AN ENHANCED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM (MOEA/D-DE) FOR THE APPLICATIONS OF ANALOG SIZING WITH BOTH W/L AND A NOVEL OPERATING POINT DRIVEN (OPD) BASED METHODS

By

Murat Pak

B.S. Electronics Engineering, İstanbul Technical University, 2008

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electronics Engineering
Boğaziçi University

2011

ACKNOWLEDGMENTS

First I would like to give my thanks to Professor Günhan Dündar for his guidance, leading and helpful supervising during my thesis.

I would like to thank to my fiance Hacer Kenar for her existence in my life and her infinite support during my thesis work.

Later I would like to give my appreciation to my family, my father Halil Pak, my mother Hediye Pak and my sister Aylin Pak for their support during the whole life I had so far.

I also feel so lucky to know Mustafa Çelik, my best friend, who supported me for my whole education life.

I am very grateful to all members of ESAT-MICAS, Katholieke Universiteit Leuven and especially to my supervisors Bo Liu and Professor Georges Gielen. Also I will not forget to thank to my project friend Suha Sipahi and all members of BETA, Boğaziçi.

Last but not the least, I would like to express my thanks to TUBITAK for their financial support during my graduate degree.

ABSTRACT

AN ENHANCED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM (MOEA/D-DE) FOR THE APPLICATIONS OF ANALOG SIZING WITH BOTH W/L AND A NOVEL OPERATING POINT DRIVEN (OPD) BASED METHODS

In today's electronics world, due to the growing requirements of mixed signal VLSI designs and the SoCs (system on chip), the design complexity is increasing drastically. Since the well-designed CAD tools can easily support the design of the digital circuits, analog CAD tools are still not enough for the needs of mixed signal VLSI designs and SoCs. One of the biggest reasons for this deficiency is the complex design procedure of an analog system. To design an analog circuit is much harder than designing a digital circuit. However, the world we live in is analog and there is no way to avoid analog circuitry. For all these reasons strong algorithms are trying to be implemented to automate analog circuit design. The sizing problem of analog circuits to obtain the best performance is an important subject of analog design automation.

First of all, the reason why Evolutionary Algorithms are used for the analog sizing problem has been explained. Then, a Multiobjective Evolutionary Algorithm based on the Decomposition of the objective functions has been used as a background work. Lots of improvements have been realized to improve the quality of this method for more complex analog circuit sizing problems. Also, the optimization variables have been changed to DC operating points of the transistors instead of W/L values, in order to improve the search space. All the methods were implemented and the results are given in the work. It can be seen that the proposed W/L or the novel OPD (operating point driven) based methods are so powerful algorithms to optimize the analog sizing problem. During the thesis work, a folded cascode amplifier and a gain boosted amplifier have been optimized.

ÖZET

ANALOG DEVRELERİN TRANSİSTÖR BOYUTLARININ BELİRLENMESİ AMACIYLA KULLANILAN W/L VE OPD YÖNTEMLERİNİ İÇEREN GELİŞTİRİLMİŞ ÇOK OBJEKTİFLİ EVRİMSEL ALGORİTMA

Bugünün elektronik dünyasında, karışık sinyal VLSI (çok geniş ölçekli tümleşik devre) tasarımlarının ve SoC (tek çip üzerinde bir bütün sistem) yapılarının yüksek performans gereksinimleri devre tasarımlarını giderek karmaşık hale getirmektedir. Sayısal devre tasarımları için geliştirilmiş çok sayıda kaliteli CAD (bilgisayar destekli tasarım) aracı olmasına rağmen analog devre tasarımları için bu durum bu kadar iyimser değildir. Bunun en önemli nedeni analog devre tasarımının karmaşıklığıdır. Analog devre tasarımları sayısal devre tasarımlarına göre çok zordur. Ancak, içinde yaşadığımız dünya analog bir dünyadır ve her ne kadar sayısal tasarımlar tercih edilse de analog devre tasarımı sürekli var olacaktır. Bütün bu sebeplerden dolayı analog devre tasarımının otomasyonu için güçlü algoritmalar gerçeklenmeye çalışılmaktadır. Analog tümdevrelerdeki transistörlerin boyutlarının en iyi devre sonuçlarının eldesi için belirlenmesi analog tasarım otomasyonunun önemli bir öğesidir.

Tez çalışmasında önce analog devrelerde transistör boyutlarının optimizasyonunun neden Evrimsel Algoritmalarla yapıldığı ve bu algoritmaların özellikleri açıklandı. Daha sonra Çok Performans Fonksiyonu olan ve farklı problemlerin ayrıştırılmasına dayanan bir Evrimsel Algoritma ön çalışma olarak alındı. Bu algoritmanın kompleks analog devrelerde çalışabilmesini sağlayacak çok sayıda iyileştirme yapıldı. Optimizasyon parametreleri W/L ve transistörlerin DC akım ve gerilim değerleri olacak şekilde algoritma geliştirildi. Tüm sonuçlar tez boyunca raporlandı. Yapılan testler her iki yöntemin de transistör boyutları optimizasyonundaki gücünü göstermektedir. Tez boyunca bir katlanmış kaskod devre yapısı ve bir kazancı arttırılmış kuvvetlendirici yapısının optimizasyonu yapılmıştır.

TABLE OF CONTESTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

B	The neihgborhood space
d	Distance between the objective functions
f	Fitness Function / Objective Function
F	Scaling Factor for DE
g	Optimization problem
L	Transistor Length
m	Dimension of the Objective Functions
N	The Population Size
O	Complexity of the problem
P	Population
T	Number of the individuals in the neighborhood
v_{gs}	Gate to Source Voltage
v_{ds}	Drain to Source Voltage
v_{bs}	Bulk to Source Voltage
W	Transistor Width
x	Optimization variable / solution
Z*	Ideal (True) Pareto Set
δ	The probability for the selection of parent solutions
γ	Greediness of the DE
λ	Weight Vector Matrix
Ω	Solution space for the optimization variables
θ	Penalty Factor
2-D	Two Dimensional
CAD	Computer Aided Design
CR	Crossover Rate
DE	Differential Evolution
DM	Decision Maker

