorithms are capable of solving problems that have a partially concave Pareto front.

$$\text{Min} \quad \left(f_1(x), f_2(x)\right) \tag{6-6}$$

$$f_1(x) = e^{-x} + 1.4e^{-x^2} \tag{6-7}$$

$$f_2(x) = e^{x} + 1.4e^{-x^2} \tag{6-8}$$

$$\text{s.t} \quad -2 \leq x \leq 2 \tag{6-9}$$

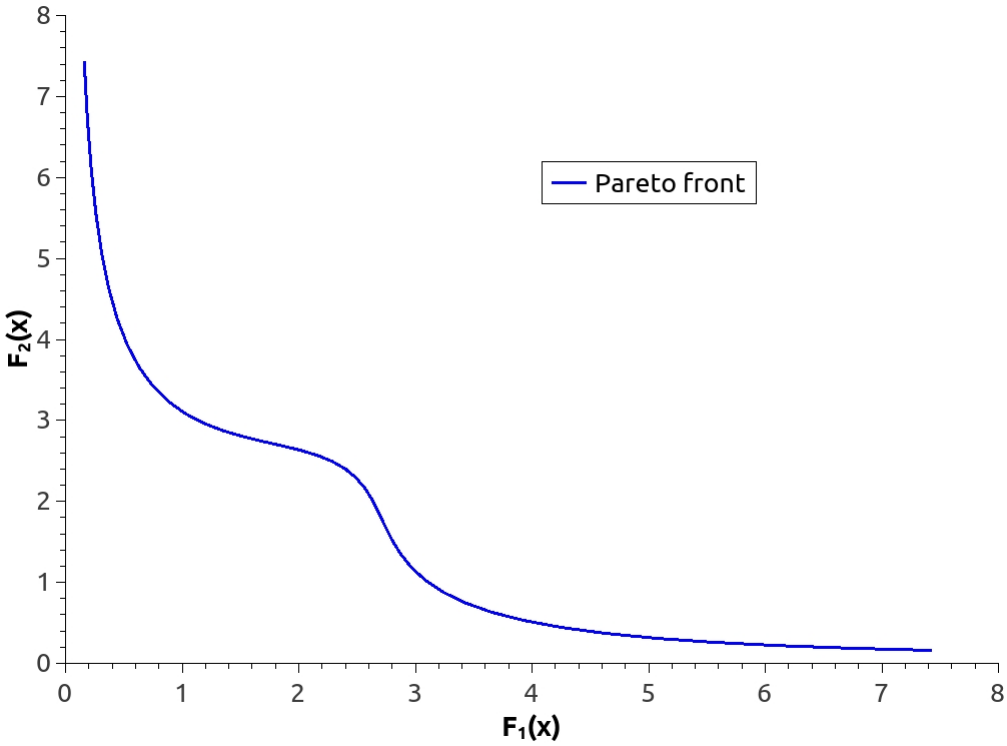

Figure 6.6: The Pareto front of the Messac problem

In order to set the range of the objectives between zero and one, the original objective functions are divided by 7.5. The inputs to solve this problem with FSQP are the number of initial points (50), the minimum weight for WSF (0.02) and the weight

increment (0.02). The results of the algorithm are shown in Figure 6-7, and the performance statistics are given in Table 6-2.

Table 6-2: Messac problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0	0
PF	3.12	2.29	1.53
Execution time	0.05s	0.01s	0.02s
Number of solutions	49	50	50

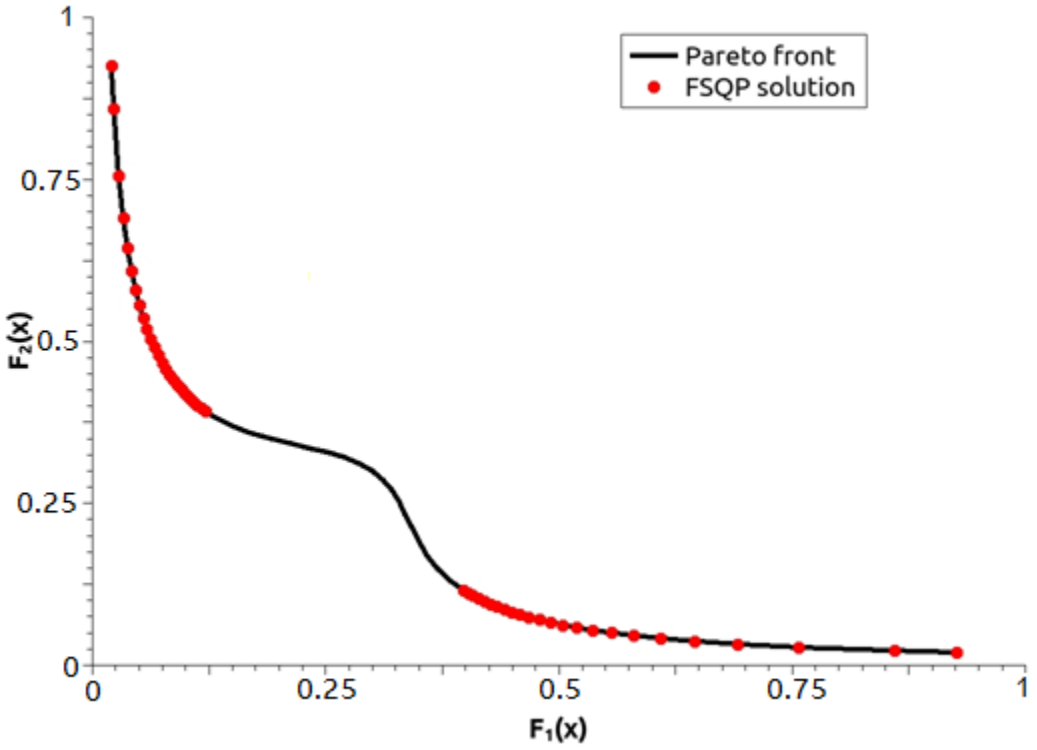

Figure 6.7: The Messac problem solved by FSQP

As seen on Figure 6-7, FSQP is unable to provide solution points where the Pareto front is non-convex and thus leaves this portion blank on the Pareto front.

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to set the maximum number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-8.

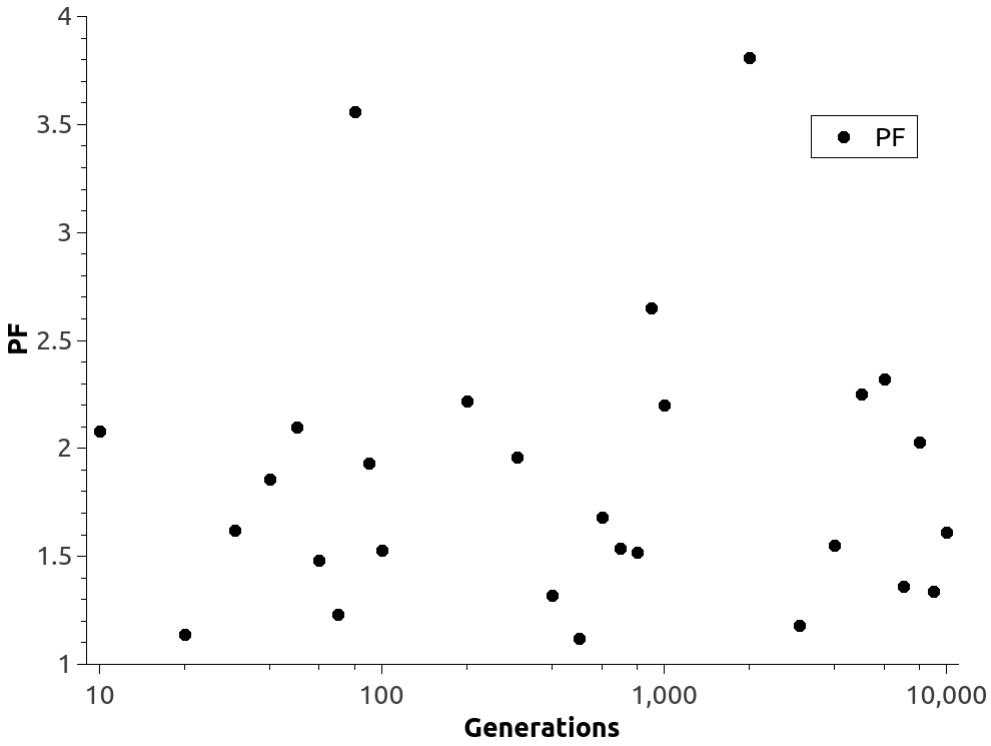

Figure 6.8: The PF vs. number of generations
for the Messac problem solved by NSGA-II

As shown in Figure 6.8, the best solutions are obtained at 20 generations. The results of the algorithm are shown in Figure 6-9 and the performance statistics are given in Table 6-2.

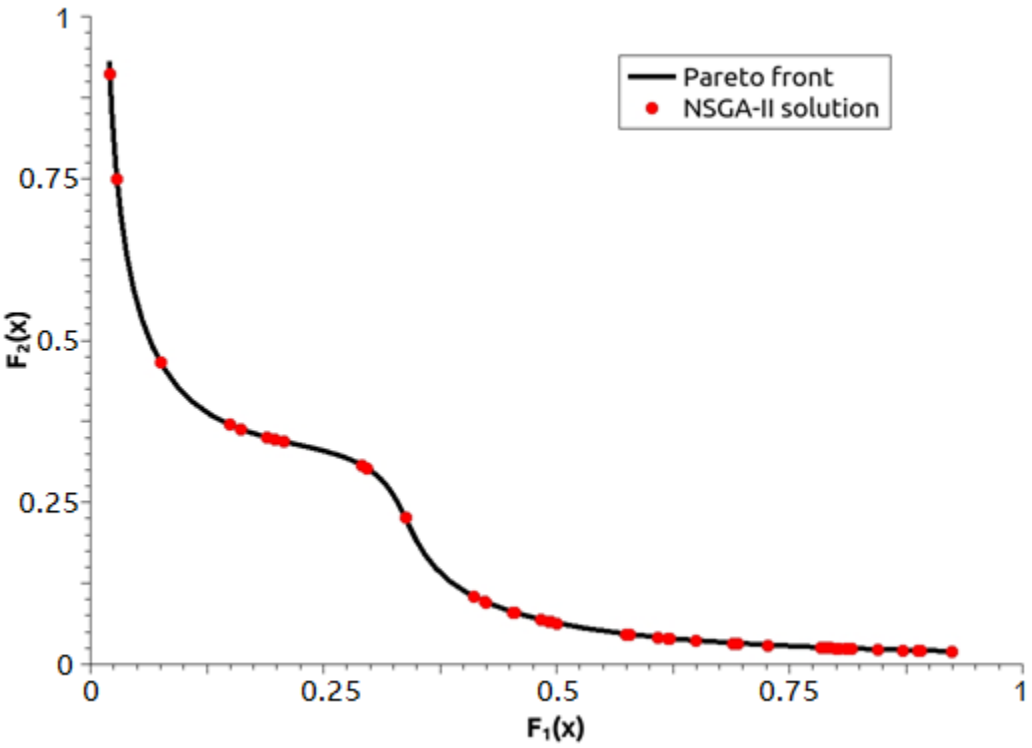

Figure 6.9: The Messac problem solved by NSGA-II

The inputs for the hybrid algorithm are the number of initial points (50), maximum number of generations (100), and minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The results of the algorithm are shown in Figure 6-10 and the performance statistics are given in Table 6-2.

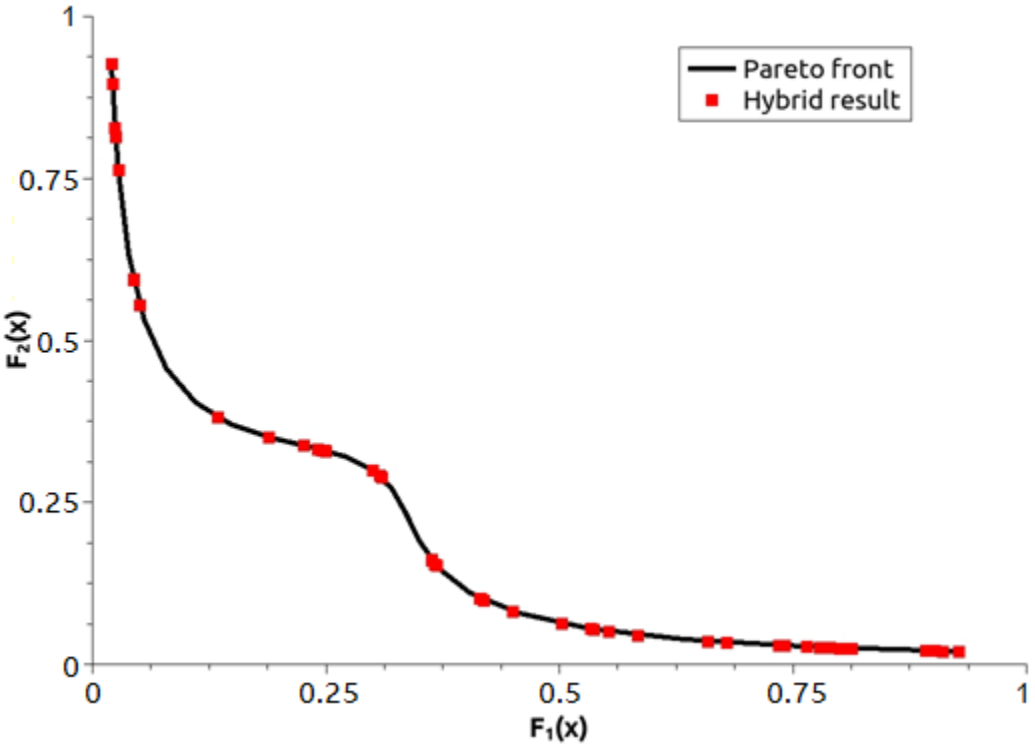

Figure 6.10: The Messac problem solved by the hybrid algorithm

The solutions to the Messac problem show that FSQP is unable to provide solutions on the non-convex region of the Pareto front. This is also demonstrated by the largest PF by FSQP. The hybrid algorithm has the lowest PF among the three. The fastest algorithm is NSGA-II followed closely by the hybrid algorithm and the slowest algorithm is FSQP. This problem shows that the hybrid algorithm is capable of properly solving MOOPs that have both convex and non-convex regions on the Pareto fronts.

6.3. The Murata Problem

Introduced by Murata (1995), this problem is an unconstrained minimization problem with two objectives and its formulation is given by:

$$\text{Min} \qquad \big(f_1(x), f_2(x)\big) \qquad (6\text{-}10)$$

$$f_1(x) = 2\sqrt{x_1} \tag{6-11}$$

$$f_2(x) = x_1(1 - x_2) + 5 \tag{6-12}$$

The Pareto front of this problem is non-convex, as shown in Figure 6-11, and can be analytically given by:

$$f_2(x) = 5 - \frac{{f_1}^2(x)}{4} \tag{6-13}$$

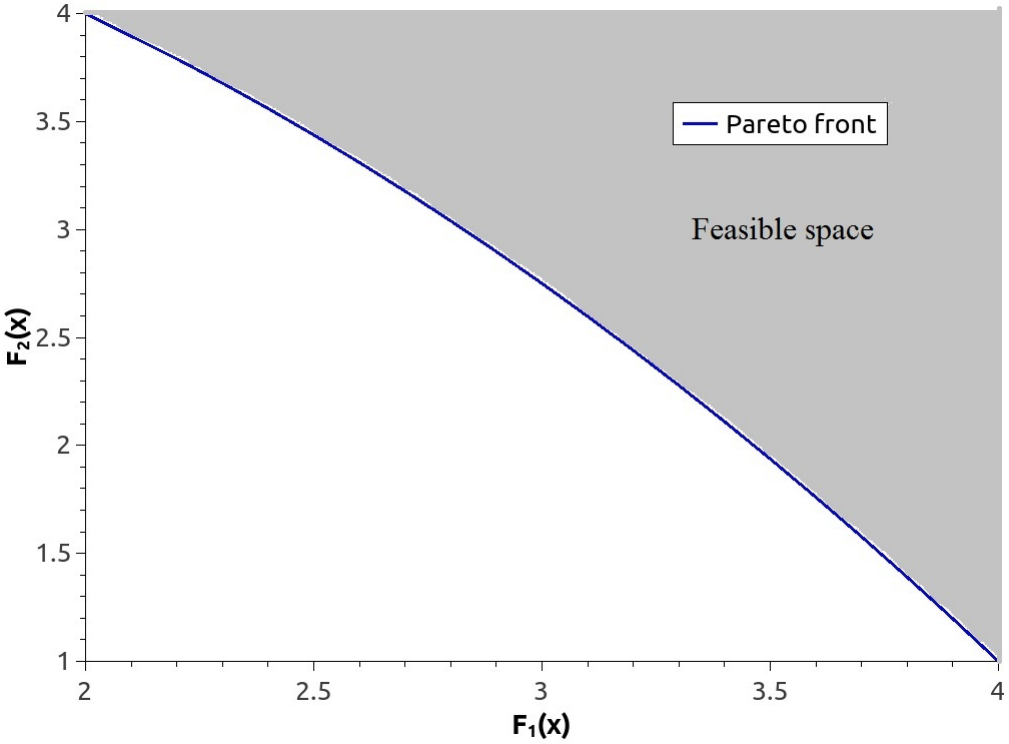

Figure 6.11: The Pareto front and feasible domain of the Murata problem

In order to set $F_1(x)$ range from zero to one, the objective function is subtracted by two and then divided by two. For $F_2(x)$, the function is subtracted by one and then divided by three. To solve this problem with FSQP, the inputs for the algorithm are number of points (50), minimum weight for WSF (0.02), and weight increase (0.02). The

results of the algorithm are shown in Figure 6-12 and the performance statistics are given in Table 6-3.

Table 6-3: Murata problem results

	FSQP	NSGA-II	Hybrid
RMSE	6.26E-02	2.04E-06	1.16E-04
PF	3.62	0.36	0.58
Execution time	0.04s	0.49s	0.02s
Number of solutions	5	50	50

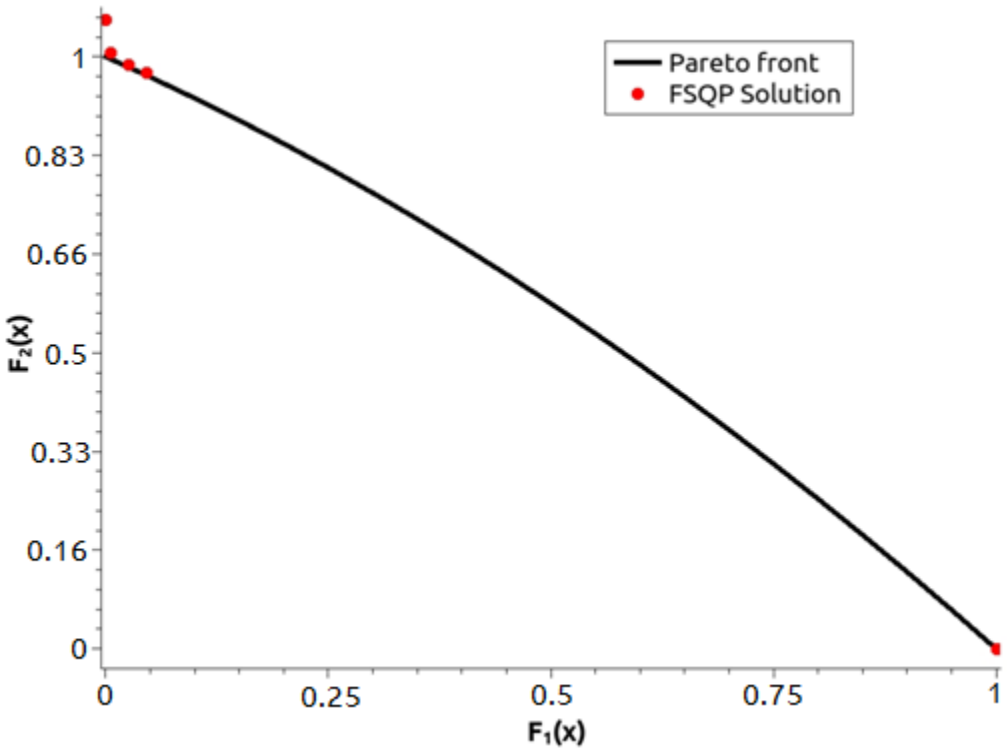

Figure 6.12: The Murata problem solved by FSQP

It can be seen from Figure 6.12 that FSQP is unable to properly solve this problem because of the concavity of the Pareto front.

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the number of generations that yields the best solutions, the problem was solved with different maximum numbers of generations and .The results are shown in Figure 6-13.

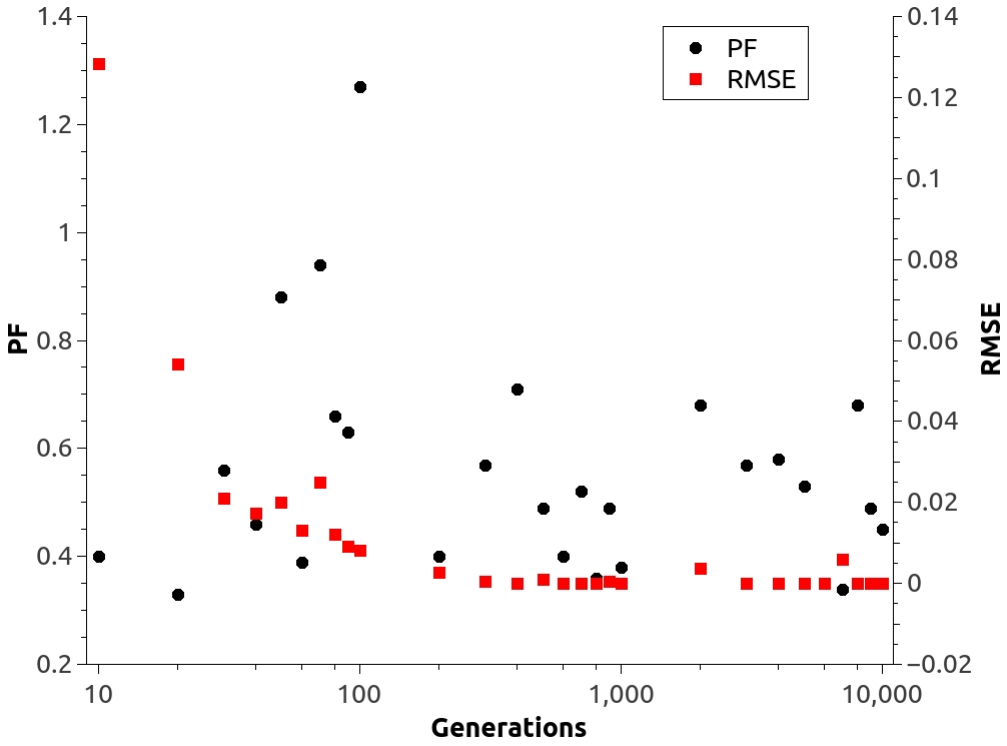

Figure 6.13: The PF and RMSE vs. number of generations for the Murata problem solved by NSGA-II

From Figure 6.13, it is determined that the adequate number of generations is 800. The results of the algorithm are shown in Figure 6-14 and the performance statistics of NSGA-II are given in Table 6-3.

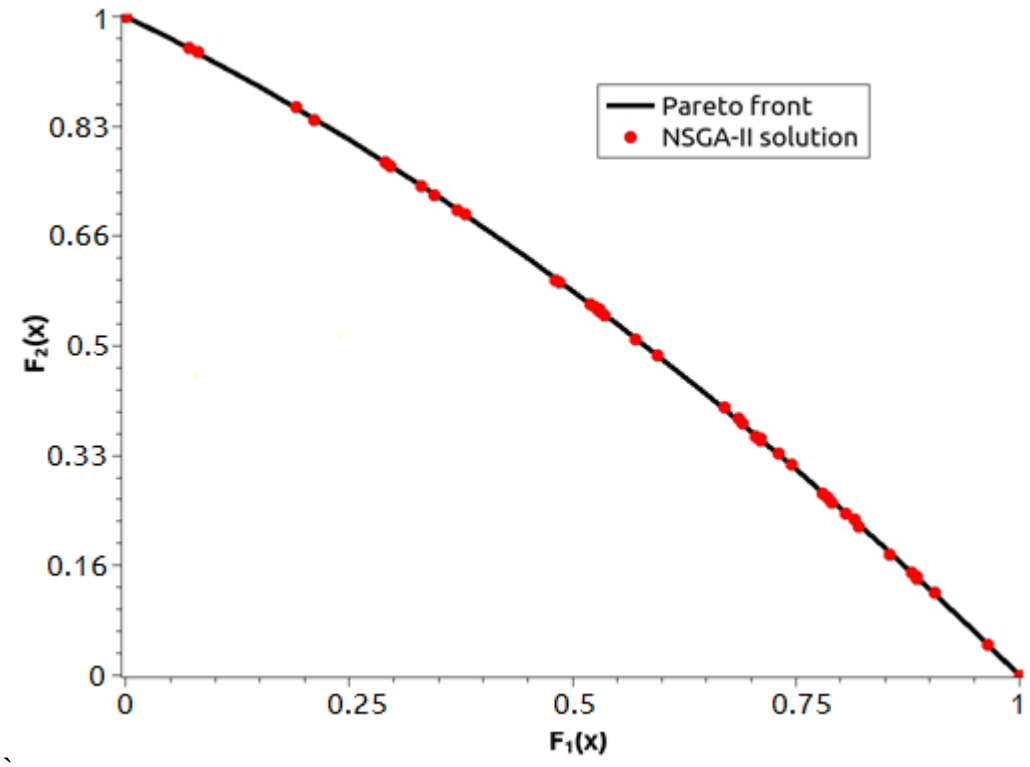

Figure 6.14: The Murata problem solved by NSGA-II

The inputs for the hybrid algorithm are the number of initial points for NSGA-II (50), the maximum number of generations for NSGA-II (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The results of the solutions for this problem are shown in Figure 6-15, and the performance statistics are given in Table 6-3.

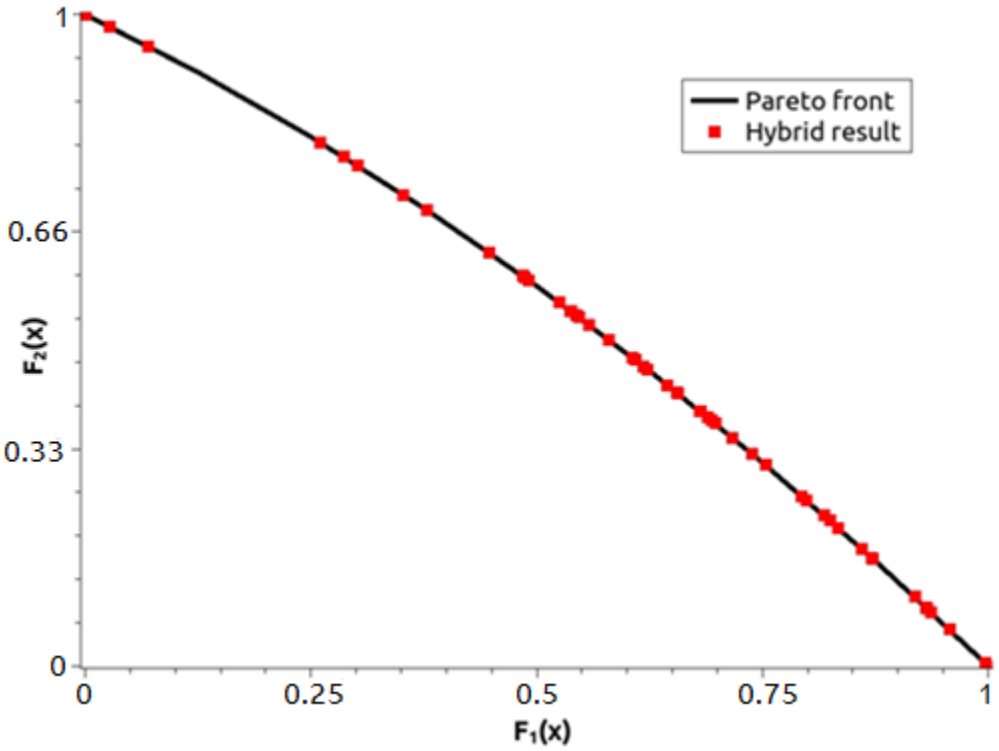

Figure 6.15: The Murata problem solved by the hybrid algorithm

The optimization results of the Murata problem show once again that FSQP is unable to obtain a satisfactory set of solutions for optimization problems with non-convex Pareto front. This can be seen from the highest PF of FSQP and the least number of solutions. The hybrid algorithm gives an error that is 100 times larger than that by NSGA-II. Furthermore, the hybrid algorithm has a larger PF than that of NSGA-II yet both PFs are significantly smaller than that of FSQP. Finally, the hybrid algorithm is about 25 times faster than NSGA-II.

6.4. The Rendon Problem

Introduced by Rendon (1997), this is a minimization problem of two objectives. The problem is stated as follows.

Min $$\left(f_1(x), f_2(x)\right) \quad (6\text{-}14)$$

$$f_1(x) = {}^{1}/_{(x_1^2 + x_2^2 + 1)} \quad (6\text{-}15)$$

$$f_2(x) = x_1^2 + 3x_2^2 + 1 \quad (6\text{-}16)$$

$$-3 \geq x_{1,2} \geq 3 \quad (6\text{-}17)$$

The Pareto front, as shown in Figure 6-16, is convex and can be given explicitly by the following equation:

$$f_2(x) = \frac{1}{f_1(x)} \quad (6\text{-}18)$$

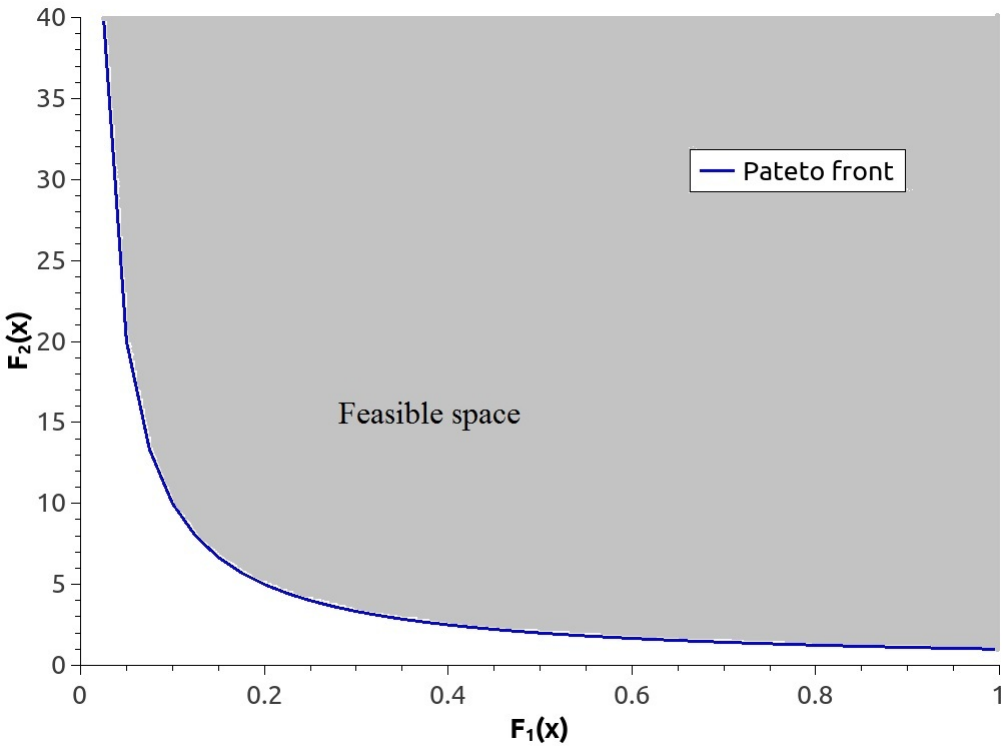

Figure 6.16: The Pareto front and feasible domain of the Rendon problem

For this problem, $F_2(x)$ is divided by 40 in order to keep the range of the objective functions between zero and one. To solve this problem with FSQP, the inputs to

the algorithm are the number of initial points (50), minimum weight for WSF (0.02) and the weight increment (0.02). The results of this problem are shown in Figure 6-17 and the performance statistics are given in Table 6-4.

Table 6-4: Rendon problem results

	FSQP	NSGA-II	Hybrid
RMSE	8.36	8.12	9.23
PF	10.37	7.58	8.21
Execution time	0.12s	0.05s	0.06s
Number of solutions	41	50	50

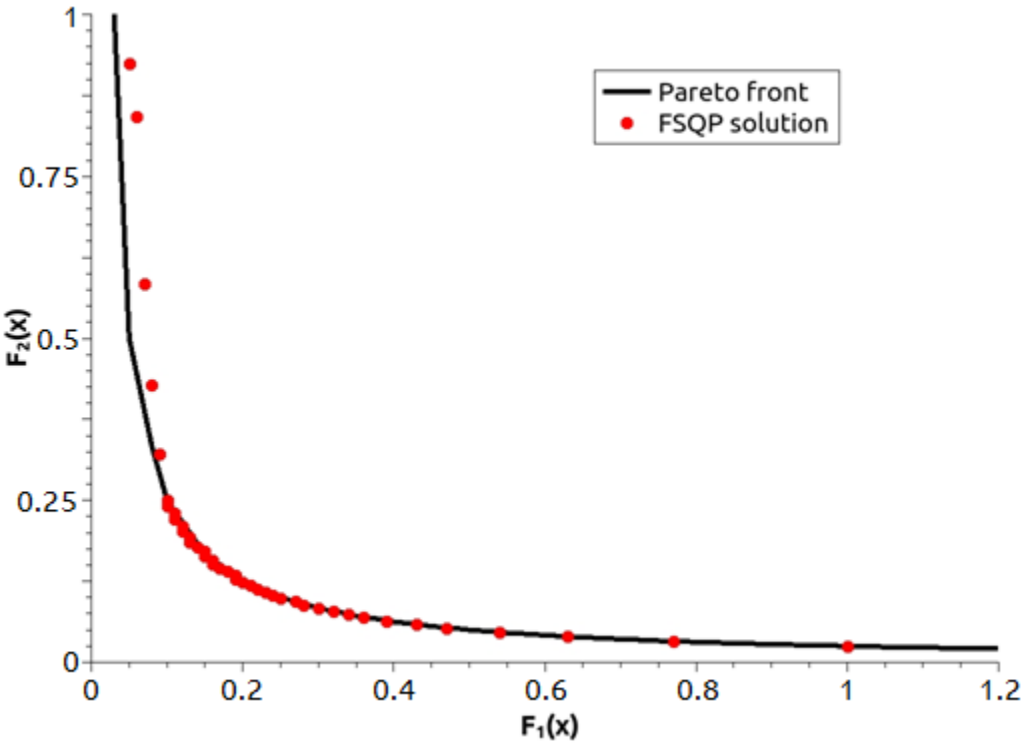

Figure 6.17: The Rendon problem solved by FSQP

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the maximum number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-18.

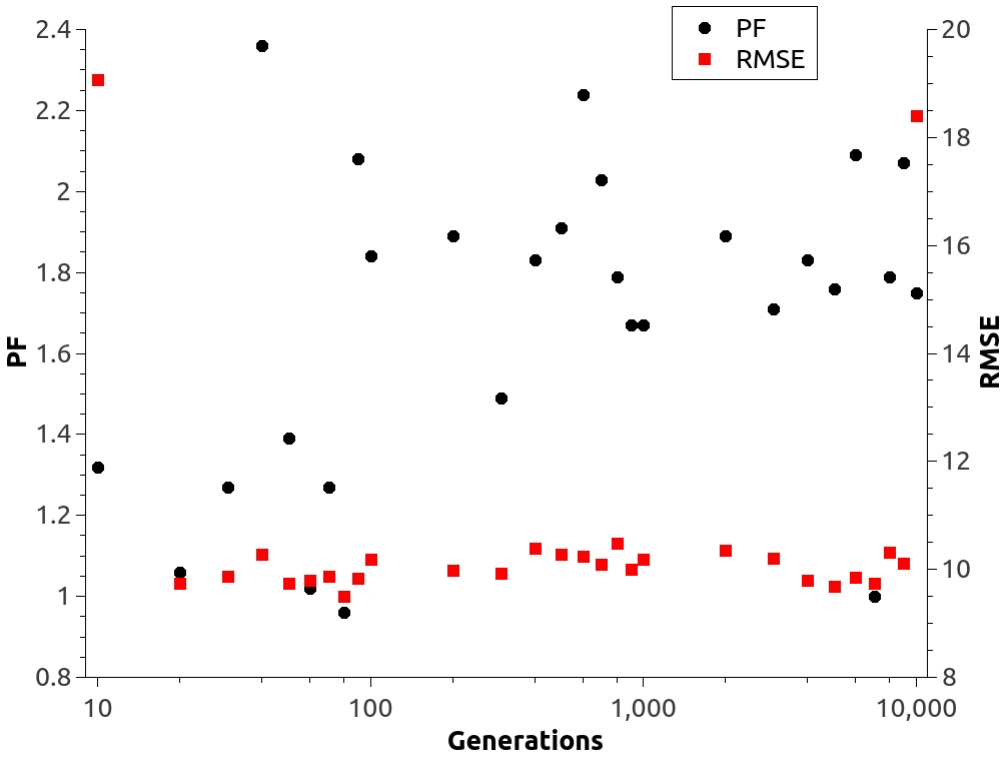

Figure 6.18: The PF and RMSE vs. number of generations

for the Rendon problem solved by NSGA-II

According to Figure 6.18 the best solutions are obtained at 80 generations. The problem's solution is shown in Figure 6-19, and the performance statistics given in Table 6-4.

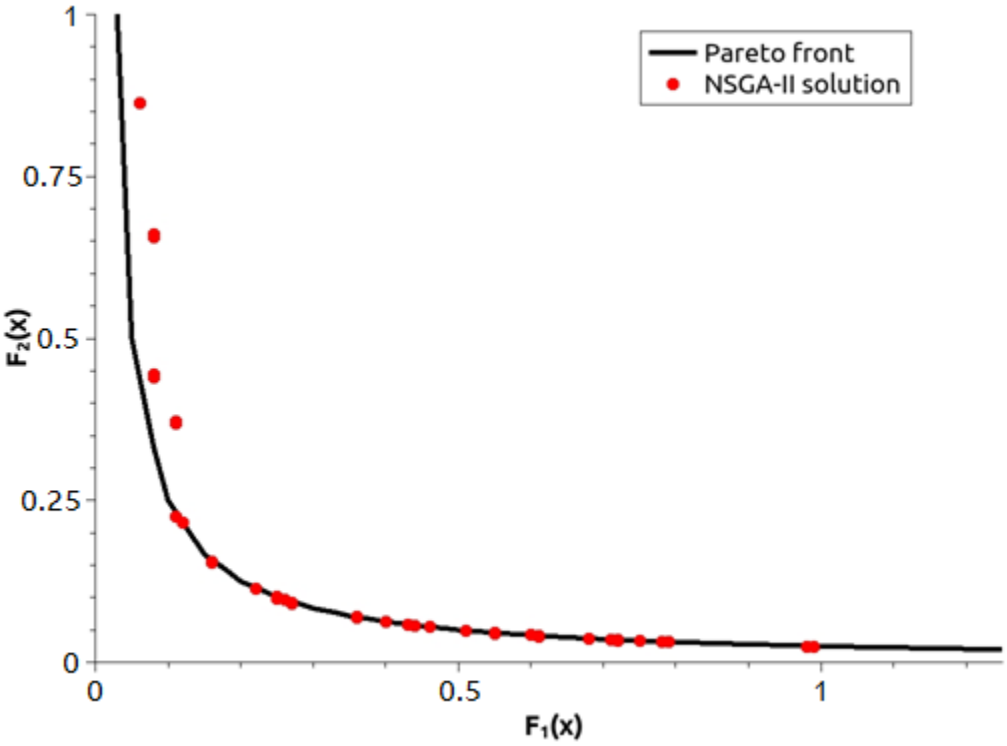

Figure 6.19: The Rendon problem solved by NSGA-II

The inputs for the hybrid algorithm are the initial points for NSGA-II (50), maximum number of generations (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The results of the hybrid algorithm are shown in Figure 6-20, and the performance statistics are summarized in Table 6-4.

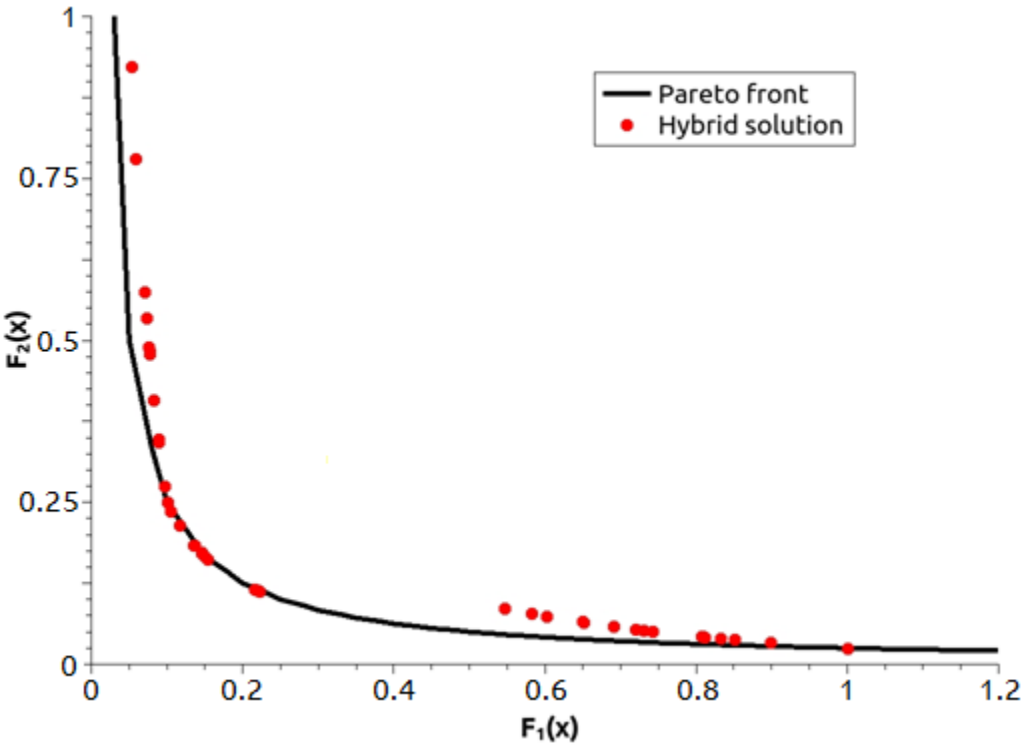

Figure 6.20: The Rendon problem solved by the Hybrid algorithm

This problem has a Pareto front with steep slopes, which makes the problem to have a very high RMSE when solved by the three algorithms. The PFs are also relatively high when it is considered that the three algorithms to some extent are able to have a good solution along the Pareto front. Furthermore, the execution time of NSGA-II and the hybrid algorithm are along the same value (0.01s difference), and the solution time by FSQP doubles that by the hybrid algorithm.