EA	Evolutionary Algorithm
EDA	Electronic Design Automation
EP	External Population
GA	Genetic Algorithm
GBW	Gain Bandwith Product
IGD	Inverted Generational Distance
LUT	Look Up Table
MOEA	Multi Objective Evolutionary Algorithm
MOGA	Multi Objective Genetic Algorithm
MOP	Multi-objective Optimization Problem
NN	Neural Network
NSGA	Non-dominated Sorting Genetic Algorithm
OIOPD	Online Interpolation Operating Point Driven
OPD	Operating Point Driven
PBI	Penalty based Boundary Intersection
PF	Pareto Front
PM	Phase Margin
PS	Pareto Set
SBX	Simulated Binary Crossover
SoC	System on Chip
SPEA	Strength Pareto Evolutionary Algorithm
te	Tchebycheff
VLSI	Very Large Scaled Integrated
VEGA	Vector Evaluated Genetic Algorithm
ws	Weighted Sum

1. INTRODUCTION

1.1. Background Information and Aim of the Thesis

The nature, we live in, is mostly analog and the communication with the nature is necessarily analog. The advent of computers and digital processing methods resulted in processing of discrete signals which are not like the ones of the nature. This suggested that analog circuit design was going to lose its importance; however, especially the need for interface between the real world and digital circuits resulted in analog circuit design to be even more vital. They provide the necessary signal modification and conditioning for digital processing [1].

Because of some economic and other reasons, complete systems that occupy more than one board have started to be integrated on a single chip in recent years, which is known as System on a Chip (SoC). Like application specific integrated circuits (ASIC), such systems also require important amount of analog hardware. A general figure of a SoC is given in Figure 1.1.

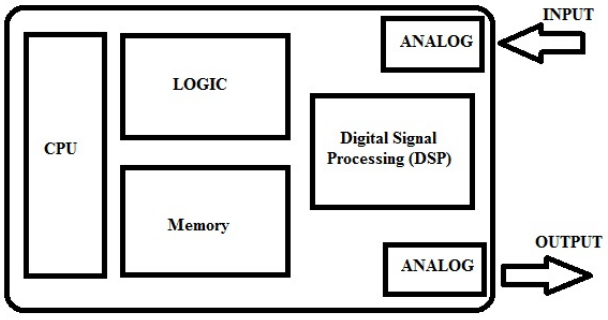

Figure 1.1. The block diagram of a SoC

A trend to replace analog circuit functions with digital computations (like digital signal processing instead of analog filtering) are quite often; however, there are some circuits that will always remain analog for ASIC and for SoC needs. These analog circuits can be classified into three main parts:

i. System's input side: Signals coming from a sensor, antenna, microphone, wireline and so on must firstly be sensed and then amplified and/or filtered up to a level which is useful for digitization with reasonable signal to noise and distortion ratio (SNDR). Low noise amplifiers (LNA), variable gain amplifiers (VGA), filters, oscillators etc. are the analog circuits used for the input side of the system. These blocks are used in applications such as instrumentation, sensor interfaces, process control loops, smart cards, telecommunication receivers, and recording.

ii. System's output side: The signal reconverted from digital to analog has to be strengthened to be able to drive the output load. This load can be an antenna, loudspeaker etc. The analog circuits of the output side of the system are typically the drivers, buffers, filter, oscillators and mixers. Sample applications are telecommunication transmitters, audio and video, process control loops, etc.

iii. Mixed signal (analog and digital signal together) circuits: DSP (digital signal processing) unit and the analog interface parts are integrated to each other through mixed signal circuits. Analog design is an important part of these kind of circuits. Typical circuits used here are the sample and hold (S/H) circuits, analog to digital converters (ADC), digital to analog converters (DAC), phase locked loops (PLL) and frequency synthesizers [2].

In addition, the above circuits, given in three main parts, require stable references for their operations which are generated by voltage and current reference circuits, crystal oscillators, etc [2].

Clearly, analog circuits are necessary in all electronic applications that forms the interface with the outside world, and they will be more important in life as long as the designs go towards intelligent homes, mobile road/air offices and wireless workplaces of the future [2].

Growing requirements, especially for single chip mixed VLSI designs together with the common trends towards smaller feature sizes and higher scales of integration have

brought about new dimensions in the complexity of the circuit design. Still, while the design of digital circuits is throughly supported by well-designed CAD tools, analog CAD tools are still not enough [1].

In digital circuit design, structured abstractions and hierarchy are fully used to generate complex systems with large numbers of devices. For example, digital algorithms can be easily developed with the use of hardware description languages (HDL) for previously determined (in terms of transistor dimensions and gate layouts) logic gates. In contrast, much of the design of analog circuits are still hand-crafted by expert circuit designers. To also speed up the analog design with high quality solutions, efficient methodologies supported by analog CAD programs are needed. CAD tools specifically tailored to analog integrated circuit (IC) design can improve the design process in several ways. These are as given below:

i. Reducing the design times: With the use of the CAD tools the productivity of course increases. Also the time-to-market process gets faster and easier.

ii. Making the design process simpler: With the use of CAD tools, any designer will be able to design standard analog circuits.

iii. Improving the probability of correct designs of the first fabrication run: Automating the correctness capable design tasks reduces the possibility of making errors.

iv. Reducing production and design cost: Shorter design times and smaller design cycle/success ratio (the ratio which defines the successful number of designs in a whole design cycle) are obtained.

v. Improving the yield: Some computer-aided methods based on estimating and enhancing the manufacturing yield of circuit are also popular. The main goal is helping the return backs of the circuit to be faster.

vi. Allowing designs with different fabrication processes: CAD tools can be used to design circuits for different technology files of different fabrications.

vii. Design reuse: The knowledge held by the design systems can be re-used in order to design circuits. This helps designer to spend less time especially on the design of a complex system (since there are lots of same circuit blocks) [1].

Due to all these reasons, analog EDA (electronic design automation) has been a challenging topic for today's electronic systems in the last decades.