6.5. The Rendon 2 Problem

Introduced by Rendon (1997), this is a minimization problem of two objectives. The problem is stated as follows.

Min $\left(f_1(x), f_2(x)\right)$ (6-19)

$$f_1(x) = x_1 + x_2 + 1 \quad (6\text{-}20)$$

The Pareto front is a convex given by:

$$f_2(x) = {f_1}^2(x) + 4f_1(x) - 3 \quad (6\text{-}21)$$

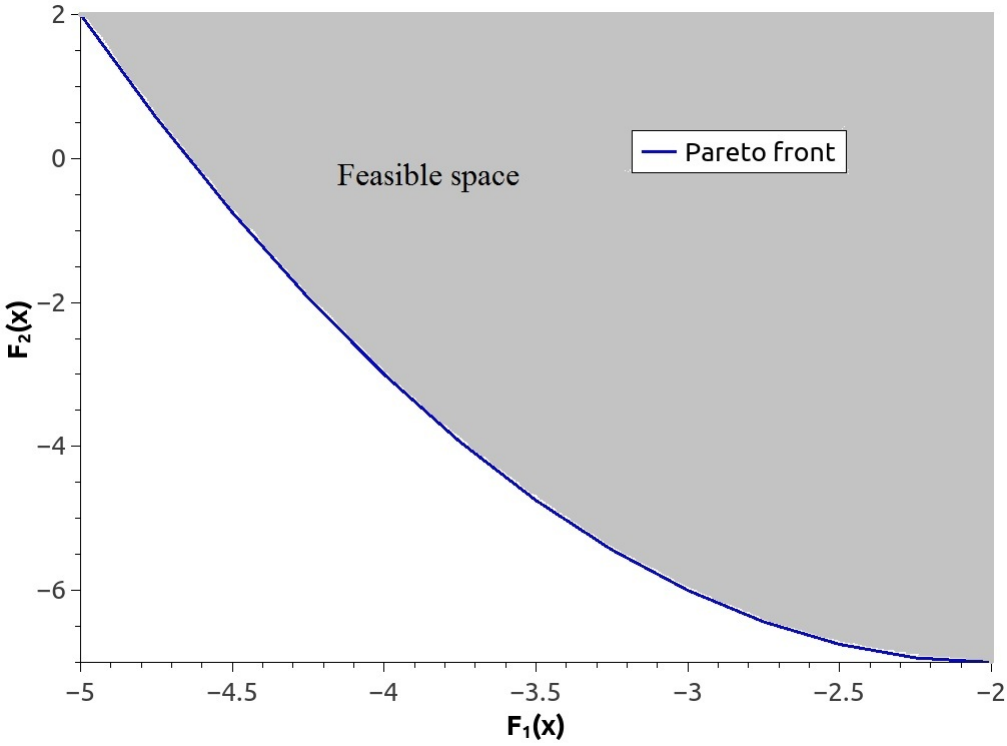

Figure 6.21: The Pareto front and feasible domain of the Rendon 2 problem

In order to normalize the objective functions, $f_1(x)$ is added 5 and divided by 3, and $f_2(x)$ is added 7 and divided by 9. To solve this problem with FSQP, the number of initial points is set to (50), the minimum weight is set to (0.02) and the weight increase is selected as (0.02). The solution to this problem is shown in Figure 6-22 and the performance statistics are given in Table 6-5.

Table 6-5: Rendon 2 problem results

	FSQP	NSGA-II	Hybrid
RMSE	1.22E-02	1.71E-05	1.54E-04
PF	2.14	3.44	1.15
Execution time	0.4s	0.3s	0.07s
Number of solutions	44	50	50

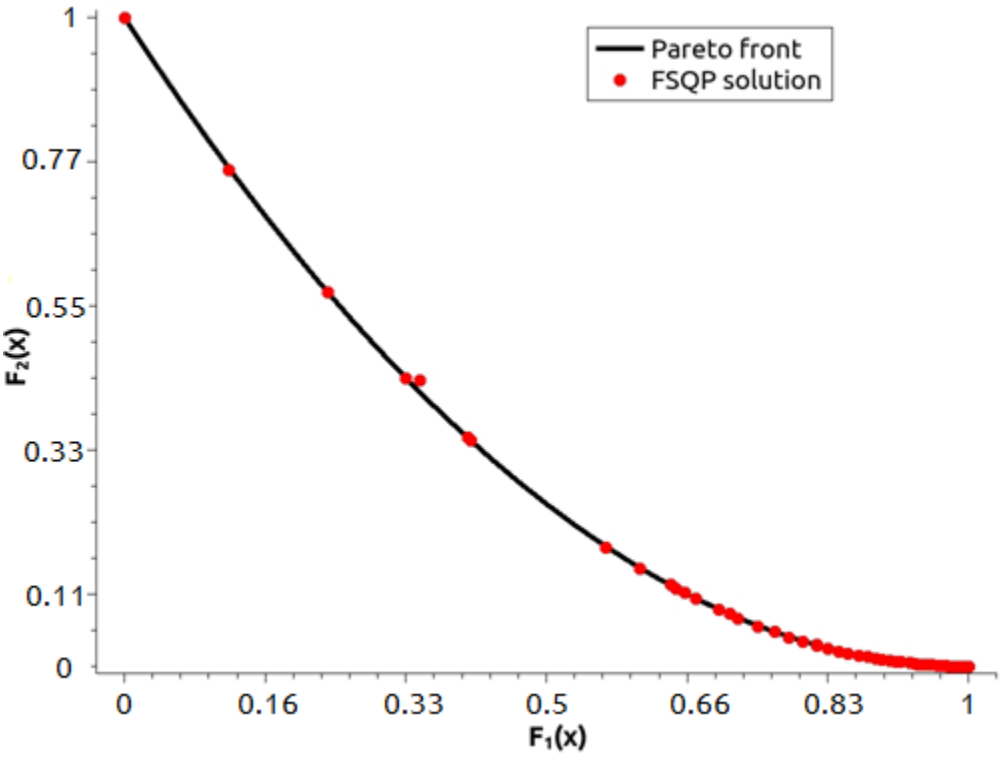

Figure 6.22: The Rendon 2 problem solved by FSQP

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the maximum

number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-23.

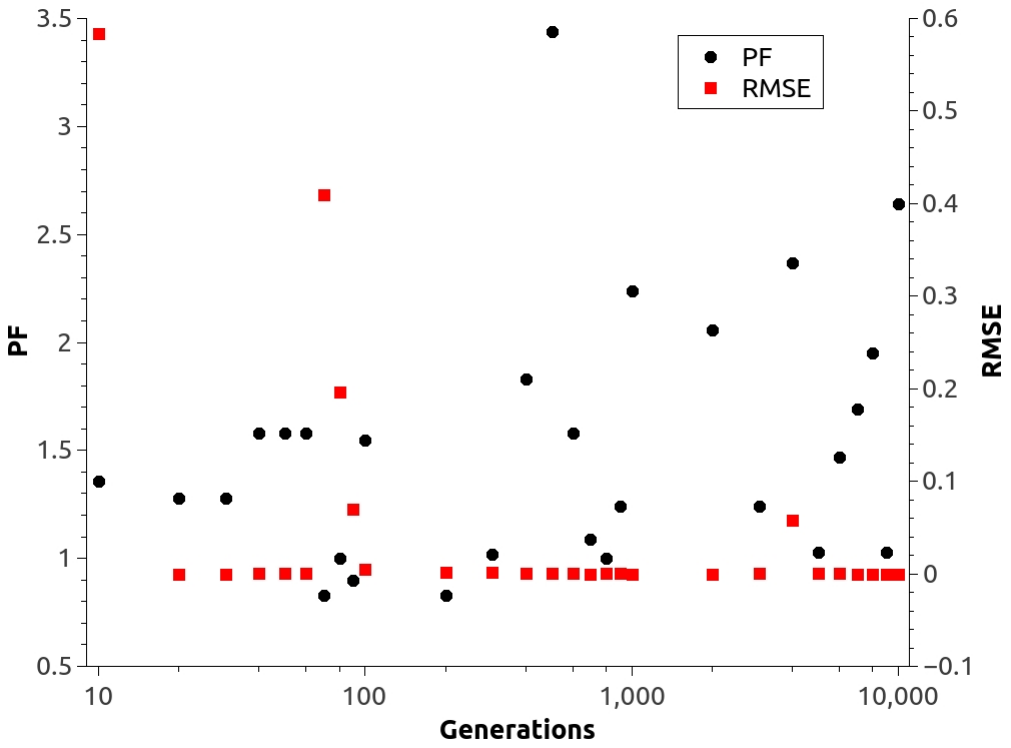

Figure 6.23: The PF and RMSE vs. number of generations for the Rendon 2 problem solved by NSGA-II

From Figure 6.23, it is determined that the best solutions are obtained at 700 generations. The results of the algorithm are shown in Figure 6-24, and the performance statistics are given in Table 6-5.

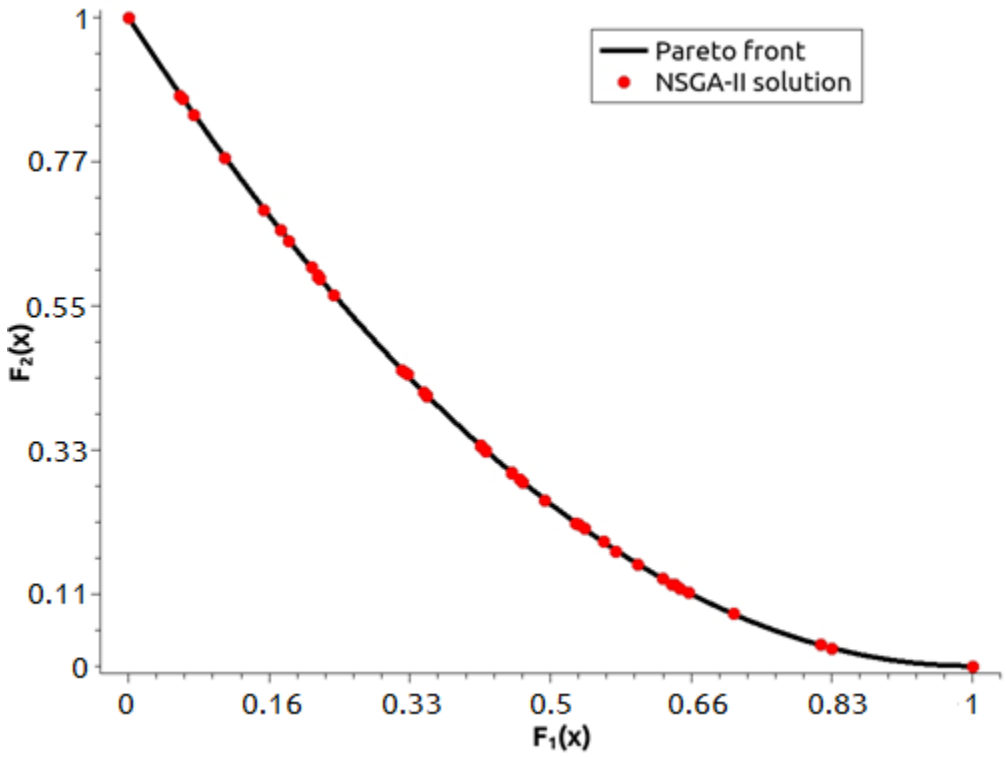

Figure 6.24: The Rendon 2 solved by NSGA-II

The inputs for the hybrid algorithm are the initial points for NSGA-II (50), maximum number of generations (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The plot of the solutions for this problem is shown in Figure 6-25 and the performance statistics are summarized in Table 6-5.

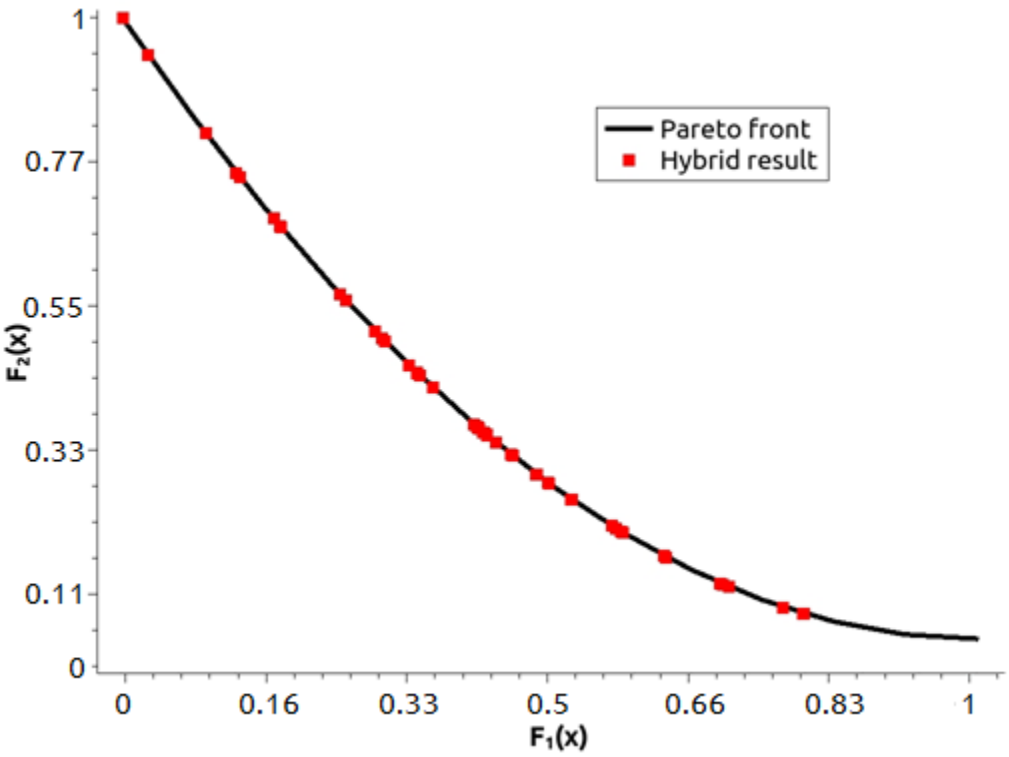

Figure 6.25: The Rendon 2 problem solved by the hybrid algorithm

Solving the Rendon 2 problem, it can be concluded that the RMSE of NSGA-II is the lowest, which is approximately ten times smaller than the hybrid algorithm, and the RMSE of FSQP is approximately 80 times larger than the one obtained by the hybrid algorithm. The lowest PF is the one from the hybrid algorithm followed by FSQP and the largest value is obtained from NSGA-II. Finally, regarding the execution time, the fastest is the hybrid algorithm followed by NSGA-II whose time is about three times more than the hybrid algorithm. The FSQP has the most time.

6.6. The ZDT-3 Problem

Introduced by Zitzler et al (2000), this problem is an unconstrained optimization problem with two objectives and thirty design variable. The problem is stated as follows.

$$\text{Min} \quad \left(f_1(x), f_2(x)\right) \tag{6-22}$$

$$f_1(\vec{x}) = x_1 \tag{6-23}$$

$$f_2(\vec{x}) = 1 - \sqrt{{x_1}/{g(\vec{x})}} - \left({x_1}/{g(\vec{x})}\right)\sin(10\pi x_1) \tag{6-24}$$

where

$$g(\vec{x}) = 1 + 9\,{\sum_{i=2}^{n} x_i}/{(n-1)},\ n = 30 \tag{6-25}$$

The Pareto front is obtained at $g(x) = 1$; it is discontinuous, though the feasible domain is continuous as shown in Figure 6-26

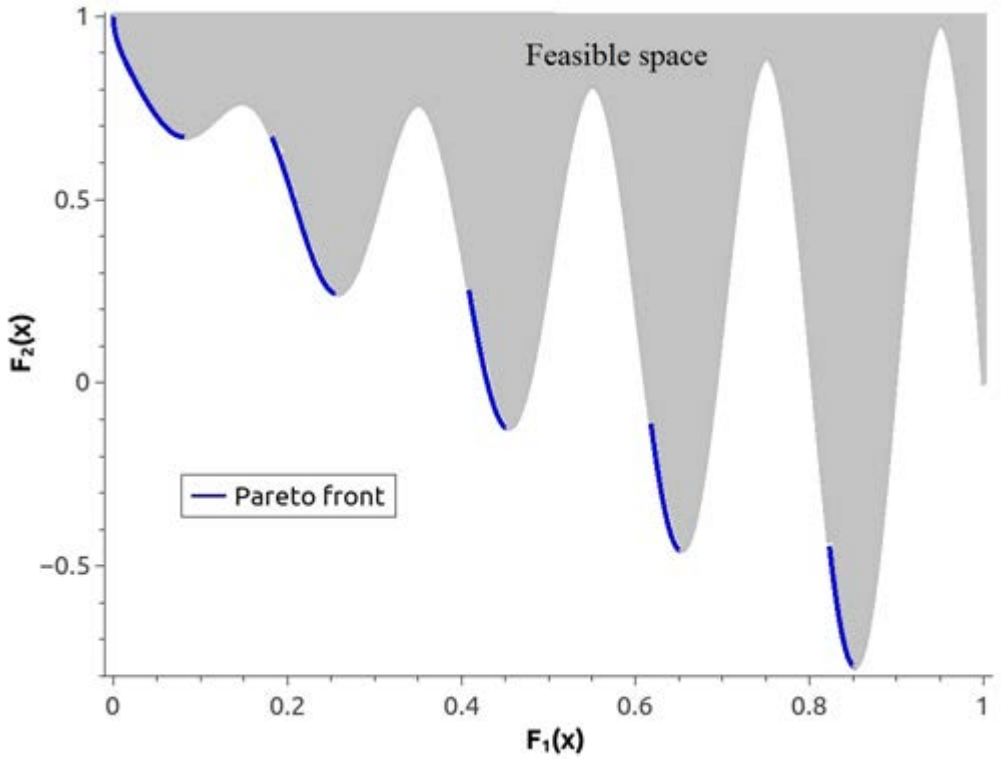

Figure 6.26: The Pareto front and feasible domain of the ZDT-3 problem

To solve this problem with FSQP, the number of initial points is set to (50), the minimum weight for WSF is (0.02) and the weight increment is set to (0.02). The results

of the algorithm are shown in Figure 6-27 and the performance statistics are given in Table 6-6. This algorithm is unable to solve properly the problem because of the discontinuity of the Pareto front.

Table 6-6: ZDT-3 problem results

	FSQP	NSGA-II	Hybrid	Hybrid 2
RMSE	2.40E-06	2.69E-04	0.198	4.60E-01
PF	0.28	0.18	0.34	0.19
Execution time	34.21s	1.5s	1.71s	4.78s
Number of solutions	2	50	20	27

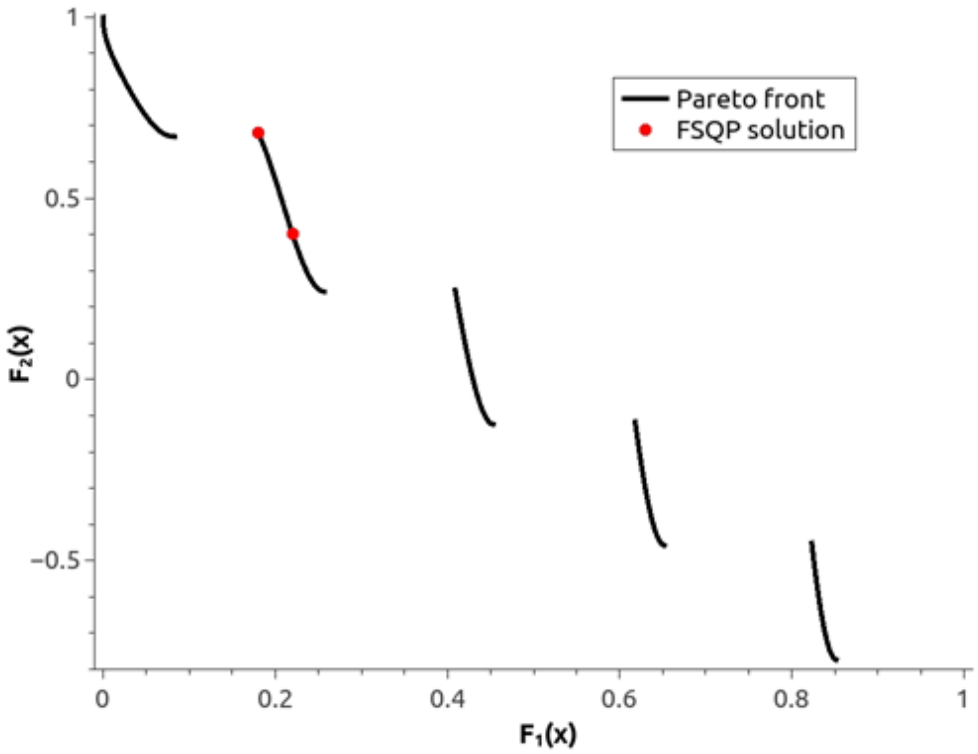

Figure 6.27: The ZDT-3 problem solved by FSQP

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the maximum number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-28.

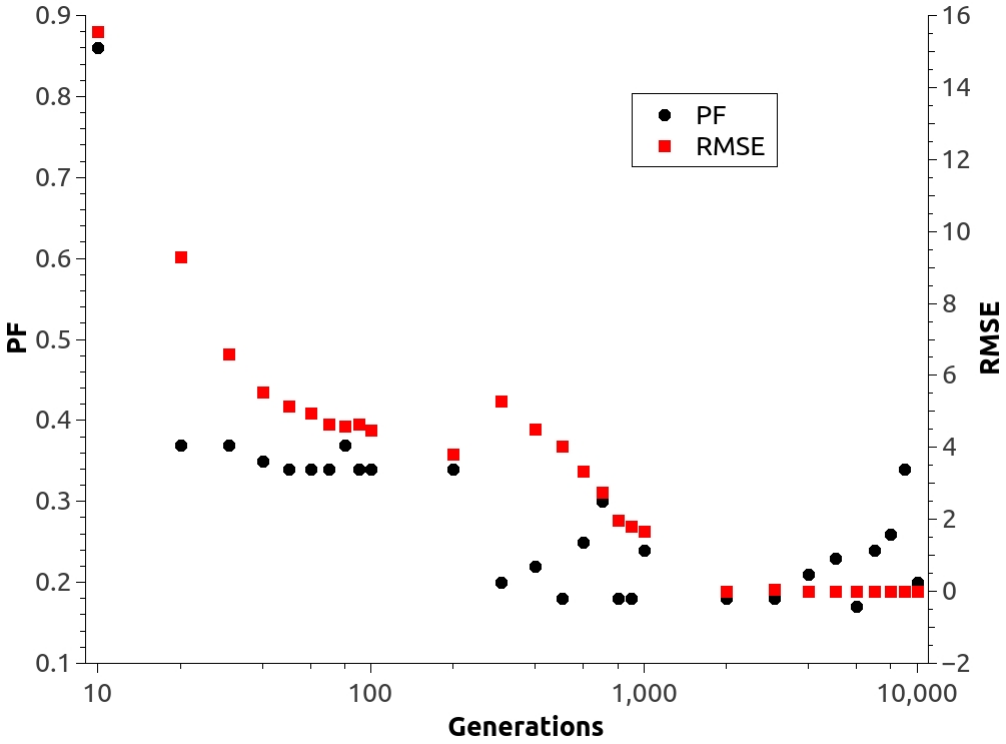

Figure 6.28: The PF and RMSE vs. number of generations for the ZDT-3 problem solved by NSGA-II

According to Figure 6.28, the best solutions are obtained at 1000 generations. The results of the algorithm are shown in Figure 6-29, and the performance statistics are given in Table 6-6.

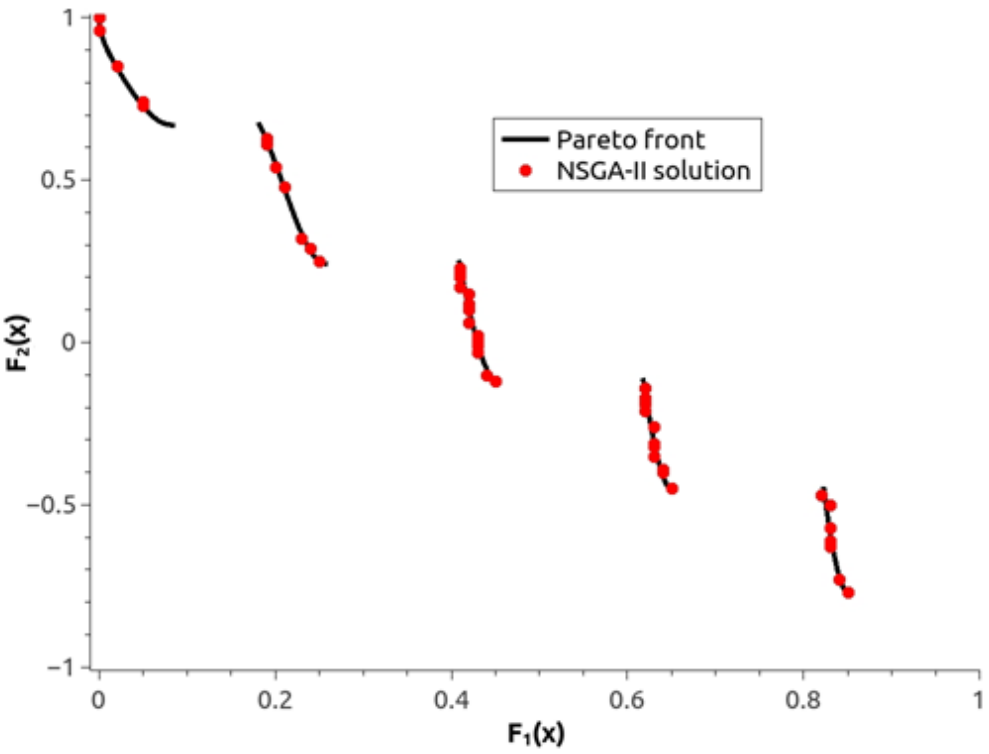

Figure 6.29: The ZDT-3 problem solved by NSGA-II

The inputs for the hybrid algorithm are the number of initial points (50), maximum number of generations for NSGA-II (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The plot of the solutions for this problem is shown in Figure 6-30, and the performance statistics are summarized in Table 6-6.

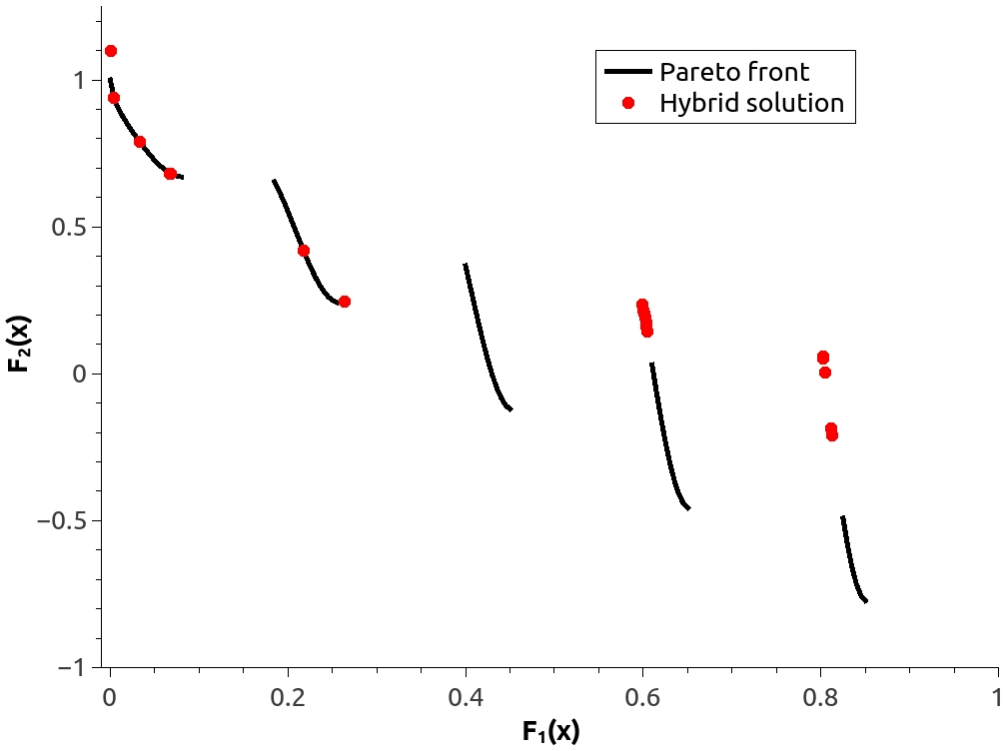

Figure 6.30: The ZDT-3 problem solved by the hybrid algorithm

As seen on the above figure almost all the solutions to this problem are off the true Pareto front. To overcome this, the parameters of initial points and maximum number of generations of the NSGA-II stage, these are set to 50 initial points and 5000 generations. The results are shown in Figure 6-31 and the performance statistics given in Table 6-6.

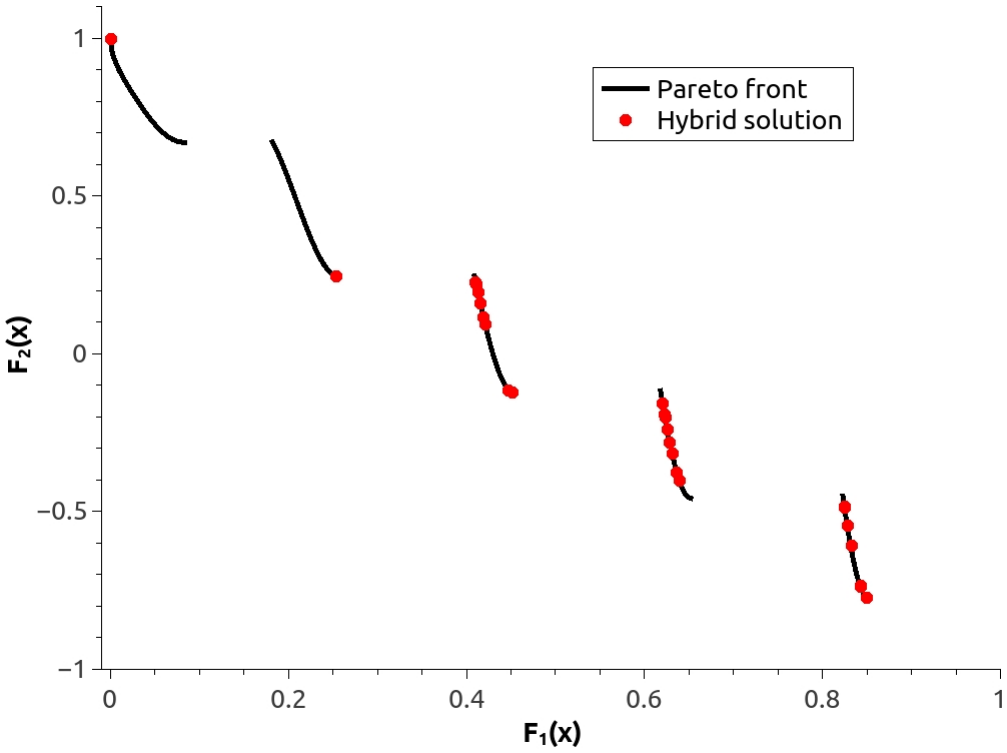

Figure 6.31: The ZDT-3 problem solved by hybrid algorithm, second attempt

The solutions to the ZDT-3 problem show that FSQP is unable to provide adequate solutions due to the limitation of its capabilities to solve problems with discontinuous Pareto fronts. NSGA-II has the smallest error and the best distribution of the solutions. The hybrid algorithm does not have better performance than NSGA-II in this case; this is primarily caused by the second stage of the algorithm, which uses FSQP and thus is unable to obtain solutions on the Pareto front. However, it is observed that by increasing the number of generations for the NSGA-II portion, the solution quality of the hybrid algorithm can be improved. This is mainly attributed to the fact that many of the initial points for the second stage are already close to the Pareto front.

6.7. The Bihn 2 Problem

Introduced by Bihn (1997), this constrained minimization problem has two objective functions, two design variables and two constraints. The problem is stated as follows.

$$\text{Min} \quad \big(f_1(x), f_2(x)\big) \tag{6-26}$$

$$f_1(x) = 4x_1^2 + 4x_2^2 \tag{6-27}$$

$$f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \tag{6-28}$$

$$\text{s.t.} \quad (x_1 - 5)^2 + x_2^2 - 25 \leq 0 \tag{6-29}$$

$$-(x_1 - 8)^2 - (x_2^2 + 3)^2 + 7.7 \leq 0 \tag{6-30}$$

The Pareto front is shown in Figure 6-32 and its equation is given by:

$$f_2(x) = 2\big(\sqrt{f_1(x)/8} - 5\big)^2 \tag{6-31}$$

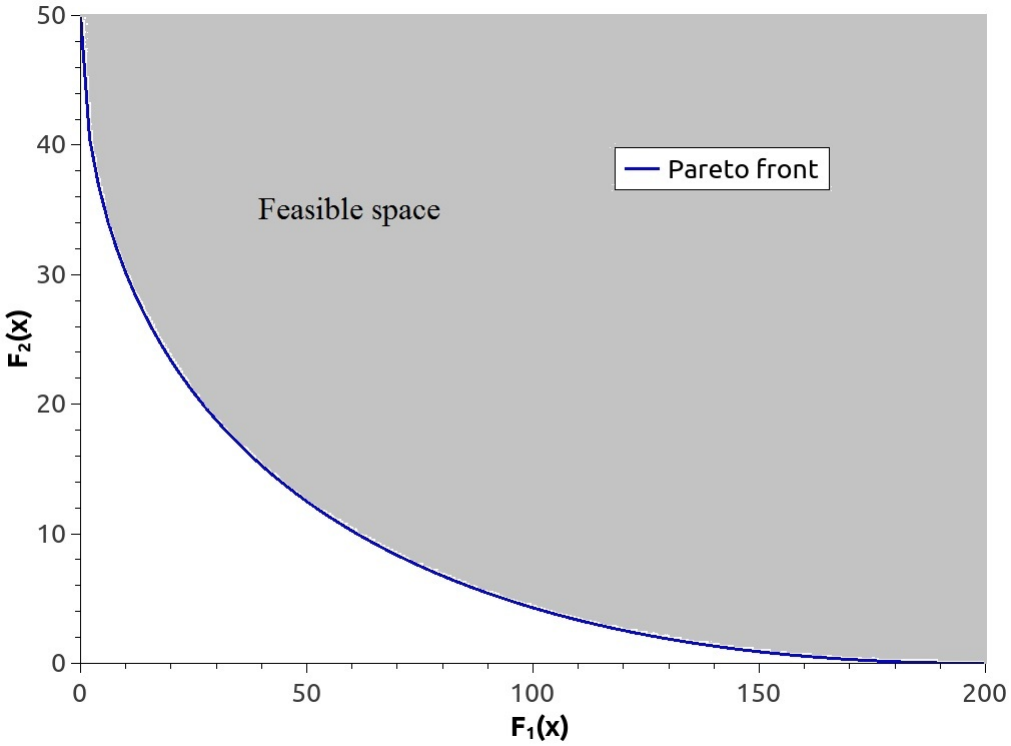

Figure 6.32: The Pareto front and feasible domain of the Bihn 2 problem

In order to set the proper ranges of both objective functions between zero and one, $F_1(x)$ is divided by 200 and $F_2(x)$ is divided by 50. The inputs for solving this problem using FSQP are number of initial points (50), minimum weight for WSF (0.02), and weight increase of (0.02). The solution is shown in Figure 6-33, and the performance statistics are given in Table 6-7.

Table 6-7: Bihn 2 problem results

	FSQP	NSGA-II	Hybrid
RMSE	3.43	1.583	5.59
PF	12.01	10.86	12.66
Execution time	0.1s	0.0s	0.03s
Number of solutions	49	50	50

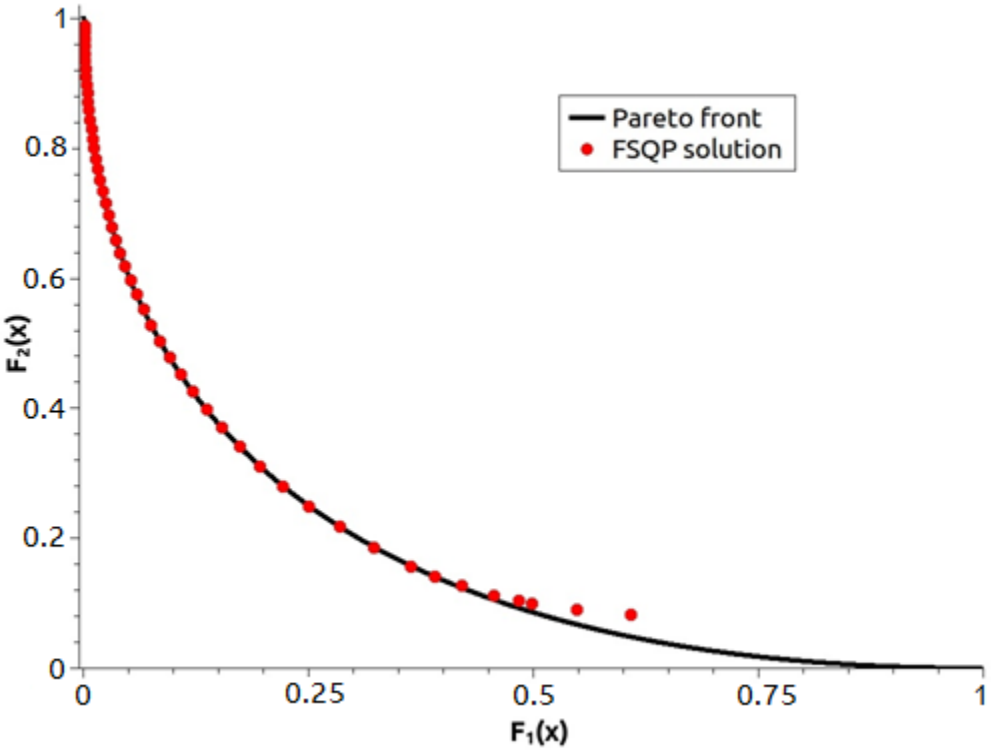

Figure 6.33: The Bihn 2 problem solved by FSQP

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the maximum number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-34.

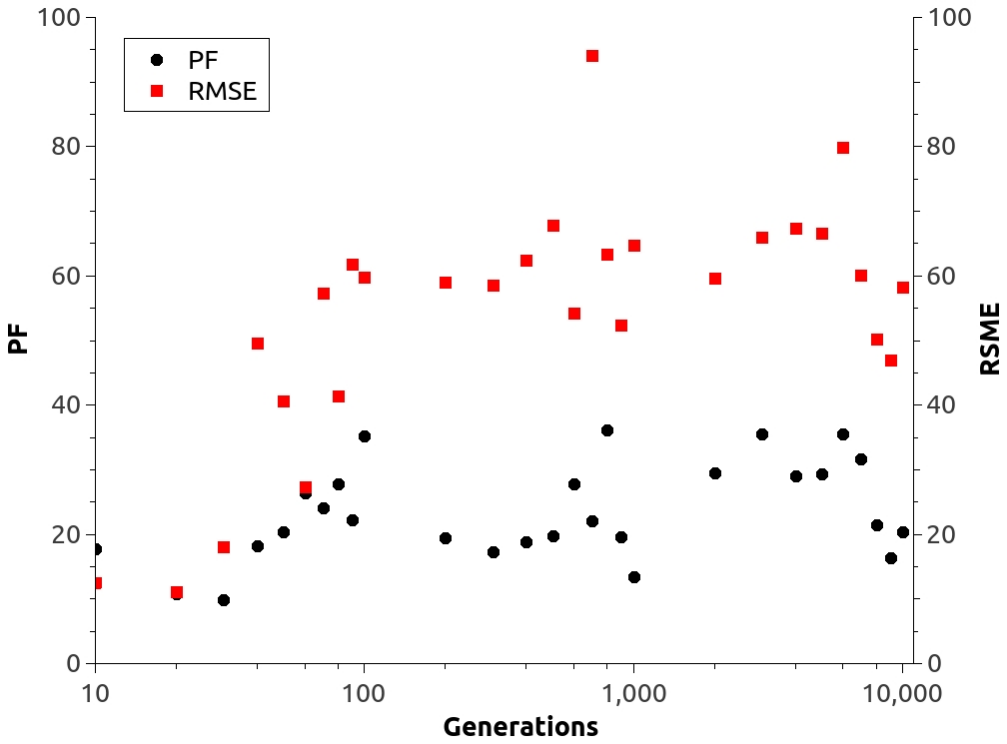

Figure 6.34: The PF and RMSE vs. number of generations for the Bihn 2 problem solved by NSGA-II

From Figure 6.34, it is determined that the best solutions are obtained at 20 generations. The results of the algorithm are shown in Figure 6-35 and the performance statistics are given in Table 6-7.