Analog EDA industry started with drawing schematics with transistors and layouts. However, there are lots of variables in a design cycle which can be automated by CAD tools. These can be classified as follows:

i. Simulation: The first and most important simulation CAD tools is known as SPICE and it is used for calculating the time domain (transient analysis) and AC behavior (AC analysis) of an electronic circuit. In today's technology, time efficient computer simulations are very important since the number of the transistors are increasing for the circuits to be simulated. However, the original simulation methods are still useful for small size circuits operating at moderate frequencies. The commercialy available simulation techniques are however very diverse with frequency domain approaches, discrete time methods and large scale algorithms etc.

ii. Modeling: Simulation results are formed by using the models describing the behavior of the system. This comes with a need of accurate models. For example, models of larger dimensioned transistors and low frequencies are improved for deep-submicron technologies or RF applications. Beyond the level of individual transistors, various types of models are used for the whole building blocks with different trade-offs between accuracy and complexity and written in different HDL's. There are also some algorithms developed to automatically generate good models.

iii. Layout: Layout is an important part of the analog design cycle since it determines the fabrication properties. There are some tasks used to automate the layout

drawing in analog IC's. The generation of physical structure of individual devices (transistors), their placement on the chip and interconnections between transistors are some of them. Nowaday's popular and come tools are the ones extracting the layout parasitics and checking the correctness of layouts.

iv. Analysis: Analog designers are generally used to do the calculations by hand analysis by using characteristic numbers, tranfer functions etc. However with the improvement in IC design, more complex analysis such as yield estimations and noise effects started to be analysed.

v. Synthesis: The design cycle of an electronic circuit is based on selecting the transistors, sizing them and then using interconnections. Generally, analog designer chooses the circuit schematic and then determines the dimensions by executing the design plan which is basically translation of the analysis results into a set of equations from which the sizes are calculated. Several CAD tools have been implemented by using mathematical optimization methods for solving the problem of transistor sizing. This is the topic of the thesis work. More recently, efforts have been made to enhance the degree of automation of the design process towards the level of the actual synthesis of analog and mixed signal systems, which also includes the selection of the topology. Sizing problem is solved for elementary blocks like op-amps or repetitive structures like filters.

vi. Verification: Since the simulations are using the behavior of the system for a set of inputs, verification is used to show the correctness of the system for all input conditions. Works on this research area has started to be worked on in last 10 years [3].

The CAD tool realized for the thesis work is based on the synthesis process. The goal of the thesis work is finding optimal dimensions for the transistors of an already determined topology in order to get the best circuit performance in terms of gain, gain bandwidth product, phase margin etc.

For the synthesis of the analog circuit, the topology should be chosen. There are several methods that can be used to obtain the architecture in the design cycle. Analog synthesis tools can be classified into four categories according to their architecture selection mechanisms:

i. Selection before or after sizing: Finding optimal values for the optimization parameters is not dependent on the selection of the architecture. To choose a topology, the designer uses a knowledge-assistant tool or his/her experience.

ii. Selection during sizing: The topology is selected during the execution of the sizing algorithm. Options for the topology are stored in a library as entire architectures or as different choices for subblocks of a generalized architecture.

iii. Top-down creation: The functionality of the system is described at a higher abstraction level (usually with a kind of hardware description language). Following steps are used to map this description onto a specific topology. Sizing happens either during or after the mapping operation in a constraint transformation step.

iv. Bottom-up generation: The architecture can also be created from a low abstraction level. The design usually starts at the circuit level by connecting individual transistors with each other in a knowledge-based, systematic or stochastic way. It is possible to obtain several new circuit topologies with this class of design approaches [4].

In Figure 1.2. an overview of the analog synthesis tools developed in the last 20 years has been given. The tools are classified according to their topology selection mechanisms which are selection before or after sizing, selection during sizing, top-down creation and bottom-up generation as given above. The analog synthesis tool developed for the thesis work is a selection before sizing based method. The goal here is first selecting a topology and then optimizing the dimension of the transistors by using optimization techniques.

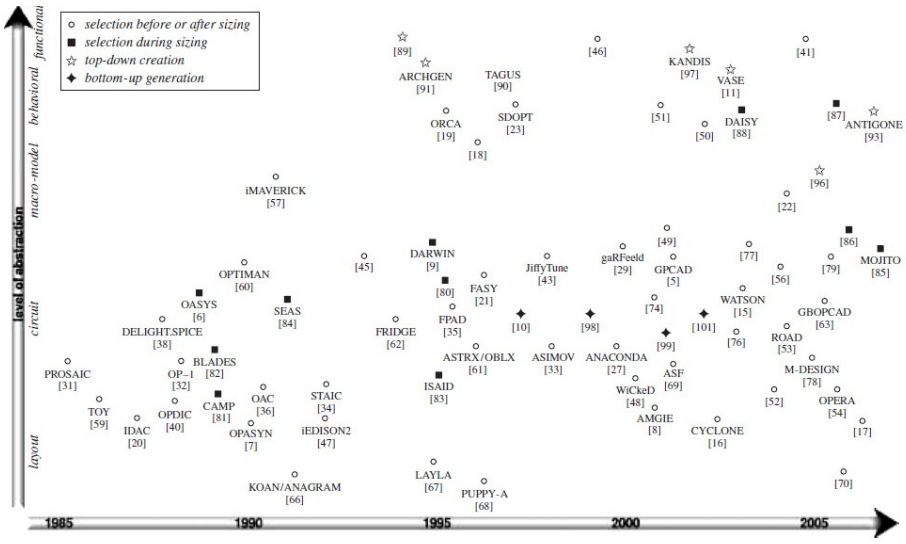

Figure 1.2. Overview of major analog EDA tools for analog synthesis developed in the last 20 years and published in open literature [4]

In most design strategies the topology is selected before and then the sizes are changed to obtain the performance values required. If the specifications are not met at the end, redesign is done. The design flow is shown in Figure 1.3 and Figure 1.4.

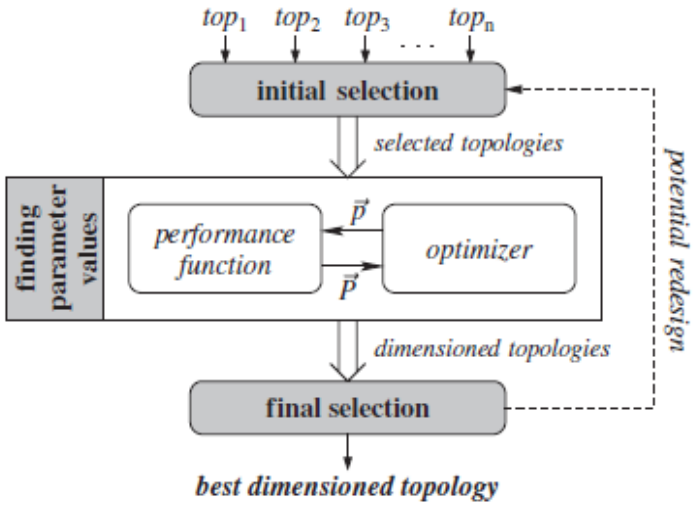

Figure 1.3. Schematic representation of the design strategy applying selection of the topology before or after sizing [4]