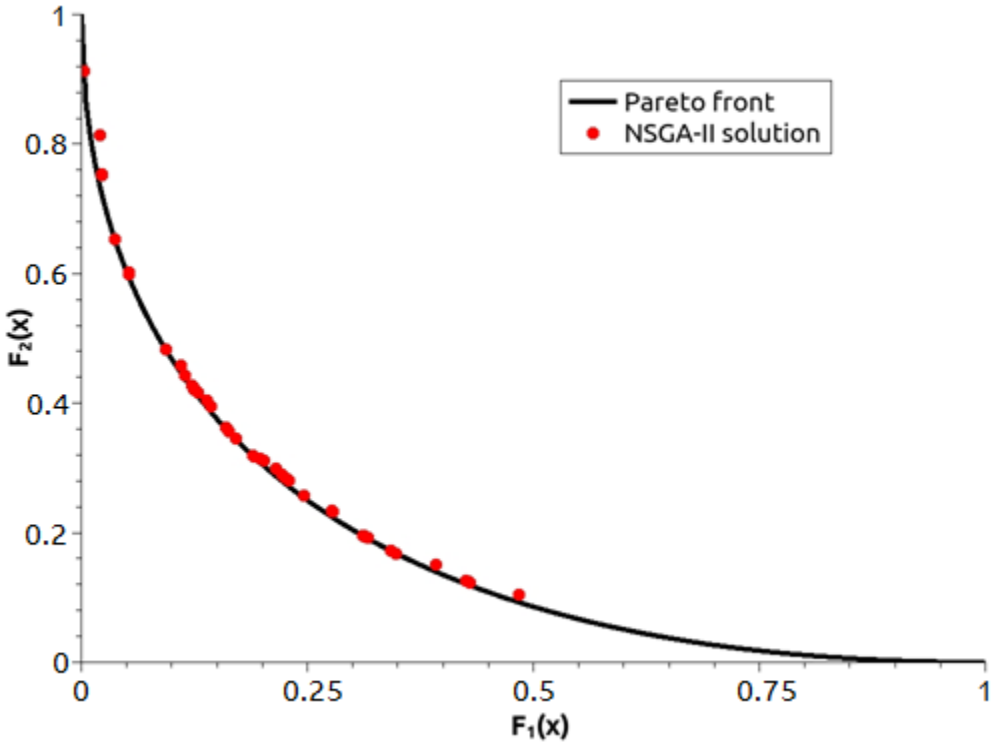

Figure 6.35: The Bihn 2 problem solved by NSGA-II

The settings for the hybrid algorithm are the number of initial points for NSGA-II (50), maximum number of generations (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The results of this algorithm are shown in Figure 6-36, and the performance statistics are given in Table 6-7.

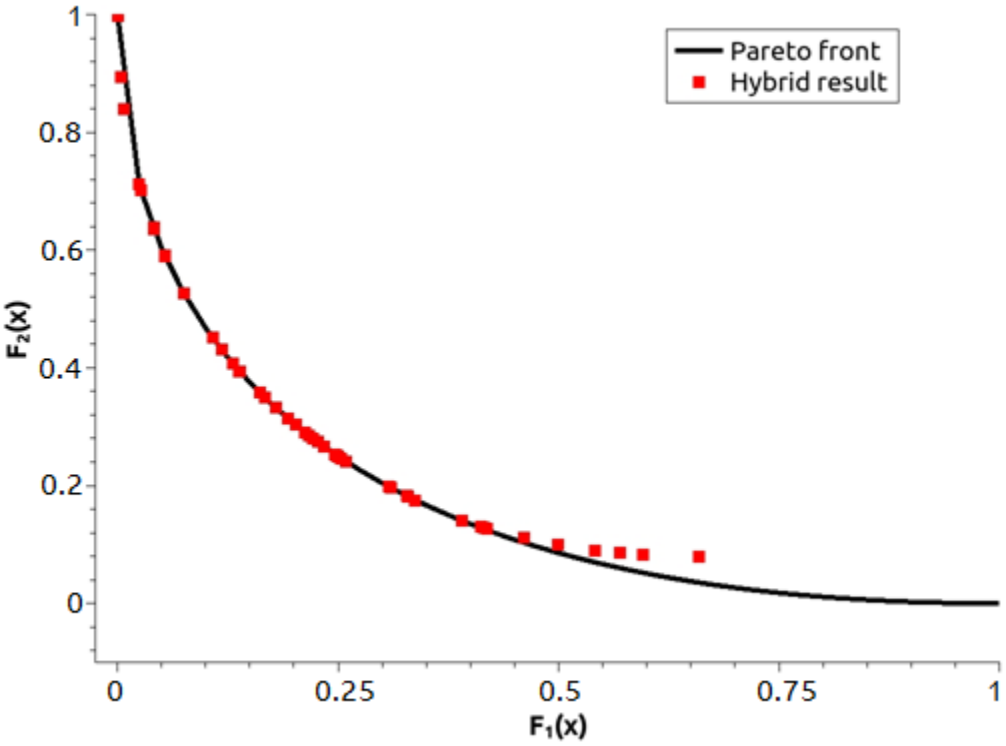

Figure 6.36: The Bihn 2 problem solved by the hybrid algorithm

It can be concluded that the smallest RMSE for the Bihn 2 problem is obtained by NSGA-II, followed by the error obtained by FSQP which is approximate twice as much; the largest RSME is obtained from the hybrid algorithm which is about twice as much as FSQP. The piling factor given by the three algorithms is similar, though NSGA-II has the smallest value. Regarding the execution time NSGA-II took less than 0.01 seconds followed by the hybrid algorithm that took three times as much and finally FSQP took a tenth of a second.

6.8. The Jimenez Problem

Introduced by Jimenez (1998), this constrained maximization problem has two objective functions, and three constraints.

Max $\quad \big(f_1(x), f_2(x)\big)$ (6-32)

$$f_1(x) = 5x_1 + 3x_2 \quad (6\text{-}33)$$

$$f_1(x) = 2x_1 + 8x_2 \quad (6\text{-}34)$$

s.t. $\quad x_1 + 4x_2 \leq 100$ (6-35)

$$3x_1 + 2x_2 \leq 150 \quad (6\text{-}36)$$

$$2x_1 + 8x_2 \geq 200 \quad (6\text{-}37)$$

The Pareto front is a straight line section given by the following equation and shown in Figure 6-37:

$$f_2(x) = 20f_1(x) - 5100 \quad (6\text{-}38)$$

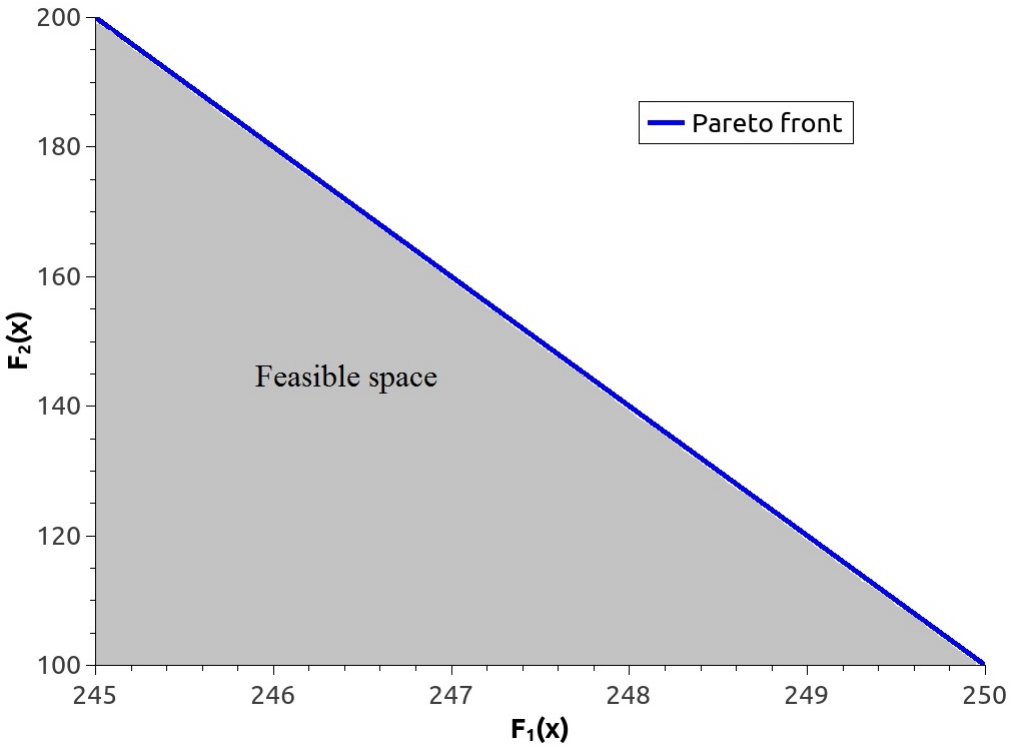

Figure 6.37: The Pareto front and feasible domain of the Jimenez problem

To solve this problem as a minimization problem as seen in section 2.3, the objective functions need to be multiplied by -1 and then normalized. To normalize the objectives, $F_1(x)$ is added by 250 and divided by 5. $F_2(x)$ is added 200 and divided by 100. To solve this problem with FSQP, the inputs for the algorithm are the number of initial points (50), minimum weight for WSF (0.02) and the weight increment (0.02). The solution to this problem is shown in Figure 6-38 and the performance statistics are given in Table 6-8.

Table 6-8: Jimenez problem results

	FSQP	NSGA-II	Hybrid
RMSE	0	0.19	0.24
PF	100.12	6	52.25
Execution time	0.15s	3.12s	0.04s
Number of solutions	2	50	23

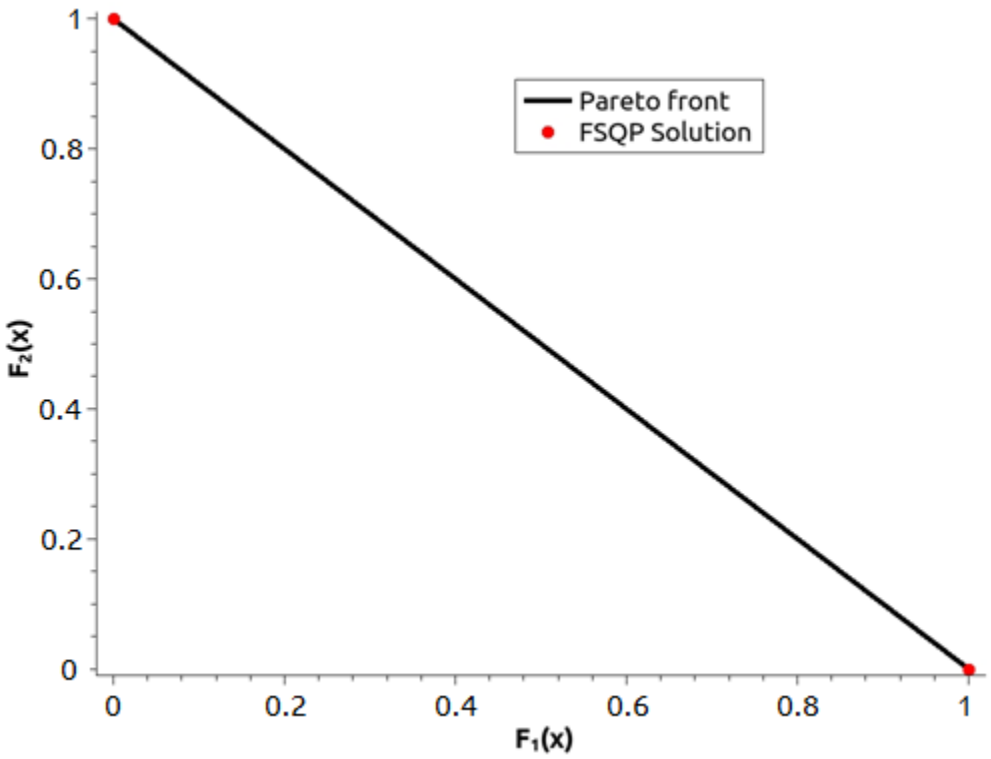

Figure 6.38: The Jimenez problem solved by FSQP

The inputs for solving this problem with NSGA-II are the number of initial points (50), crossover rate (0.9), and mutation rate (0.01). In order to determine the number of generations that yields the best solutions, the problem is solved with different maximum numbers of generations and the results are shown in Figure 6-39.

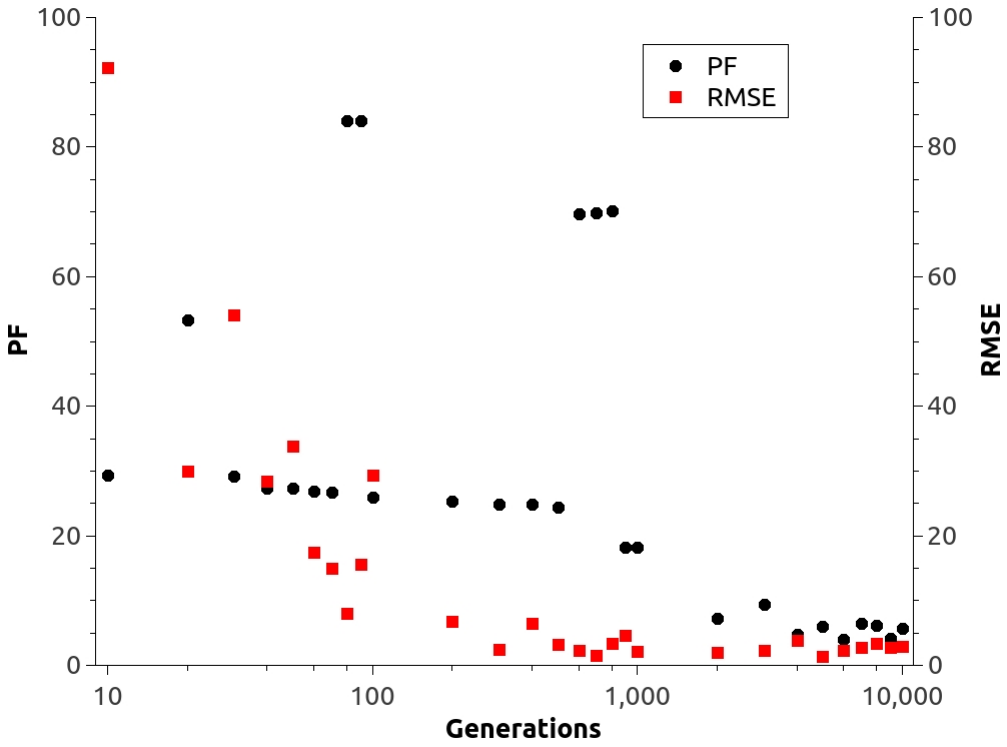

Figure 6.39: The PF and RMSE vs. number of generations for the Jimenez problem solved by NSGA-II

According to the Figure 6.39, the best solutions are obtained at 1000 generations. The results of the algorithm are shown in Figure 6-40, and the performance statistics are given in Table 6-8.

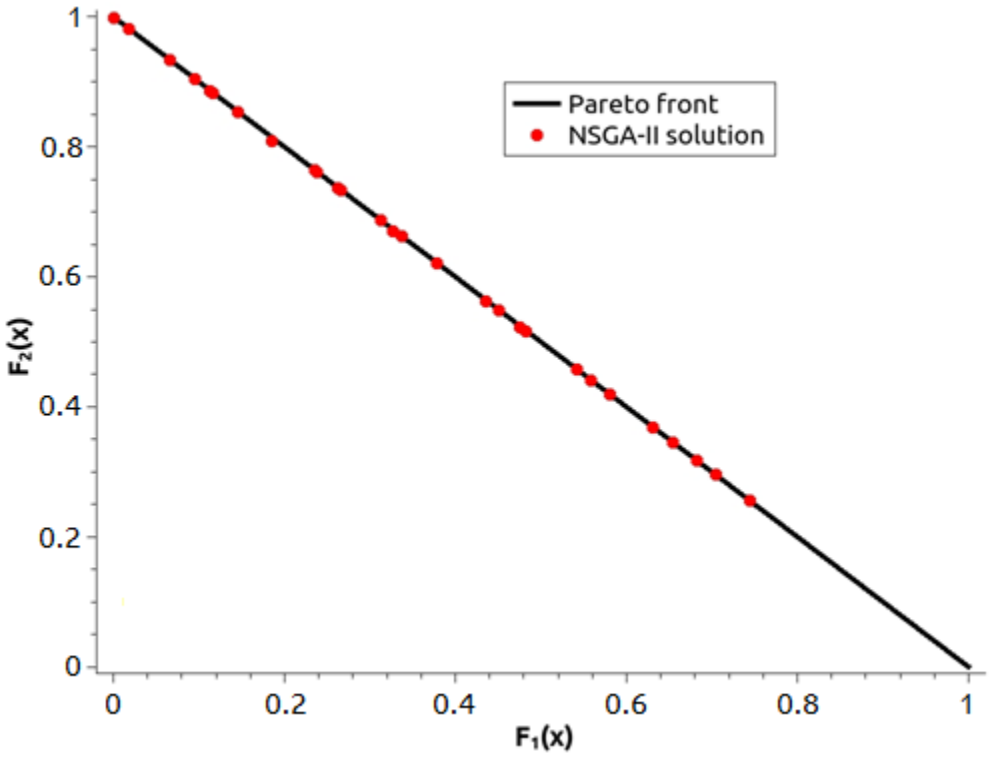

Figure 6.40: The Jimenez problem solved by NSGA-II

The inputs for the hybrid algorithm are the number of initial points for NSGA-II (50), maximum number of generations (100), and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP portion and thus the distribution of the final solutions. The results of the hybrid algorithm are shown in Figure 6-41, and the performance statistics are given in Table 6-8.

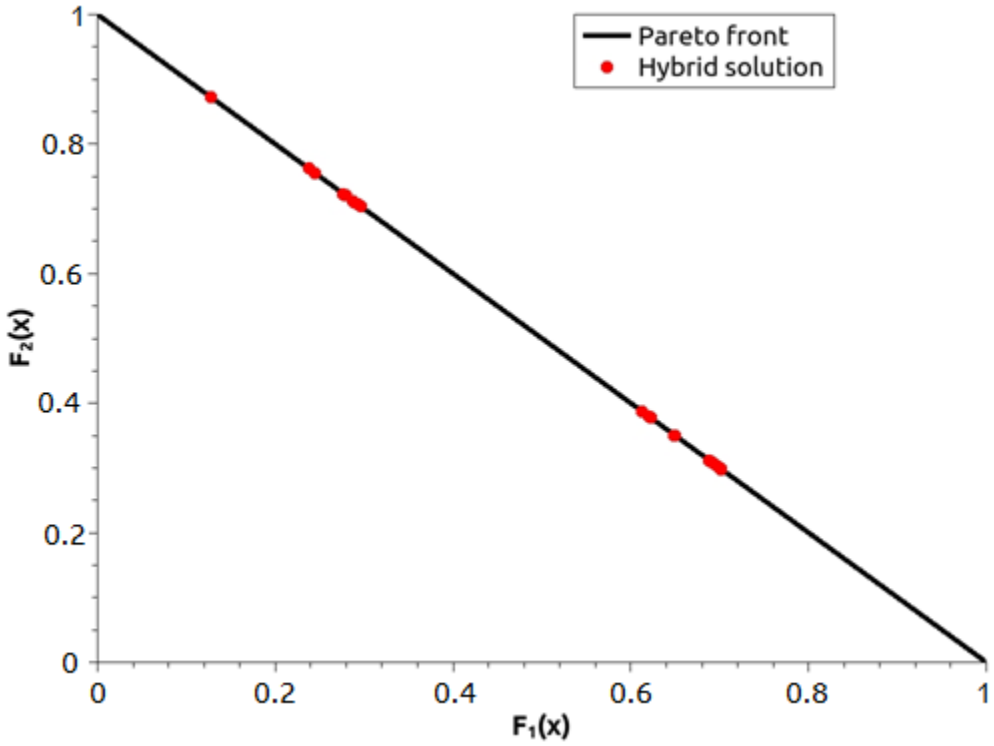

Figure 6.41: The Jimenez problem solved by the hybrid algorithm

The optimization results of the Jimenez problem show that FSQP is unable to obtain a satisfactory set of solutions, since it only provides two points in the extreme points of the Pareto front. This is also shown by the largest PF obtained by this algorithm. The RMSE of the hybrid algorithm doubles that of NSGA-II and the PF of the hybrid solutions is significantly larger than that of NSGA-II. The execution time of the hybrid algorithm is significantly less than that of NSGA-II, but the hybrid algorithm loses more than half of its solution points.

The solutions of the benchmark problems in this chapter showed that the hybrid algorithm is able to solve both convex and non-convex optimization problems. It can also solve problems with constraints and discontinuities in their Pareto fronts. As stated in the

no free lunch theorem (Wolpert et al. 1996), there are some drawbacks from the other algorithms such as a sacrifice on the spread of the solution when compared to the one obtained from NSGA-II. It also shows that the hybrid algorithm is capable of solving multi-objective problems with linear and nonlinear Pareto fronts.

CHAPTER 7: APPLICATION PROBLEMS

This chapter presents a real world application problem that has an unknown Pareto front. This problem will be first solved without any constraints and later will be solved by adding constraints to the formulation.

7.1. Problem Statement and Formulation

The application problem is the design of a square, hollow aluminum tube that is 200 mm long and subjected to a quasi-static crash at 1 mm/s into a rigid wall. During this crash, the energy absorbed by the hollow tube is to be maximized and the mass of the tube is to be minimized. In order for the tube to provide adequate crash protection and sufficient structural stiffness, the maximum crushing force is set to be 200 kN and the mean crushing force (MCF) is set to have a minimum of 70 kN. The design variables are the side length of the outer part of the tube and the wall thickness of the tube. As side constraints, the thickness of the tube is set from 1 to 3 mm, and the side length of the tube is set from 30 to 70 mm. The problem is formulated as follows.

$$\text{Min} \quad (\text{mass}(t, s), -\text{energy}(t, s)) \tag{7-1}$$

$$\text{s.t.} \quad \text{Max force}(t, s) - 200\text{kN} \leq 0 \tag{7-2}$$

$$-\text{MCF}(t, s) + 70\text{kN} \geq 0 \tag{7-3}$$

$$1mm \leq t \leq 3mm \tag{7-4}$$

$$30mm \leq s \leq 70mm \tag{7-5}$$

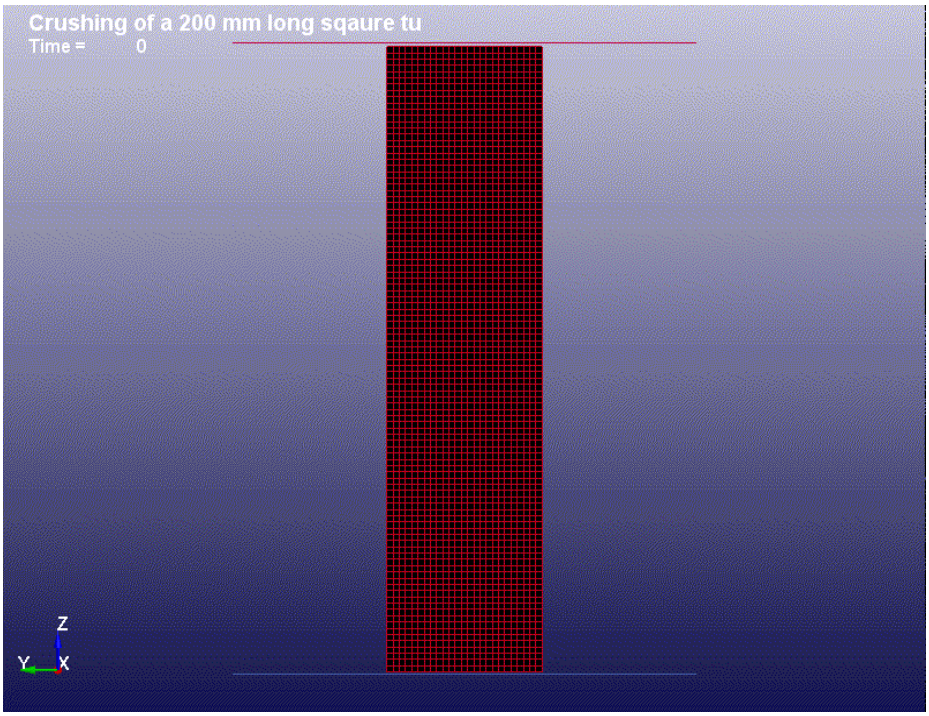

Figure 7.1: Side view of the 200 mm aluminum tube

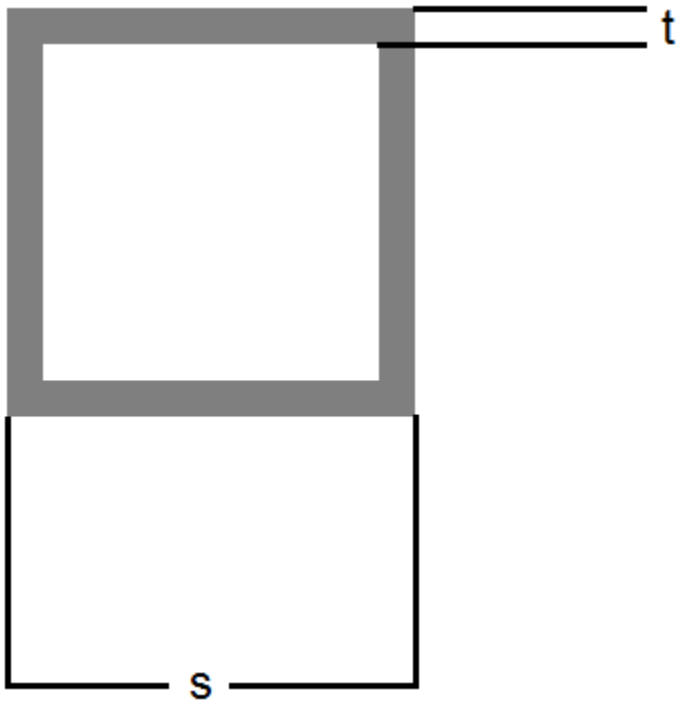

Figure 7.2: Cross section of the 200mm aluminum tube

Since there is no analytical equation that states the relationship between the absorbed energy by a crushing tube and design variables, metamodeling is needed in order to obtain explicit functions that relate the absorbed energy, the maximum crushing force, and the MCF to the design variables.

As stated by Fang et al (2005) and (2006), radial basis functions (RBFs) are a series of functions that are symmetric and concentric at each sampling point. An RBF approximation of a true function can be stated as:

$$f'(x) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) \qquad (7\text{-}6)$$

where n is the number of is sampling points, x is the vector of design variables, x_i is the vector of design variables at the ith sampling point, $\|x - x_i\|$ is the Euclidean distance

and ϕ is the basis function and λ_i is the coefficient for the *i*th basis function. Some radial basis functions are given in Table 7-1.

Table 7-1: Commonly used radial basis functions

Name	Function
Linear	$\phi(r) = r$
Gaussian	$\phi(r) = r^2 \ln(cr),\ 0 < c \leq 1$
Compactly supported (2, 0)	$\phi_{2,0}(t) = (1-t)^5(1+5t+9t^2+5t^3+t^4),\ t = r/r_o$
Compactly supported (2, 1)	$\phi_{2,1}(t) = (1-t)^4(4+16t+12t^2+3t^3)$
Compactly supported (3, 0)	$\phi_{3,0}(t) = (1-t)^7(5+35t+101t^2+147t^3+101t^4+35t^5+5t^5)$
Compactly supported (3, 1)	$\phi_{3,1}(t) = (1-t)^6(6+36t+82t^2+72t^3+30t^4+5t^5)$

Due to its formulation characteristics, the RBF models have an exact match of the predicted values at the sampling points. Therefore, as stated by Fang et al. 2006, there is the need to obtain some off-design points to calculate the difference between the approximation model and the true model.

To compare the RBF approach to other methods of generating the approximation of the objective and constraint functions, response surface methodology (RSM) is used as well. The RSM typically uses a linear or quadratic polynomial to approximate the true response. The approximation function $f'(x)$ can be written in the general form as:

$$f'(x) = \beta_0 + \sum_{i=1}^{m} \beta_i x_i + \sum_{i=1}^{m} \beta_{ii} x_i^2 + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta_{ij} x_i x_j \qquad (7\text{-}7)$$

where $\beta's$ are the unknown coefficients which are solved by the least square method.

In order to obtain the function values at the sampling points, simulations are conducted using LS-Dyna, where the geometry can be changed according to the simulation matrix 25 design points given by Table 7-2.

Table 7-2: Simulation matrix for the application problem

t (mm)	s (mm)	Mass (kg)	Energy Absorbed (J)	MCF (N)	Max Force (N)
1	30	0.000189	1682.052	66893.7	16536.91
1	40	0.000252	1585.880	54371.7	16114.74
1	50	0.000316	1621.139	66893.7	16536.91
1	60	0.000379	1960.639	80979.4	20329.53
1	70	0.000442	1871.044	94989.2	20272.34
1.5	30	0.000284	3532.179	65861.4	35785.56
1.5	40	0.000379	3671.653	85829.3	37327.19
1.5	50	0.000473	4007.591	105310	40964.98
1.5	60	0.000568	4338.155	125479	44123.23
1.5	70	0.000663	3832.673	145957	38930.81
2	30	0.000379	6043.962	93797.3	60944.34
2	40	0.000505	6469.260	117815	64651.39
2	50	0.000631	6934.390	145734	64712.46
2	60	0.000757	7471.032	171616	66211.03
2	70	0.000884	7317.908	196723	75106.1
2.5	30	0.000473	8650.141	125263.4	87865.56
2.5	40	0.000631	9573.997	155586	96494.26
2.5	50	0.000789	9507.073	186933	97367.45
2.5	60	0.000947	10791.100	220528	110589
2.5	70	0.001105	10828.870	251487	112039.7
3	30	0.000568	12371.290	161141	123355.4
3	40	0.000757	13506.290	197489	135181.8
3	50	0.000947	14002.310	233296	142203.7
3	60	0.001136	16207.650	272658	149138.7
3	70	0.001326	15868.780	309819	159806.2

In the metamodeling work of this study, the range of the variables are set to [-1, 1] and the function values are set to the range of [0, 1]. The normalized values of design variables and functions are given in Table 7-3.

Table 7-3: Simulation matrix

Thickness	Side	Mass	Energy Absorbed	MCF	Max Force
-1	-1	0	0.006577	0.002938	0.04902
-1	-0.5	0.055556	0	0	0
-1	0	0.111111	0.002411	0.002938	0.04902
-1	0.5	0.166667	0.02563	0.029332	0.104161
-1	1	0.222222	0.019503	0.028934	0.159005
-0.5	-1	0.083333	0.13311	0.136896	0.044979
-0.5	-0.5	0.166667	0.142648	0.147625	0.123147
-0.5	0	0.25	0.165624	0.172942	0.199408
-0.5	0.5	0.333333	0.188231	0.194921	0.278364
-0.5	1	0.416667	0.153661	0.158785	0.358529
0	-1	0.166667	0.304893	0.311985	0.154339
0	-0.5	0.277778	0.33398	0.337784	0.248362
0	0	0.388889	0.365791	0.338209	0.357656
0	0.5	0.5	0.402492	0.348638	0.458976
0	1	0.611111	0.39202	0.410542	0.557263
0.5	-1	0.25	0.483133	0.499339	0.27752
0.5	-0.5	0.388889	0.546317	0.55939	0.396224
0.5	0	0.527778	0.54174	0.565466	0.518938
0.5	0.5	0.666667	0.629556	0.65748	0.650452
0.5	1	0.805556	0.632139	0.667576	0.771648
1	-1	0.333333	0.737627	0.746326	0.41797
1	-0.5	0.5	0.815251	0.82863	0.560262
1	0	0.666667	0.849174	0.877498	0.700435
1	0.5	0.833333	1	0.925761	0.854526
1	1	1	0.976824	1	1

To select the adequate formulation for the metamodels, sixteen off-design points are selected at the mid points of the ones given in Table 7-3. The function values of these off-design points are also obtained by simulations and are given in Table 7-4.

Table 7-4: Off-design points matrix

Thickness	Side	Mass	Energy Absorbed	MCF	Max Force
-0.75	-0.75	0.076389	0.081901	0.084108	0.031609
-0.75	-0.25	0.145833	0.090057	0.092042	0.094774
-0.75	0.25	0.215278	0.081091	0.084523	0.158615
-0.75	0.75	0.284722	0.091575	0.096161	0.224897
-0.25	-0.75	0.173611	0.235668	0.24392	0.139163
-0.25	-0.25	0.270833	0.254721	0.262151	0.233595
-0.25	0.25	0.368056	0.281553	0.293989	0.322753
-0.25	0.75	0.465278	0.268052	0.278757	0.41184
0.25	-0.75	0.270833	0.405545	0.417487	0.268377
0.25	-0.25	0.395833	0.475043	0.492097	0.377641
0.25	0.25	0.520833	0.385671	0.397206	0.497814
0.25	0.75	0.645833	0.523107	0.541738	0.608694
0.75	-0.75	0.368056	0.62872	0.641475	0.409749
0.75	-0.25	0.520833	0.693936	0.720539	0.542332
0.75	0.25	0.673611	0.727528	0.751652	0.685379
0.75	0.75	0.826389	0.809144	0.838767	0.81797

To select the adequate formulation for the functions to be used in optimization, six RBFs and a polynomial function are used and compares. The R^2 statistics and the RSME values are used to select the most adequate function, as seen in Tables 7-5a and 7-5b.

Table 7-5a: R^2 statistic for the crashing tube problem

	Gauss	Poly	CS(2,0)	CS(2,1)	CS(3,0)	CS(3,1)
Mass	1	1	0.667543	0.878221	0.999998	0.999993
Absorbed Energy	0.98227	0.986719	0.540395	0.984769	0.985528	0.986488
Max Force	0.998835	0.998752	0.644272	0.905028	0.999862	0.999805
MCF	0.949235	0.986186	0.556634	0.987641	0.987354	0.98795

Table 7-5b: RMSE statistic for the crashing tube problem

	Gauss	Poly	CS(2,0)	CS(2,1)	CS(3,0)	CS(3,1)
Mass	1.65E-4	6.22E-10	0.542477	0.291961	0.001015	0.002143
Absorbed Energy	0.127406	0.113846	0.738313	0.120956	0.117488	0.113204
Max Force	0.038555	0.036879	0.597573	0.274382	0.011536	0.013139
MCF	0.227895	0.120502	0.727672	0.110344	0.112076	0.110014

For the mass function, the highest values of R^2 are given by the Gaussian and the polynomial functions. Upon looking at the RSME values, the polynomial function is shown to be a better choice than the Gaussian function to approximate the mass.

The absorbed energy is not well represented by the compactly supported function (2, 0) according to the RMSE value. Looking at the R^2 statistic, the lowest one is obtained from the compactly supported function (3, 1), and thus this function is selected to model the absorbed energy.

For the maximum force, the compactly supported functions (2, 0) and (2, 1) do not represent the model well according to the RMSE values in Table 7-5b. By checking the R^2 values in Table 7-5a, the compactly supported function (3, 0) is selected.

Finally, the mean crushing force is best represented by the compactly supported function (3, 1) and thus selected, because it has the highest R^2 statistic and the smallest RMSE value.

7.2. Unconstrained Optimization of a Crushing Tube

Now that the approximation functions for the objectives and constraints have been defined, the problem can be solved. As a first step, this problem will be solved without any constraints, which will later on be added and solved in section 7.3.

Before solving this problem properly, the maximization of absorbed energy needs to be converted to a minimization problem. Therefore, this function is multiplied by -1 and then added by one (1) to keep the range of function values between zero (0) and one (1).

7.2.1 FSQP Solution

To solve this problem with FSQP, the inputs to the algorithm are the number of initial points (100), minimum weight for WSF (0.01), and weight increment (0.01). This way 100 solution points are expected. The results of this problem are shown in Figure 7.3 and the performance statistics are given in Table 7-6. It is important to note that the PF is calculated after the objective functions are normalized.

Table 7-6: Unconstrained tube problem results

	FSQP	NSGA-II	Hybrid
PF	0.803	0.22	0.15
Execution time	11.74s	6.6s	1.08s
Number of solutions	51	100	96

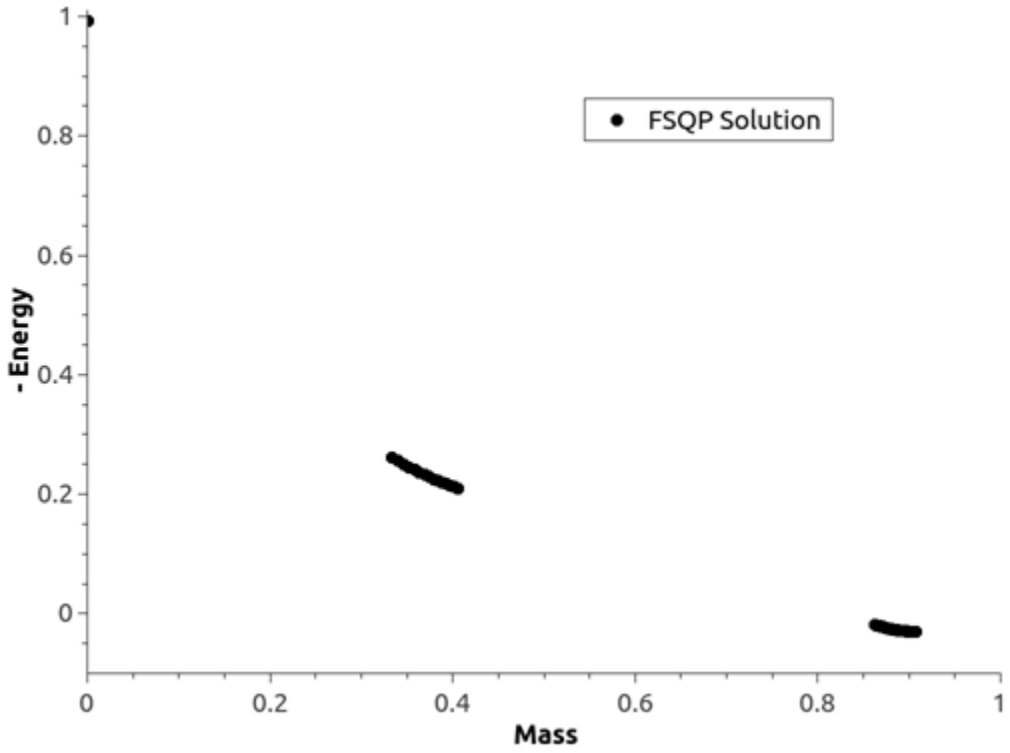

Figure 7.3: The unconstrained optimization problem solved by FSQP

As seen in Figure 7.3, FSQP identifies certain clusters on the solution space. At this point there is no information of the objective space regarding its convexity; therefore a judgment on parts of the Pareto front without solutions cannot be given.

7.2.2 NSGA-II Solution

Next, the unconstrained optimization problem is solved using NSGA-II using the following inputs: the number of initial points (100), number of generations (1000), crossover rate (0.9) and mutation rate (0.01). The solutions for this problem are shown in Figure 7.4 and the performance statistics given in Table 7-6.

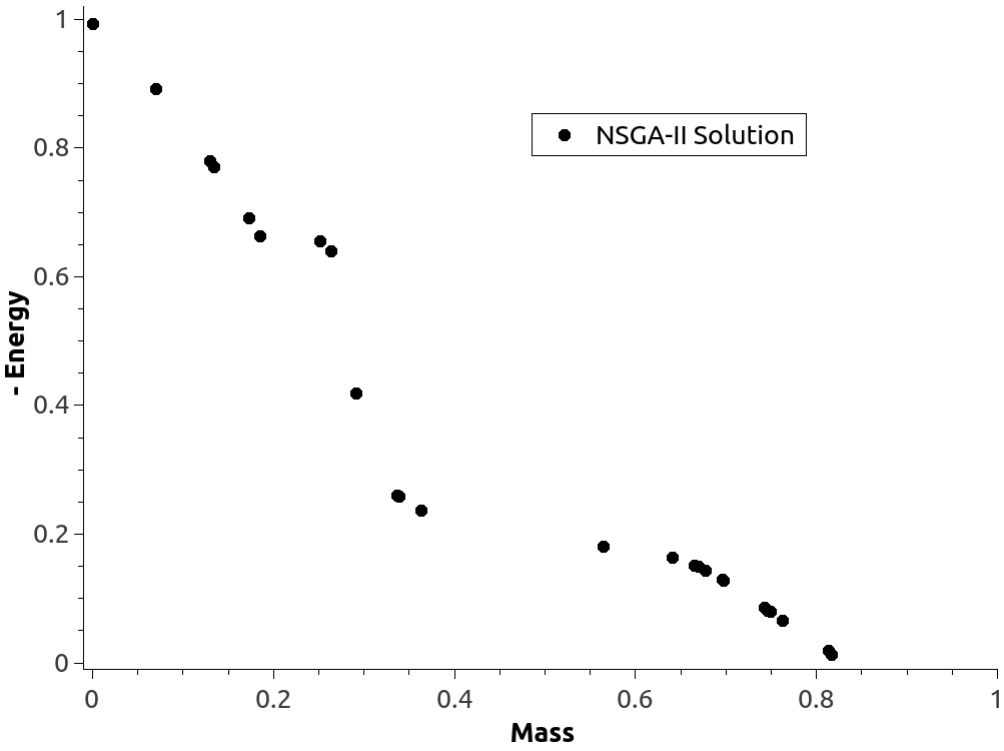

Figure 7.4: The unconstrained optimization problem solved by NSGA-II

It can be seen from Figure 7.4 that NSGA-II identifies a wider range of solutions and spreads more evenly than FSQP solutions. It is also observed that there exists a non-convex region on the Pareto front, which could not be identified by FSQP.

7.2.3 Hybrid Algorithm Solution

The parameters for solving this problem using the hybrid algorithm are the number of initial points for NSGA-II (100), the maximum number of generations (10) and the minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for

the FSQP portion and thus the distribution of the final solutions. The results of this algorithm are shown in Figure 7.5 and summarized in Table 7-6.