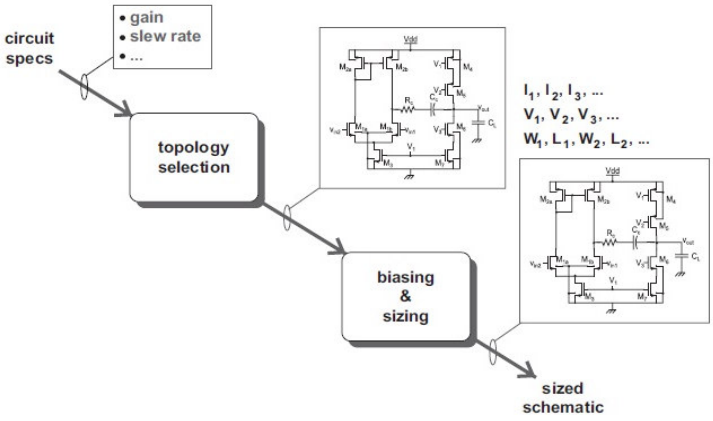

Figure 1.4. Example of a typical design flow for a basic analog cell consisting of topology selection followed by circuit sizing [5]

The calculation of the performance is satisfied by evaluating a performance function (analytical equations, simulation, lookup tables, symbolic analysis or performance models etc.). In the thesis work the performance evaluator is SPICE simulations realized by using HSpice. In case of multi-objective algorithms, multiple combinations of parameter values are returned for a single topology, from which the best one is then selected. Re-design is only necessary when none of the dimensioned topologies meets the specifications. For specific building blocks, like op amps, the circuit level is usually selected. Sometimes this is combined with the physical level, e.g., VCOs or LNAs. On the other hand, the behavioral level is more appropriate for larger systems, like PLLs or complete RF systems [4].

So far what has been mentioned is the history of EDA; CAD tools existing for analog circuit design, the classification based on the topology selection for analog synthesis tools and their design flows. From the design flows given in Figure 1.3, it can be seen that the sizing is mainly realized by the optimizer which is an optimization algorithm. There are several algorithms used for single or multi objective optimization. The details of optimization tools will be given in Chapter 2. For now it should be noted that evolution based algorithms are generally used for the analog sizing problem because of their following properties: They can be used for both single and multi objective optimization, no derivatives of the optimization variables are needed, for any type of problem global convergence can be obtained, there is no need for specific design knowledge, and

convergence is satisfied after enough iterations. For all these reasons, the thesis work is also a multi-objective evolutionary algorithm synthesizer.

Decomposing the objective functions (the functions to be optimized) into several numbers of different scalar problems is known to be improving the convergence quality. This information has been used as the background information of the thesis work and then lots of improvements have been realized. They are basically the generation of weight vectors of the objective functions, using differential evolution as a search algorithm, implementing a novel replacement mechanism for the selection of the individuals, different normalization and decomposition methods, etc. All these concepts will be explained in the following chapters. Also the literature shows that there are important advantages of optimizing the DC operating points of a transistor instead of W values in order to obtain better performance results of the circuit.

As a result, since analog CAD tools are not good enough to satisfy the needs of analog design automation, a thesis work on analog synthesis has been realized. The tool shows good performance in terms of every criteria of design automation (like convergence rate etc.) compared to the other methods realized.

1.2. Outline of the Thesis

The next chapter focuses on different optimization techniques by mentioning their advantages and disadvantages. From this classification why Evolutionary Algorithms are used for analog synthesis tools is also clarified. Since the optimization of multiple performance functions is an important concept for analog synthesis, some works on multi-objective optimization has also been realized. Later, evolutionary algorithms are explained with their components used for optimization. At the end, differential evolution (DE) is explained since it will be used as a search engine for the optimization algorithm realized.

In Chapter 3, the metrics used for comparing the test results of different methods during the thesis work is given. Also, analog circuits used for optimization (as test cases) are introduced with their variables to be optimized.

In Chapter 4, the algorithm MOEA/D which has been used as a background work is introduced. The idea of using decomposition in evolutionary algorithms is explained and the flow chart of the algorithm is given by explaining its advantages and disadvantages.

In Chapter 5, the new algorithm proposed, MOEA/D-DE is given with its flow chart. The test results comparing different enhancing methods are also given. A novel method for the weight matrix initialization of the objective function is explained and the results are compared with the previous technique. Also, different methods of decomposition and normalization are performed. A new replacement mechanism and implementing DE to the Evolutionary Algorithm is also performed and the comparisons of the tests are given. The algorithm developed is used for sizing of an analog circuit.

Chapter 6 includes another novel work based on operating point driven methods. The advantages of OPD methods are given and a novel OPD method called OIOPD is explained. The proposed method is compared with the existing methods and the results are given. Also, some tests on sizing of a complex analog circuit are realized.

The last chapter concludes the thesis work by giving the final conclusions and the possible future works.

2. OPTIMIZATION METHODS AND EVOLUTIONARY ALGORITHMS

This chapter focuses on the different optimization methods and the selection of the evolutionary algorithms as the main optimization search algorithm for the analog sizing problem. The concept of multi-objective optimization is also given with the definition and components of the evolutionary algorithms. The differential evolution which was used in the thesis work is also introduced.

2.1. Optimization Methods

2.1.1. Introduction to Optimization Methods and their Properties

The values of the fitness functions can be changed by altering the variables. Finding the optimal values for these variables is known as optimization. For example, for an analog IC optimizations, W and L values have to be optimized in order to obtain best specifications like gain, power etc. Some methods have been realized to cope with the optimization problem. For analog circuit optimization it should be noted that complex device models can make the feasibility space of an analog system quite difficult. It should be noted that the problem is nonlinear and nonconvex and multiple locally optimal points can be faced. Furthermore, lack of analytical expressions for the performance function results in the need for time-consuming numerical function evaluations through simulations leading to long optimization processes. There are several properties which can characterize the different methods for optimizing the parameters of analog and mixed-signal systems:

i. The objective functions to be optimized can be single or multiple.

ii. For both deterministic function or methods based on stochastic variables, the individuals are optimized in each iteration and they can result in different solutions.

iii. An optimization algorithm may guarantee that the result of the optimization process can reach to the global optimum, whereas others can get stuck in a local optimum.