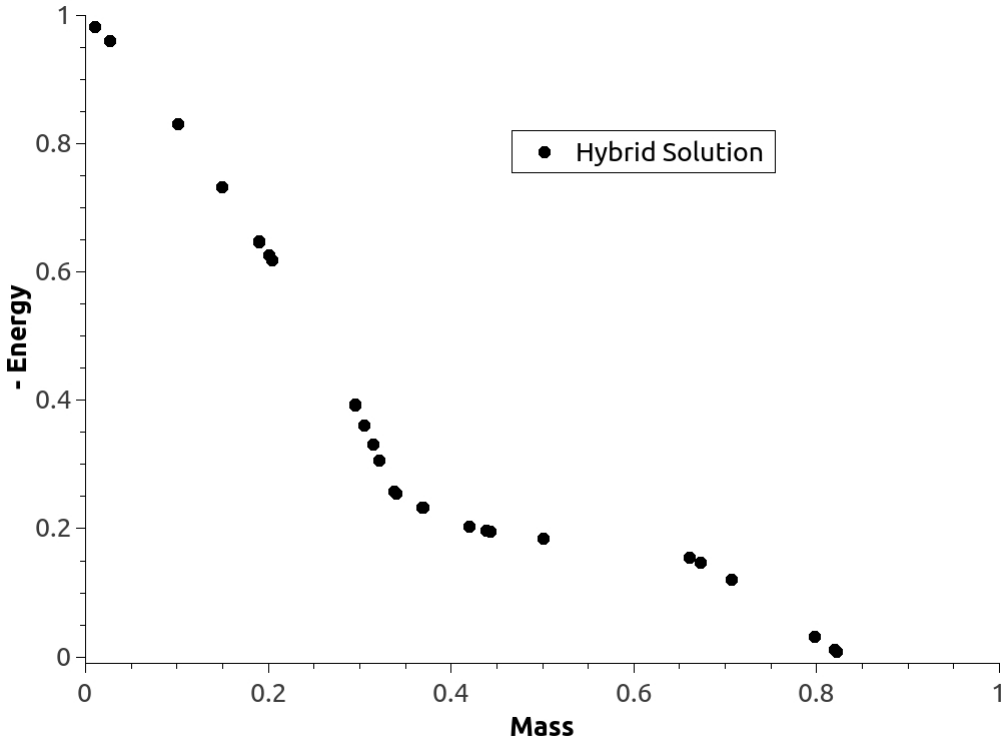

Figure 7.5: The unconstrained optimization problem solved by the hybrid algorithm

To compare the solutions of different methods to this problem, Figure 7.6 shows the sets of solutions obtained by the three algorithms. The combined set of solutions helps to identify solutions dominated by solutions by other algorithms. There are two solution points by NSGA-II that are evidently off the Pareto front, as these are dominated by solutions obtained by the hybrid algorithm.

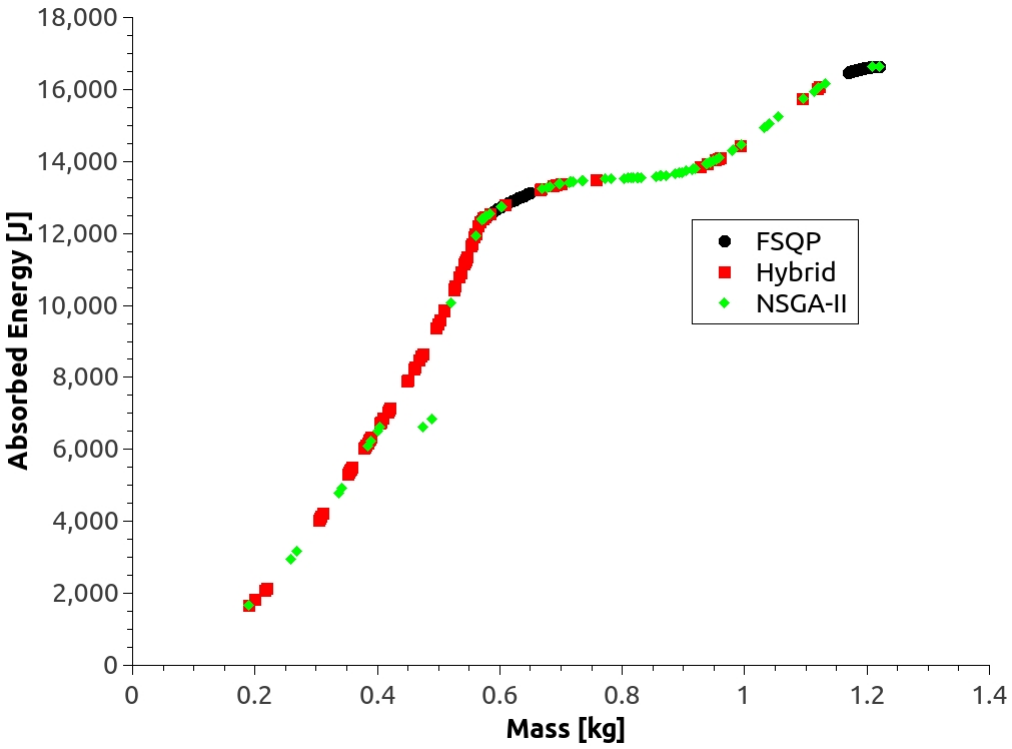

Figure 7.6: Comparison of solutions to the unconstrained optimization problem

As seen in Table 7-6 the largest PF is on the solutions obtained from FSQP, this is because FSQP solutions are in separate clusters, and no solutions are obtained in the non-convex regions of the Pareto front. The solutions of the hybrid algorithm have the lowest PF and thus have the best distribution out of the three algorithms. Regarding the execution time, FSQP takes approximately twice as much as NSGA-II, which is approximately six times more than that of the hybrid algorithm. Finally, the number of solutions obtained by FSQP is 51, an indication of losing nearly half of the initial points through the algorithm's execution. The hybrid algorithm loses four solution points out of the total 100 solution points provided by NSGA-II.

In order to determine the accuracy of the solutions obtained with the approximation functions, two points are chosen from the solutions by the hybrid algorithm and validated by simulation results on these two points. The points chosen in the objective space are (459.8E-3 kg, 8241.47 J) and (607.9E-3 kg, 12793.65 J) where the first value is the mass and the second one is the absorbed energy. The design variables corresponding to the two points are (2.428 mm, 30 mm) and (3.0 mm, 32.1 mm), respectively. The simulation results for the first point are 459.7 kg for mass and 8345.5 J for absorbed energy. The simulation results for the second point are 607.8E-3 kg for mass and 12180.4 J for absorbed energy. The relative errors of the solutions from the approximation functions are calculated as follows:

$$\eta = \left|\frac{x_{simulated} - x_{calculated}}{x_{simulated}}\right| \qquad (7\text{-}8)$$

The relative errors for the first point are 0.00633% for the mass and 1.25% for the absorbed energy. For the second point, the relative errors are 0.0163% for the mass and 5.03% for the absorbed energy. These results show that the optimization results by the approximation models are sufficiently accurate.

In this case, the hybrid algorithm takes less time than its counterparts, while it yields 96 nondominated solutions. In addition, the solutions by the hybrid algorithm have the best distribution among the solutions of the three algorithms. This can be seen from the lowest PF of the solutions by the hybrid algorithm as well as by a visual comparison of the solution distributions.

7.3. Constrained Optimization of a Crushing Tube

In this section, the optimization problem of the crushing tube will be solved with constraints to show the difference in the solutions between the constrained and

unconstrained optimization problems. As stated earlier, the constraints are the maximum crushing force less than 200 kN and the mean crushing force being at least 70 kN.

7.3.1 FSQP Solution

In order to solve this problem with FSQP, the inputs for the algorithm are the number of initial points (100), minimum weight for WSF (0.01), and weight increment (0.01). This results in 100 possible solution points. The results of this problem are shown in Figure 7.7 and the performance statistics are given in Table 7-7.

Table 7-7: Constrained tube problem results

	FSQP	NSGA-II	Hybrid
PF	0.018	0.552	0.078
Execution time	3min 2.26s	19.95s	3.99s
Number of solutions	25	99	97

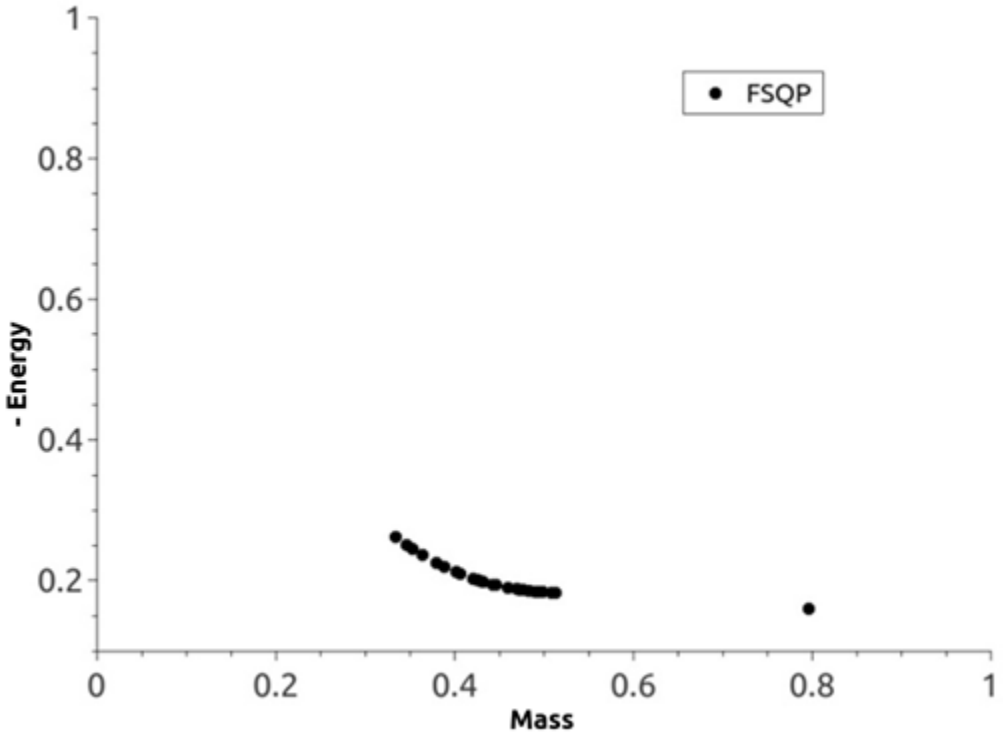

Figure 7.7: The constrained optimization problem solved by FSQP

As seen in Figure 7.7, FSQP identifies points on a small section of the objective space. At this point there is no information regarding the convexity of the objective space and thus a judgment on the part of the Pareto front without solutions cannot be given.

7.3.2 NSGA-II Solution

In order to solve the constrained optimization problem with NSGA-II, the input parameters for the algorithm are the number of initial points (100), number of generations (1000), crossover rate (0.9) and mutation rate (0.01). The results for this problem are shown in Figure 7.8 and the performance statistics are given in Table 7-7.

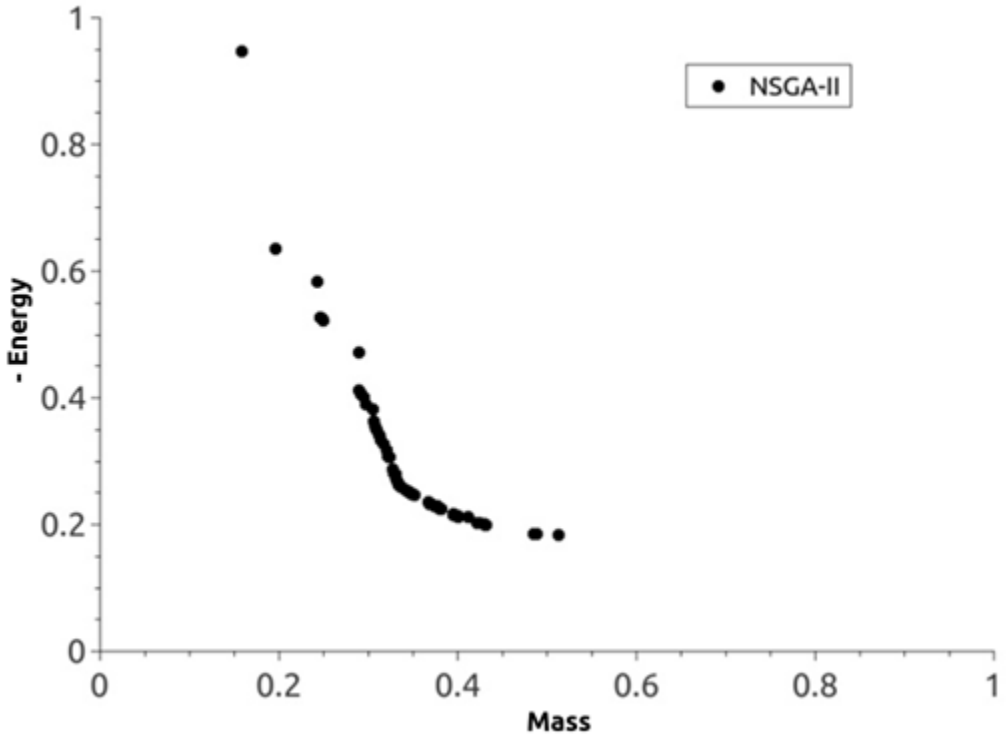

Figure 7.8: The constrained optimization problem solved by NSGA-II

This algorithm identifies more points in the neighboring region than the ones identified by FSQP. In addition, the solutions by NSGA-II also show that there is a concave region on the Pareto front where FSQP fails to yield any solution points.

7.3.3 Hybrid Algorithm Solution

The input parameters for solving this problem with the hybrid algorithm are the number of initial points for NSGA-II (100), the maximum number of generations (10) and minimum crowding distance (0.01). It is observed that, when the maximum number of generations is reached, the NSGA-II portion of the algorithm does not fully satisfy the crowding distance condition, which affects the distribution of initial points for the FSQP

portion and thus the distribution of the final solutions. The results of this algorithm are shown in Figure 7.9 and summarized in Table 7-7.

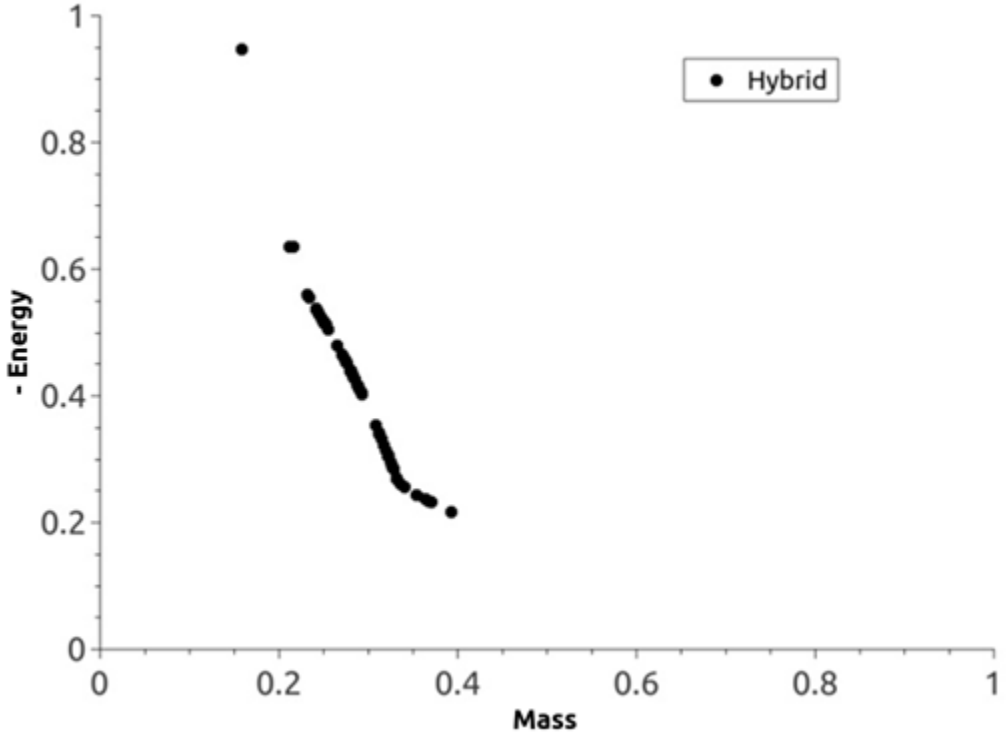

Figure 7.9: The constrained optimization problem solved by the hybrid algorithm

To compare the quality of the solutions by different approaches on this problem, Figure 7.10 shows the combined set of solutions from the three algorithms. It can be seen that FSQP obtains the least amount of points on the solution space. The hybrid algorithm obtains most of its points in the mid section (between 0.25 and 0.35 kg), but NSGA-II is the algorithm whose solutions cover the most of the objective space with its solutions. Finally, two solutions by NSGA-II are dominated by solutions given by the hybrid algorithm and thus should be removed from the final solutions.

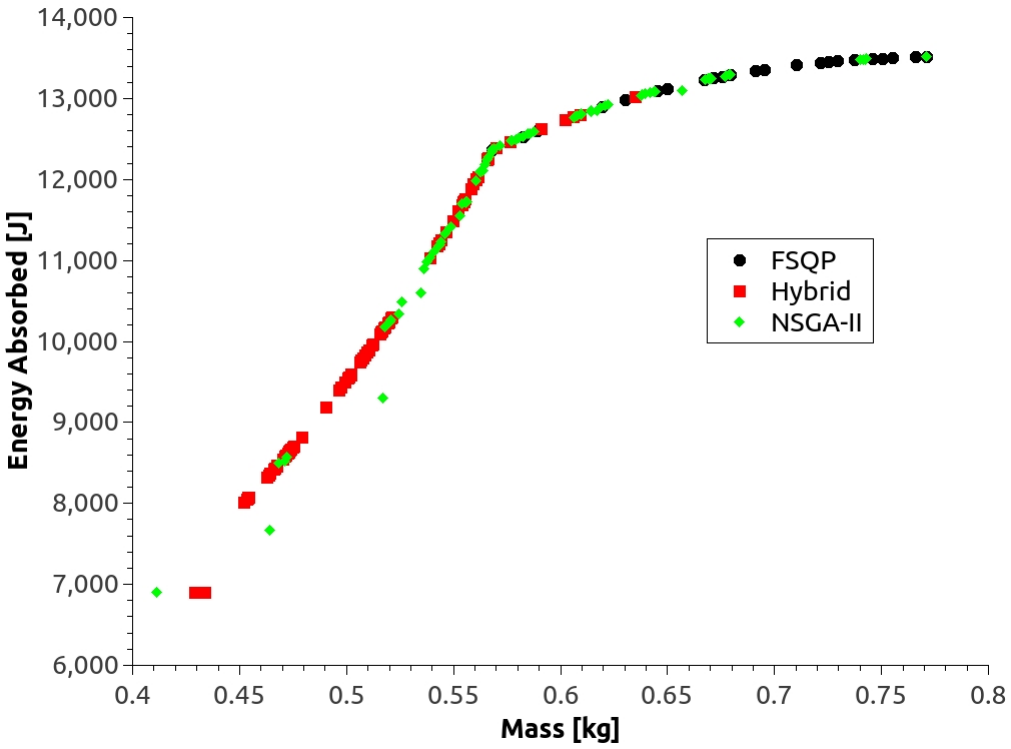

Figure 7.10: Comparison of the constrained tube solutions

In this case, FSQP has the smallest PF followed by the one obtained by the hybrid algorithm and that by FSQP being the highest. Since range of solutions by FSQP is much narrower than the other algorithms, it cannot be said having a good spread of the solution set. The execution time of the hybrid algorithm is the least followed by that of NSGA-II whose time is more than one and half minutes, and that by FSQP that is more than two and half minutes. Regarding the number of solution points, FSQP loses seventy five points out of the total of 100 points. The hybrid algorithm loses three points and NSGA loses one solution point.

To verify that the accuracy of the solutions, two simulations are conducted on two solution points selected from the two main clustering areas on the Pareto front. These two

solution points are (0.542 kg, 11177.08 J) and (0.605 kg, 12774.73 J). The simulation results are given in Table 7-8 along with those predicted by the approximate functions.

Table 7-8: Constrained Tube Problem Results

Mass [kg]	Absorbed Energy [J]	t [mm]	d [mm]	Sim. Mass [kg]	Sim. Energy [J]	Sim. MCF [kN]	Sim. Max Force [kN]
0.542	11177.08	2.864	30	0.542	11574.27	115.805	147.541
0.605	12774.73	3	31.99	0.605	12244.1	122.554	170.176

The relative errors for the mass in both cases are zero. The relative errors for the absorbed energy are 3.4% on the first point and on 4.3% on the second point, respectively. Therefore, both solution points have sufficient accuracy and are feasible since they do not violate any of the constraints.

In this chapter the optimization of an aluminum crushing tube is presented and solved with and without constraints. On the unconstrained problem the hybrid algorithm is the fastest and has the lowest PF among the three algorithms; this makes its distribution the best among these methods. On the constrained problem, the solutions of the hybrid algorithm once again yield the smallest PF, though it failed to identify a feasible solution that is obtained by NSGA-II. Regarding the execution time, the hybrid algorithm is the fastest among the three algorithms for both the constrained and unconstrained problems. The biggest difference on the execution time is on the constrained problem in which the hybrid algorithm takes only 5.62 seconds while NSGA-II takes more than 1.5 minutes and FSQP takes more than 2.5 minutes.

CHAPTER 8: CONCLUSIONS

In this dissertation, the strategies to hybridize a gradient-based method and a genetic algorithm are studied for solving multi-objective optimization problems (MOOPs). Genetic algorithms (GAs) have the characteristics of being capable of solving MOOPs with both convex and non-convex Pareto fronts, but they have the drawbacks of slow convergence and inaccurate solutions when approaching the Pareto front. Gradient-based methods, on the other hand, are computationally stable and robust and are capable of obtaining accurate solutions on the Pareto fronts. However, gradient-based methods are computationally expensive for solving MOOPs using the weighted sum formulation (WSF). Additionally, they are unable to obtain a full set of solutions for MOOPs with non-convex Pareto fronts. The hybrid algorithms investigated in this study aim at improving the solution accuracy and computational efficiency in solving MOOPs by taking advantage of both GAs and gradient-based methods. In this study, the NSGA-II and FSQP are combined using three hybrid strategies and the performance is evaluated.

The first hybrid algorithm is a brute-force approach that demonstrates the potential to successfully hybridize the two algorithms. This algorithm uses a two-stage process: in the first stage, the problem is solved by NSGA-II for a certain number of generations, and in the second stage, solutions from NSGA-II are used as the initial points of FSQP with WSF. While demonstrating the potential advantages of combining

the two methods, the first hybrid algorithm or strategy is computationally more expensive than NSGA-II and is unable to solve MOOPs with non-convex Pareto fronts.

The second hybrid algorithm is developed to improve the computational efficiency by eliminating the expensive WSF used in the first hybrid algorithm. In this second strategy, a direction towards the Pareto front is calculated for each of the initial points before they are used by FSQP. Each of the directions is represented by a certain combination of the weighted objectives based on the point's initial location; therefore, only one solution on the Pareto front is expected instead of attempting to obtain a number of solutions by the WSF. The second hybrid algorithm is shown to have significant improvement in computational efficiency; however, it is still unable to obtain a full set of solutions for MOOPs with non-convex Pareto fronts.

The third hybrid algorithm is proposed to further improve the computational efficiency and solution quality as well as its applicability to MOOPs with non-convex Pareto fronts. In this third strategy, the NSGA-II is first run for only a small number of generations to obtain a set of well spread points. For each initial point, FSQP is used to solve a single-objective minimization subjected to a set of equality constraints, which are converted from all but one objective functions. The third hybrid algorithm is shown to obtain better solutions than NSGA-II and FSQP, or to have better computational efficiency with comparable solutions. In addition, this algorithm is capable of solving MOOPs with non-convex Pareto fronts as well as constrained problems. The third hybrid algorithm, however, is incapable of obtaining a satisfactory set of solutions of MOOPs with discontinuous Pareto fronts, such as the ZDT-3 problem.

Using a real world optimization problem to minimize the weight of an aluminum tube subject to crushing forces, the third hybrid algorithm is compared to NSGA-II and FSQP with and without constraints. For the unconstrained optimization, the hybrid algorithm is shown to produce solutions with better spread on the Pareto front than both NSGA-II and FSQP. In particular, FSQP is unable to obtain solutions because the Pareto front is non-convex, and some of the solutions by NSGA-II are dominated by those obtained by the hybrid algorithm. The computational time of the hybrid algorithm is the least among the three compared methods. For the constrained optimization, the hybrid algorithm and NSGA-II produced comparable solutions, with the former having slightly better spread on the Pareto front. Similar to the case of unconstrained optimization, FSQP is not capable obtaining a full set of solutions on the non-convex Pareto front.

The contribution of this dissertation is the third hybrid algorithm that was proposed in this work. Despite its limitation on MOOPs with discontinuous Pareto front, the hybrid algorithm is shown to be generally efficient and effective, and is applicable to a wide range of MOOPs. For future research on the hybrid strategy, the calculation or determination of the directions for each initial point of FSQP can be investigated to improve the spread of final solutions on the Pareto front. With the capability of obtaining a solution at a given direction, further research can be conducted on constructing the Pareto front using adaptive metamodeling methodology based on a small set of solution points.

REFERENCES

Adeli, H., and Kumar, S. (1995). "Distributed genetic algorithm for structural optimization." Journal of Aerospace Engineering 8(3): 156-163.

Alotto, P., Kuntsevitch, A., Magele, C., Molinari, G., Paul, C., Preis, K., Repetto, M., and Richter, K. (1996). "Multiobjective optimization in magnetostatics: a proposal for benchmark problems." Magnetics, IEEE Transactions on 32(3): 1238-1241.

Anderson, E., and Ferris, M. (1994). "Genetic algorithms for combinatorial optimization: the assemble line balancing problem." ORSA Journal on Computing 6(2): 161-173.

Arora, J. A. (2004). "Introduction to optimum design." USA: El Sevier.

Athan, T. W. and P. Y. Papalambros (1996). "A note on weighted criteria methods for compromise solutions in multi-objective optimization." Engineering Optimization 27(2): 155-176.

Balamurugan, D., Yang, W., and Beratan, D. (2008). "Exploring chemical space with discrete, gradient, and hybrid optimization methods." Journal of Chemical Physics 129(17): 174105-174115.

Balsa-Canto, E., Peifer, M., Banga, J., Timmer, J., and Fleck, C. (2008). "Hybrid optimization method with general switching strategy for parameter estimation." BMC Systems Biology 2008(2): 9.

Bentley, P., and Wakefield, J. (1997). Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. Soft Computing in Engineering Design and Manufacturing. R. R. P. K. Chawdhry, and R. K. Pant. London, Springer Verlag London Limited: 231–240.

Bihn, T., and Korn, U. (1996). "An evolutionary strategy for the multi objective optimization." The second international conference on genetic algorithms 23-28.

Bihn, T., and Korn, U. (1997). "MOBES: a multiobjective evolution strategy for constrained optimization problems." Proceedings of the third international Conference on Genetic Algorithms: 23-28.

Chan, C., Zhang, L., and Ng, J. (2009). "Optimization of Pile Groups Using Hybrid Genetic Algorithms." Journal of Geotechnical & Geoenvironmental Engineering 135(4): 497-505.

Cochran, J., Horng, S., and Fowler, J. (2003). "A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines." Computers & Operations Research 30(7): 1087-1102.

Coello, C., and Christiansen, A. (1999). "MOSES: A Multiobjective Optimization Tool for Engineering Design." Engineering Optimization 31(3): 337–368.

Coello, C. (1999a). "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques." An International Journal Knowledge and Information Systems 1(3): 269-308.

Coello, C. (1999b). "Treating constraints as objectives for single-objective evolutionary optimization." Engineering Optimization 32(3): 275–308.

Coello, C., and Hernandez, A. (2002). "Design of combinational logic circuits through an evolutionary multiobjective optimization approach." Artificial Intelligence for Engineering, Design, Analysis and Manufacture 16(1): 39–53.

Corne, D., Knowles, J., and Oates, M. (2000). "The pareto-envelope based selection algorithm for multiobjective optimisation." Proceedings of the Parallel Problem Solving from Nature VI Conference: 839–848.

Corne, D., Jerram, N., Knowles, J., and Oates, M. (2001). "PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization." Proceedings of the Genetic and Evolutionary Computation Conference: 283-290.

Cui, H., and Turan, O. (2009). "Application of a new multi-agent Hybrid Co-evolution based Particle Swarm Optimisation methodology in ship design." Computer-Aided Design 42(11): 1013-1027.

Das, I., and Dennis, J.E. (1997). "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems." Structural Optimization 14: 63-69.

Deb, K. (1999a). "Multi-objective genetic algorithms: problem difficulties and construction of test problems." Evolutionary Computation 7(3): 205-230.

Deb, K. (1999b). "Solving goal programming problems using multi-objective genetic algorithms." Congress on Evolutionary Computation: 77-84.

Deb, K. (2000). "An efficient constraint handling method for genetic algorithms." Computer Methods in Applied Mechanics and Engineering 186(2/4): 311-338.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." IEEE transactions on evolutionary computation 6(2): 182-197.

Deb, K., and Tiwari, S. (2008). "Omni-Optimizer: A genetic evolutionary algorithm for single and multi-objective optimization." European Journal of Operational Research(185): 1062–1087.

Deb, K., Gupta, S., Daum, D., Branke, J., Mall, A., and Padmanabhan, D. (2009). "Reliability-Based Optimization Using Evolutionary Algorithms." IEEE transactions on evolutionary computation 13(5): 1054-1074.

Dengiz, B., Altiparmak, F., and Belgin, O. (2010). "Design of reliable communication networks: A hybrid ant colony optimization algorithm." IIE Transactions 42(4): 273-287.

Duan, H., Xu, C., and Xing, Z. (2010). "A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems." International Journal of Neural Systems 20(1): 39-50.

Dumasa, L., Druezb, B., and Lecerf, N. (2009). "A fully adaptive hybrid optimization of aircraft engine blades." Journal of Computational and Applied Mathematics 232: 54-60.

Elhossini, A., Areibi, S., and Dony, R. (2010). "Strength Pareto Particle Swarm Optimization and Hybrid EA-PSO forMulti-Objective Optimization." Evolutionary Computation 18(1): 127-156.

Fang, H., Rais-Rohani, M., Liu, Z., and Horstemeyer, M. (2005). "A comparative study of metamodeling methods for multiobjective crashworthiness optimization." Computers & Structures: 16.

Fang, H., and Wang, Q. (2006). "On the effectiveness of assessing model accuracy at design points for radial basis functions." Wiley InterScience: 17.

Fang, H., Wang, Q., Tu, Y., and Horstemeyer, M. (2008). "An Efficient Non-dominated Sorting Method for Evolutionary Algorithms." Evolutionary Computation: 30.

Fonseca, C. and P. Fleming (1993). "Genetic algorithms for multi-objective optimizadon: Formulation, discussion and generalization." Proceedings of the Fifth International Conference on Genetic Algorithms: 416-423.

Fonseca, C. M., and Fleming, P. J. (1995). "An Overview of Evolutionary Algorithms in Multiobjective Optimization." Evolutionary Computation 3: 1-16.

Griva, I., R. A. Polyak, et al. (2010). Primal–Dual Methods for Nonlinear Constrained Optimization. Wiley Encyclopedia of Operations Research and Management Science, John Wiley & Sons, Inc.

Yan, G., Yang, B., Chen, S., and Yan, R. (2006). "Pattern Recognition Using Hybrid Optimization for a Robot Controlled by Human Thoughts." Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems: 5.

Hedayat, A., Davilu, H., Barfrosh, A. and Sepanloo, K. (2009). "Optimization of the core configuration design using a hybrid artificial intelligence algorithm for research reactors." Flexible Conversion Fast Reactors Special Section with Regular Papers Nuclear Engineering and Design 239(12): 2786-2799.

Henz, B., Mohan, R., and Shires, D. (2007). "A hybrid global-local approach for optimization of injection gate locations in liquid composite molding process simulations." Composites Part A 38(8): 1932-1946.

Hiwa, S., Hiroyasu, T.,and Miki, M. (2007). "Hybrid Optimization Using DIRECT, GA, and SQP for Global Exploration." 2007 IEEE Congress on Evolutionary Computation.

Hu, X., Huang, Z., and Wang , Z. (2003). "Hybridization of the Multi-Objective Evolutionary Algorithms and the Gradient-based Algorithms." The 2003 Congress on Evolutionary Computation, 2003. CEC '03 2: 870-877.

Jaszkiewicz, A. (2002a). "Multi-objective Genetic Local Search." European Journal of Operational Research 137(1): 50–71.

Jaszkiewicz, A. (2002b). "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment." IEEE Transactions on Evolutionary Computation 6(4): 402 - 412.

Jaszkiewicz, A. (2003). "Do multiple-objective metaheuristics deliver on their promises? A computational experiment on the set-covering problem." IEEE Transactions on Evolutionary Computation 7(2): 133 - 143.

Jimenez, F., and Verdegay, J. (1998). "Constrained multiobjective optimization by evolutionary algorithms." Proceedings of the international ICSC symposium on engineering of intelligent systems: 266-271.

Jin, Y., M. Olhofer, et al. (2001). "Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how." Proceedings of the Genetic and Evolutionary Computation Conference: 1042-1049.

Kalwij, I., and Peralta, R. (2008). "Non-adaptive and adaptive hybrid approaches for enhancing water quality management." Journal of Hydrology 358(3-4): 182-192.

Kampf, J., and Robinson, D. (2009). "A hybrid CMA-ES and HDE optimisation algorithm with application to solar energy potential." Applied Soft Computing Journal 9(2): 738-745.

Karen, I., Yildiz, A., Kaya, N., Öztürk, N., and Öztürk, F. (2006). "Hybrid approach for genetic algorithm and Taguchi's method based design optimization in the automotive industry." International Journal of Production Research 44(22): 4897-4914.

Kim, D., Abraham, A., and Cho, J. (2007). "A hybrid genetic algorithm and bacterial foraging approach for global optimization." Information Sciences(177): 3918–3937.

Kim, I., and de Weck, O. (2004). "Progressive Structural Topology Optimization By Variable Chromosome Length Genetic Algorithm." The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems: 6.

Knowles, J., and Corne, D. (1999). "The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation." Proceedings of the 1999 Congress on Evolutionary Computation 1: 98-105.

Knowles, J., and Corne, D. (2000a). "Approximating the nondominated front using the pareto archived evolution strategy." Evolutionary Computation 8(2): 149-172.

Knowles, J., and Corne, D. (2000b). "M-PAES: a memetic algorithm for multiobjective optimization." Proceedings of the 2000 Congress on Evolutionary Computation 1: 325 - 332.

Lam, S., Tang, K., and Cai, X. (1996). "Genetic algorithm with pigeon-hole coding scheme for solving sequencing problems." Applied Artificial Intelligence 10(3): 239-256.

Lee, K., Yi, J., Park, J, and Park, G (2003). "An optimization algorithm using orthogonal arrays in discrete design space for structures." Finite Elements in Analysis and Design 40(1): 121-135.

Li, Y., and Li, J. (2010). "Swarm Intelligence Optimization Algorithm Based on Orthogonal Optimization." 2010 Second International Conference on Computer Modeling and Simulation 4: 12.

Lis, J., and Eiben, A. (1997). A multi-sexual genetic algorithm for multiobjective optimization. IEEE International Conference on Evolutionary Computation, 1997.

Manousiouthakis, V. I., N. Thomas, et al. (2011). "On a Finite Branch and Bound Algorithm for the Global Minimization of a Concave Power Law Over a Polytope." Journal of Optimization Theory and Applications.

Marler, R. and J. Arora (2010). "The weighted sum method for multi-objective optimization: new insights." Structural and Multidisciplinary Optimization 41(6): 853-862.

Messac, A., R. V. Tappeta, et al. (2000). "Ability of Objective Functions to Generate Points on Nonconvex Pareto Frontiers." AIAA 38(6): 1084-1091.

Michalewicz, Z. (1994). "Evolutionary computation techniques for nonlinear programming problems." International Transactions in Operational Research 1(2): 223-240.

Michalewicz, Z., and Schoenauer, M. (1996). "Evolutionary algorithms for constrained parameter optimization problems." Evolutionary Computation 4(1): 1-32.

Monson, C., and Seppi, K. (2008). "A Graphical Model for Evolutionary Optimization." Evolutionary Computation 16(3): 283-313.

Murata, T., and Ishibuchi, H. (1995). "MOGA multi-objective genetic algorithms." Proceedings of the 2nd IEEE International Conference on Evolutionary Computation: 289-294.

Murata, T., Ishibuchi, H., and Tanaka, H. (1996). "Multi-objective genetic algorithm and its applications to flowshop scheduling." Computers & Industrial Engineering 30(4): 957-968.

Ngo, N., Zheng, R., Ng, J., Tjin, S. and Binh, L. (2007). "Optimization of Fiber Bragg Gratings Using a Hybrid Optimization Algorithm." Journal of lightwave technology 25(3): 4.

Nicholson, G., and Lancaster, M. (2009). "Coupling matrix synthesis of cross-coupled microwave filters using a hybrid optimisation algorithm." IET Microwaves, Antennas & Propagation 3(6): 950-958.

Obayashi, S., Takahashi, S., and Fejtek, I (1998). "Transonic wing design by inverse optimization using MOGA." Sixth Annual Conference of the Computational Fluid Dynamics Society of Canada.

Padilha, R., Santos, H., Colaço, M., and Cruz, M. (2009). "Single and Multi-Objective Optimization of a Cogeneration System Using Hybrid Algorithms." Heat Transfer Engineering 30(4): 261-271.

Polyak, R., and Teboulle, M. (1995). "Nonlinear rescaling and proximal-like methods in convex optimization." Mathematical Programming 76.

Polyak, R. and M. Teboulle (1997). "Nonlinear rescaling and proximal-like methods in convex optimization." Mathematical Programming 76(2): 265-284.

Sadegheih, A. (2009). "Optimization of network planning by the novel hybrid algorithms of intelligent optimization techniques." Energy 34(10): 1539-1551.

Shi, L., Olafsson, S., and Chen, Q. (1999). "A New Hybrid Optimization Algorithm " Computers & Industrial Engineering 36(2): 18.

Srinivas, N., and Deb, K. (1995). "Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms." Journal of Evolutionary Computation 2(3): 221-248.

Tahk, M., Woo, H., and Park, M. (2007). "A hybrid optimization method of evolutionary and gradient search." Engineering Optimization 39(1): 87-104.

Tahk, M., Park, M., Woo, H., and Kim, H. (2009). "Hessian approximation algorithms for hybrid optimization methods." Engineering Optimization 41(7): 609-633.

Tamaki, H., Kita, H., and Kobayashi, S. (1996). "Multi-objective optimization by genetic algorithms: a review." Proceedings of the 1996 International Conference on Evolutionary Computation: 517–522.

Tan, K., Lee, T., and Khor, E. (2001). "Evolutionary algorithms for multi-objective optimization: performance assessments and comparisons." Proceedings of the 2001 IEEE Congress on Evolutionary Computation Seoul 2: 979-986.

Tan, K., Yang, Y., and Goh, C. (2006). "A distributed Cooperative coevolutionary algorithm for multiobjective optimization." Evolutionary Computation, IEEE Transactions on 10(5): 527-549.

Uler, G., Mohammed, O., and Chang-Seop, K. (1994). "Utilizing genetic algorithms for the optimal design of electromagnetic devices." Magnetics, IEEE Transactions on 30(6): 4296-4298.

Valenzuela-Rendon, M., and Uresti-Charre, E. (1997). "A non-generational genetic algorithm for multiobjective optimization." Proceedings of the Seventh International Conference on Genetic Algorithms: 658-665.

Vaz, A., and Vicente, L. (2006). "A particle swarm pattern search method for bound constrained global optimization." Journal of Global Optimization 39: 197-219.

Vaz, A., and Vicente, L. (2009). "PSwarm: a hybrid solver for linearly constrained global derivative-free optimization." Optimization Methods & Software 24(4/5): 669-685.

Voget, S., and Kolonko, M. (1999). "Multidimensional optimization with genetic algorithms using fuzzy fitness functions." Journal of Heuristics 4(3): 221–244.

Wang, L., Tang, F., and Wu, H. (2005). "Hybrid genetic algorithm based on quantum computing for numerical optimization and parameter estimation." Applied Mathematics and Computation 171(2): 16.

Wang, X., Gao, X., and Ovaska, S. (2008). "A Hybrid Optimization Method for Fuzzy Classification Systems." Eighth International Conference on Hybrid Intelligent Systems: 8.

Watanabe, S., Hiroyasu, T., and Miki, M. (2002). "NCGA : neighborhood cultivation genetic algorithm for multi-objective optimization problems." Proceedings of the Genetic and Evolutionary Computation Conference: 468-465.

Xia, W., Wu, Z., Zhang, W., & Yang, G. (2004). "A New Hybrid Optimization Algorithm for the Job-shop Scheduling Problem." Proceeding of the 2004 American Control Conference.

Chen, X., & Shi, Z. (2009). "A Hybrid Optimization Method and Application in Traffic Signal Timings Optimization." International Conference on Computational Intelligence and Software Engineering, 2009. CiSE 2009.: 11-13.

Yang, D., Li, G., and Cheng, G. (2007). "On the efficiency of chaos optimization algorithms for global optimization." Chaos, Solitons and Fractals 34(4): 1366-1375.

Yıldız, A. (2009). "A novel particle swarm optimization approach for product design and manufacturing." International Journal of Advanced Manufacturing Technology 40(5/6): 617-628.

Zahara, E., and Hu,C. (2008). "Solving constrained optimization problems with hybrid particle swarm optimization." Engineering Optimization 40(11): 1031-1049.

Zhang, G., Shao, X., Li, P., and Gao, L. (2009). "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem." Computers & Industrial Engineering 56(4): 1309-1318.

Zhang, H., and Yang, F. (2008). "Multimodality Medical Image Registration Using Hybrid Optimization Algorithm." 2008 International Conference on BioMedical Engineering and Informatics: 5.

Zhang, P., Yao, X., Jia, L., Sendhoff, B. and Schnier, T. (2007). "Target Shape Design Optimization by Evolving Splines." IEEE Congress on Evolutionary Computation: 2009-2016.

Zhou, G., and Gen, M. (1999). "Genetic algorithm approach on multi-criteria minimum spanning tree problem." European Journal of Operational Research 114(1): 141-152.

Zitzler, E., and Thiele, L. (1999). "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach." IEEE Transactions on Evolutionary Computation 3(4): 257 - 271.

Zitzler, E., Deb, K., and Thiele, L. (2000). "Comparison of multiobjective evolutionary algorithms: empirical results." Evolutionary Computation 8(2): 173-195.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., and da Fonseca, V. (2003). "Performance assessment of multiobjective optimizers: an analysis and review." IEEE transactions on evolutionary computation 7(2): 117-132.

Zitzler, E., and Kunzli, S. (2004). "Indicator-based Selection in Multiobjective Search." Proceedings of the 8th International Conference on Parallel Problem Solving from Nature.