This is one of the most critical properties for the selection of the optimization algorithm.

iv. The algorithm may have limitations on the models used to represent the system. For example, continuity or convexity are some of these limits.

v. The analog and mixed-signal system optimization can be defined as a mathematical problem which doesn't need introducing any design knowledge about the system. Further, some heuristics can be translated into a set of mathematical sizing rules, e.g., to guarantee the functionality or robustness of subblocks.

vi. The optimization parameters can be discrete values or continous. When the continuous values are used, they may be transformed into discrete values once the final solution has been found (e.g., for transistor dimensions), although this operation can deteriorate the performance.

vii. An algorithm can theoretically guarantee to converge to the global optimum; however, it may be reached only after an infinite number of iterations. This is also an important thing to be considered.

viii. The optimization time needed by an algorithm depends on properties, like the convergence rate, the complexity of the calculations at each step, the different possibilities for choosing a cost function, the population size, optimization algorithm, etc. This is one of the most common properties of all optimization algorithms [4].

2.1.2. Classification of the Optimization Methods

All the above properties are related to lots of different optimization methodologies. Roughly speaking, six base categories are frequently seen in CAD tools for analog synthesis:

i. Design knowledge: This optimization method is used in CAD tools by compiling complete design plans for each topology and objective or by building an expert system which contains general rules. Analog EDA tools often firmly contain design knowledge, like boundaries and typical values for parameters, initial rough sizings based on simplified models, or module generators or templates for layouts. Explicit incorporation of heuristics as mathematical relations is achieved by using the sizing rules method.

ii. Local unconstrained optimizers: Defining a scalar function to minimize is the method used by the transformation of the sizing problem for analog sizing into local unconstraint optimization problem. At low abstraction levels, models and specifications for manufacturing or operating tolerances can be considered. The scalar function may be referred to as the objective, error performance or cost function. To find the optimal value of the cost function, lots of algorithms have been developed. Some examples of the methods implemented in analog CAD tools are simplex methods, gradient-based methods, Newton-like approaches and trust region.

iii. Constrained optimization: Another method is based on considering the analog sizing problem as a constrained optimization problem, also known as linear, quadratic or nonlinear programming, instead of combining all objectives and constraints into one scalar function. Since problems in analog circuit optimization are almost always nonlinear, nonlinear programming is used most commonly. Several algorithms have been successfully applied for analog designs, like gradient-based approaches, and SQP for op amps, LNAs, analog filters etc. A global optimum is guaranteed with a Geometric Program (GP) which has been used for sizing several analog systems, like CMOS opamps, multi-stage amplifiers, on-chip inductors etc. The biggest disadvantage of this method is the restriction on the modeling strategy, especially if a model has to be derived for each topology by hand. Possible solutions consist in fitting the performance function with a posynomial model or solving a series of GPs. Of course, this makes the procedure go on slower and makes the method less attractive.

iv. Greedy stochastic optimization: These algorithms are based on accepting a new set of parameters as long as there is improvement. Elementary stochastic optimization methods like a random grid search and random step are not feasible for finding optimal parameter values for an analog system because of the rather extended design space. Instead, they are usually encountered in combination with other approaches. A straightforward combination is with the design knowledge, like heuristics or information derived during the execution. Also, the gradient used in a deterministic approach can be replaced by an estimation based on random perturbations resulting in a stochastic approximation approach. A globally optimal point is more likely found (not guaranteed) if random pattern searches are combined with a population-based approach.

v. Annealing: This method is based on starting from some point in the parameter space, and then a new set of parameters is derived by selecting statistically a new point in the neighborhood of the old one, or by applying a step of a local optimizer. Comparing to the greedy algorithms, up-hill moves have a certain probability to be accepted in annealing approaches and as a result the method can escape from the risk of the local minima. Consequently, after an infinite number of iterations, the global optimum can theoretically be reached. In a practical implementation; however, it is likely (although not guaranteed) that after enough iterations, the solution is close to the global optimum. The ability to calculate optimal dimensions without the need for derivatives makes the basic simulated annealing algorithm an attractive choice for optimization in analog CAD tools. Some of the applications for analog CAD are the sizing of general analog cells, op amps, VCOs, DS modulators and RF receivers, as well as layout generation.

vi. Evolution: A population of individuals is created in these methods where the parameters of a creature are collected in its genome represented by a binary string (in original genetic algorithms), trees (in genetic programming) or real-valued vectors (in evolution strategies). In these methods each individual has a fitness value defining the cost function which is used for ranking and selection. During the optimization process, new generations are built up by genetic operators. These operators are selection, mutation, and recombination or crossover operators.

Usually, some adaptations are made to the original algorithms to guarantee convergence to the global optimum (e.g., by employing an elitist selection mechanism). However, similar to the annealing algorithms, this global optimum results only after an infinite number of generations [4].

The direct use of the function values, without their derivatives, is the thing that makes these algorithms able to be used in a wide application area. Analog CAD tools based on EAs have been developed to find the optimal parameters of various systems, like RF building blocks, voltage references, ADCs, op amps and power amplifiers. Evolutionary algorithms can also be employed for multi-objective optimization. The fitness then becomes a function of the Pareto dominance of the individuals [4].

The properties of the six base categories given above are listed in the Table 2.1 according to their implementation in CAD tools. However, practical implementations can be based on a combination of these fundamental methods, e.g., simulated annealing followed by a traditional unconstrained algorithm, or a genetic algorithm and simulated annealing method [6].

Table 2.1. The properties of different optimization techniques [4]

	Single-/multi-obj.	Stochastic/ determ.	Population	Derivatives	Global optimum	Models	Knowledge	Memory	Cont./ discrete	Converg.	Speed
Design knowledge	Single-objective	Deterministic	Single point	May be included	No guarantee on optimum	Usually analytical models	Heuristics, rules and design plans	In expert systems	Usually continuous	If no iterations are used	Fast design plans, but slow if interactive
Local unconstrained	Single- and multi-objective	Deterministic	Single point	May be included	Only local optimum	Continuous-time models	Only to select initial point	No memory	Usually continuous	For well-behaving objective functions	Fast for small-sized problems
Constrained	Single- and multi-objective	Deterministic	Single point	For nonlinear programs	Global for convex programs	Continuous or convex models	For setting up models	No memory	Sometimes discrete mixed with continuous values	Usually convergence	Fast for small-sized problems
Greedy stochastic	Usually single-objective	Stochastic	Sometimes with multiple points	Usually no derivatives needed	No guarantee on optimum	No restrictions	To guide random search	To guide random search	Both continuous and discrete	No guarantee on convergence	Slow for large design spaces
Annealing	Usually single-objective	Stochastic	Basic method is single-point	Usually no derivatives needed	Close to global after enough iterations	No restrictions	To select initial point	Only temperature in memory	Both continuous and discrete	Probably after enough iterations	Slow for large design spaces
Evolution	Single- and multi-objective	Stochastic	Population of individuals	No derivatives needed	Likely close to global after enough generations	No restrictions	No specific design knowledge	Implicit via population	Basic GAs use discrete values	Probably after enough iterations	Very slow for large design spaces

From Table 2.1. and from the above explanations of the optimization methods, it can be seen that Evolutionary Algorithms have clear advantages on globally converging to the result, both single/multi objective optimization and no need for derivatives of the

optimization variables. These and other advantages of Evolutionary Algorithms have made them useful for complex problems like analog sizing.

2.1.3. Classification of the Multi-objective Optimization

Most optimization problems are multi-objective optimization in which all objectives should be taken into account. Since these objectives are usually creating a trade-off between each other, it is impossible to find a single and best solution which supports all of the objectives. Instead, there are a number of "Pareto-optimal" solutions which can be explained by the fact that an improvement in an objective can only be satisfied with the lose in performance in at least one other objective. This is basically known as a trade-off between the objective functions [6].

Decision maker (DM) mechanism determines the importance of objective functions. The preferences may be done either before (a priori), during (progressive), or after (a posteriori) the optimization process [6].

A priori methods are the methods based on doing the preference specification before the solution process. These methods aggregates different objectives to one overall objective function and optimization result is then obtained with one optimal design. Of course, the result is strongly dependent on how the objectives were aggregated. Different methods have been developed to support the decision maker mechanism in a priori methods and on aggregating the objectives [7].

Progressive methods are used for the case of a priori methods can not determine the the preference information because of the complexity of the problem. Thus, the decision maker specifies and adjusts his/her preferences at the same time as he/she is learning more about the problem. However, a high effort is expected from the decision maker during the overall search process. Consequently, these methods are almost never used [7].

In the a posteriori methods [8,9,34], optimization is realized without the decision maker articulating any preferences among the objectives. The result of this optimization is a set of optimal solutions which give a trade-off between the objectives. The decision

maker then has to trade the objectives against each other to select the final design. As a result, the optimization is realized before the decision maker clearly indicates his preferences. Basic a posteriori methods are the population based approach and the search based on Pareto solutions [7].

In the population based approach the population size, which is given as N, is divided into subproblems whose number is given as K. Subproblems are also known as the objective functions. Vector Evaluated Genetic Algorithm (VEGA) is an example for this type of approach. VEGA method is classified as a criterion selection technique in which subpopulations (subproblems) are shuffled together to realize a new population which has a size N. At the end crossover and mutation steps are applied to the resulting population untill a stopping criterion is satisfied. The disadvantage of VEGA can be told as the best chromosome at which the optimum decision for all set of functions is obtained, can be rejected as for some from them it is not optimum [7].

There are a several techniques based on searching the solution space first (for Pareto optimal solutions) and presenting them to the DM later. As seen, the advantage is that the solutions are independent of the preferences of the DM. The analysis has to be performed only once, since the Pareto set would not change as long as the problem remains unchanged. However, this method can have a problem about high computational effort. Another disadvantage is that the decision maker may have lots of solutions to choose from [7].

The MOEA/D-DE algorithm that has been implemented during the project is a population based method, which is working with the subproblem logic.

2.2. Evolutionary Algorithms

2.2.1 Introduction to Evolutionary Algorithms

The evolution of nature, Darwinian theroy, has been a start for the evolutionary algorithms (EAs). The change of species adaptively by the principle of natural selection, which helps those individuals for survival and further evolution that are most suitable to

their environmental conditions are all included in these algorithms. The most of the applications of evolutionary algorithms are in optimization. To compare with traditional optimization techniques, evolutionary algorithms are more robust and can obtain a better balance between efficiency and efficacy for many different real-world problems. Evolutionary algorithms are usually used to solve the problems which are characterized by chance, chaos and nonlinear interactivity which tend to be intractable to traditional methods [10].

The basic definition that can be done for the EA techniques is the population full of individuals are getting into a natural selection because of the hard enviromental conditions and this results in a rise for the fitness of the population. Lets assume that a quality function given has to be maximised, first, a set of candidate solutions can be randomly created, i.e., elements of the function's domain, and apply the quality function as an abstract fitness measure which is known as the higher the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (parents) and creates one or more new candidates (the children). Mutation is applied to one candidate and results in one new candidate. Executing recombination and mutation leads to a set of new candidates (which is known as the off-spring) that compete – based on their fitness (and possibly age) – with the old ones for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached [11].

There are two components that form the basis of evolutionary systems.

- Variation operators (recombination and mutation): They create the necessary diversity and thereby ease the novelty
- Selection: It is a force to increase the quality

The combination of recombination, mutation and selection generally leads to improving fitness values in consecutive populations. It is easy to see such a process as if

the evolution is optimising, or at least “approximising”, by approaching optimal values closer and closer over its course [11].

It should be noted that many components of an evolutionary process are stochastic. During selection, the fitter individuals have a higher chance to be selected than less fit ones, but typically the weak individuals still have a chance to become a parent or to survive. For recombination of individuals the choice of which pieces will be recombined is random. Similarly for mutation, the pieces that will be mutated within a candidate solution, and the new pieces replacing them, are chosen randomly. The general scheme of an Evolutionary Algorithm as a pseudo-code is as follows:

```
BEGIN
        INITIALISE population with random candidate solutions;
        EVALUATE each candidate;
        REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
                1 SELECT parents;
                2 RECOMBINE pairs of parents;
                3 MUTATE the resulting offspring;
                4 EVALUATE new candidates;
                5 SELECT individuals for the next generation;
        OD
END
```

Figure 2.1. The general scheme of an Evolutionary Algorithm in pseudo-code [11]

Evolutionary Algorithms (EAs) have a features which makes them available for the generation and testing of the optimization problems. They are as follows:

• They are population based and process the whole collection of candidate solutions simultaneously,

• EAs generally use recombination to mix information (for diversity aim) of more candidate solutions into a new one,

• EAs are stochastic [11] .

The structure of a single population evolutionary algorithm is given in Figure 2.2. As seen firstly an initial population is generated randomly and then according to the results of the evaluation of the objective function the optimization algorithm alters the design variables by using genetic operators.

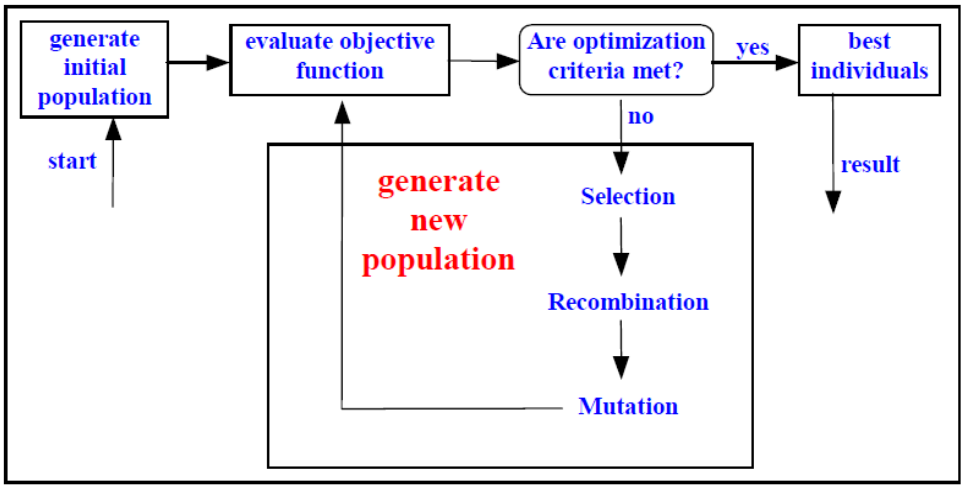

Figure 2.2. Structure of a single population evolutionary algorithm [12]

For more details of EA (evolutionary algorithms), reference [11] can be checked.

2.2.2 Components of Evolutionary Algorithms

EAs have some components and each of these components must be specified in order to define a particular EA. Also, to obtain a running algorithm the initialisation procedure and a termination condition must be also defined. The components are as follows given below [11]:

Representation: It is basically the definition of the individuals. The first step in defining an EA is to link the“real world” to the “EA world”, that is to set up a bridge between the original problem context and the problem solving space where evolution will take place.

Evaluation Function (Fitness Function): The evaluation function represents the requirements to adapt to. It forms the basis for selection, and thereby it facilitates improvements. In other words, it defines what improvement means and it represents the task to solve in the evolutionary context.

The evaluation function is commonly called the fitness function in evolutionary computation (EC). Quite often, the original problem to be solved by an EA is an optimization problem. In this case the name objective function is used in the original problem context and the evaluation (fitness) function can be identical to, or a simple transformation of, the given objective function.

Population: Possible solutions are kept in a population. A population is a set of individuals generating the solution set. Population size is an important represantation in the optimization algorithms and it means the number of the individuals in a population. In some Eas, a population has an additional spatial structure, with a distance measure or a neighbourhood relation. In such cases the additional structure has to be defined as well to fully specify a population.

Parent Selection Mechanism: The goal of parent selection or mating selection is to distinguish among individuals based on their quality, in particular, to let the better individuals to become parents of the next generation. An individual is a parent if it has been selected to experience variation in order to create the off-spring. Together with the survivor selection mechanism, parent selection is responsible for pushing quality improvements. In evolutionary computation, parent selection is typically probabilistic. As a result, high quality individuals get a higher chance to become parents than those with low quality.

Variation Operators: The role of these operators is to create new individuals from old ones. As a result they are the generation operators. Variation operators in evolutionary computation are divided into two types.

Mutation: Mutation, a variation operator, is applied to one genotype and delivers a (slightly) modified mutant, the child or its off-spring. A mutation operator is always stochastic: its output, the child, depends on the outcomes of a series of random choices. It should be noted that an arbitrary unary operator is not necessarily seen as mutation. A problem specific heuristic operator acting on one individual could be termed as mutation for being unary. However, in general mutation is supposed to cause a random, unbiased change. For this reason it might be more appropriate not to call heuristic unary operators mutation.

Recombination: A binary variation operator is called recombination or known as the crossover. As it can be understood from its name, such an operator gets information from two parent genotypes into one or two off-spring genotypes. Similar to mutation, recombination is a stochastic operator: the choice of what parts of each parent are combined, and the way these parts are combined, depend on random drawings. The principal behind recombination is simple: an off-spring can be produced by mating two individuals with different but desirable features. So the off-spring will have both of the features of the mating individuals.

Replacement: This is also known as survivor selection mechanism. The aim of survivor selection or environmental selection is to distinguish among individuals based on their quality. It is similar to parent selection; however it is used in a different stage of the evolutionary cycle. Also, the survivor selection is often called replacement or replacement strategy.

Initialisation: Initialisation in EAs is based on randomly generating the first population. During this work, to make the initialisation fair, compared tests are realized with the same seed number for Matlab's random number generation. In principle, problem specific heuristics can be used in this step aiming at an initial population with higher fitness. Whether this is worth the extra computational effort or not is very much dependent on the application at hand. There are, however, some general observations concerning this issue based on the so-called anytime behaviour of EAs.

Termination Condition: There are two cases to take into account about the termination condition. In case the problem has a well known optimal fitness level, probably coming from a known optimum of the given objective function, then reaching this level (perhaps only with a given precision > 0) should be used as stopping condition. However, there are no guarantees to reach an optimum in Eas because of their stochastic nature, as a result this condition might never get satisfied and the algorithm may never stop. This requires that this condition is extended with one that certainly stops the algorithm. Commonly used options for this purpose are the following:

- The allowed maximum CPU time;
- The limit for the number of evaluations of the fitness functions;
- The limit defining the improvement in the fitness values
- The population diversity comparing to a given threshold [11].

The termination condition for the project is based on the threshold number of the generations. That can also be told to be the number of the iterations for the multi-objective optimization algorithm running.

2.2.3. Multi-objective Evolutionary Algorithms

In recent years, the interest in “evolutionary multi objective optimization” (which the solution optimizes all of the objectives) has increased. The problem of these methods is that, an ideal solution for all objectives may never be obtained in practical applications. Optimal performance (as long as such an optimum exists) according to a single objective, often implies unacceptably low performance in one or more of the other objective dimensions, creating the need for compromise to be obtained. In their applications and nature, evolutionary algorithms explore a set of possible solutions simultaneously. This method let the search for an overall set of Pareto optimal solutions, at least approximately, in a single run of the algorithm unlike the mathematical programming methods. Moreover, these algorithms are less susceptible to problem dependent

characteristics, such as the shape of the Pareto front and the mathematical properties of the search space since they are so important on mathematical programming techniques [10].

The first practical approach for multi objective optimization using evolutionary algorithms was the VEEA, vector evaluated evolutionary algorithm. This method includes the concept of subpopulation and the implementation of recombination and mutation operators on each objective function. None of the other evolutionary algorithms, which try to promote multiple solutions, directly use the actual definition of Pareto optimality. The Pareto-based fitness assignment is assigning equal probability of reproduction to all non-dominated individuals in the population. This method assigns rank 1 to the non-dominated solutions, and then deletes them from contention, then find a new non-dominated set in the rest of the individuals, ranked 2, and so forth until all individuals in the population are assigned ranks. Since the high rank individuals have higher fitness values, they have more chance to reproduce offspring and be chosen into next generation. After VEEA, a multi-objective genetic algorithm (MOGA) has been proposed using a slightly different fitness evaluation scheme, whereby an individual's rank corresponds to the number of individuals in the current population by which it is dominated. Non-dominated individuals are, therefore, all assigned the same highest rank, while dominated ones are penalized according to the population density in the corresponding region where this individual is dominated. Also a similar sorting and fitness assignment procedure has been implemented, which is called NSGA, the non-dominated sorting genetic algorithm, but based on Goldberg's version of Pareto ranking. Also, niched Pareto genetic algorithm (NPGA) was proposed using a tournament selection method based on Pareto dominance as an alternative to the deterministic rank-based selection. More recent algorithms are NSGA-II [8], and the strength Pareto evolutionary algorithm (SPEA2) algorithm [9].

The general procedure of MOEA is like the one in evolutionary algorithms for single-objective optimization [10].

MOEAs realize the search by maintaining a population of individuals at time t which is given as $P(t) = \{p_1(t), \ldots p_N(t)\}$. The general procedure is given in Figure 2.3.

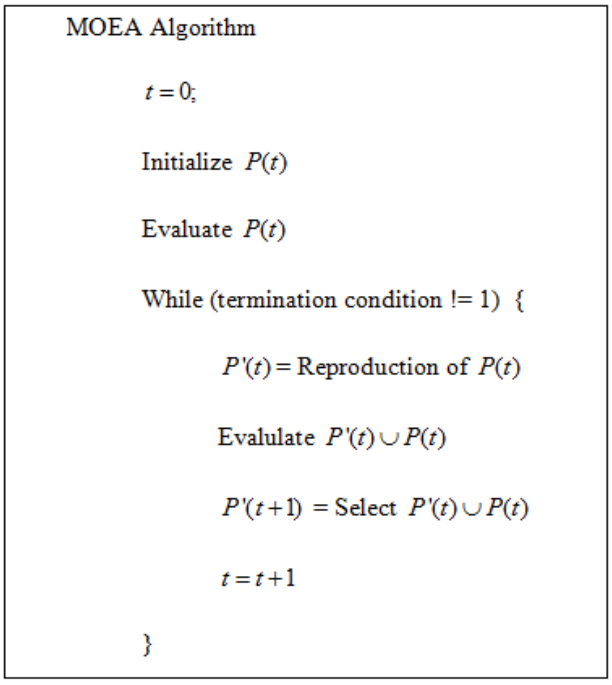

Figure 2.3. The General Procedure of MOEA [10]

The use of probabilistic operators is because of generating a new and better population $p(t+1)$. This step reproduction. Each $p_i(t) \in P(t)$ represents a potential solution to the problem. The initialization and operations are critical for the performance of the algorithm. The data structure to represent a solution is usually tailored with the consideration of specific problems. Data structure which have been specifically designed for real problems mostly reduce the effort in the search process [10].

2.3. Differential Evolution (DE)

Differential Evolution (DE) is an evolutionary algorithm used in optimization problems. There are several variants of the original differential evolution. The one described below has been chosen as an example to explain the basics of the DE. The main operators controlling the evolutionary process are the reproduction and selection operators [10].

The algorithm works with the general idea of an evolutionary algorithm. An initial population is created by random selection and then evaluated; then the algorithm works in

the flow chart of generating offspring, evaluating offspring, and selecting individuals to create the next generation. In DE, for every individual in the parent population, the following reproduction operator is used to create its offspring [10]:

$$p_i' = \gamma . p_{best} + (1-\gamma) . p_i + F . \sum_{k=1}^{K} (p_{i_a^k} - p_{i_b^k}) \qquad (2.1)$$

p_{best} is the best individual in the parent population, p_i' is the offspring that is generated , γ represents greediness of the operator, and K is the number of perturbation vectors, F is the scale factor of the perturbation, and $p_{i_a^k}$ and $p_{i_b^k}$ are randomly selected mutually distinct individual pairs in the parent population. The DE approach is shown schematically in Figure 2.4.

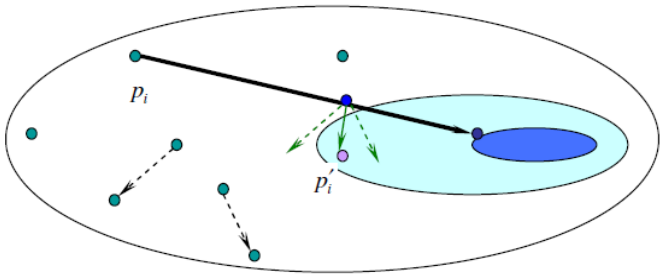

Figure 2.4. Illustrative example of differential evolution for single objective optimization, in a 2-dimensional decision space [10]

The basic idea behind the DE is adapting the search step naturally along the evolutionary process in a manner that trades exploitation off against exploration. The scale of the perturbation vectors is proportional (roughly) to the extent of the population diversity. At the beginning of the evolution, since the individuals are far away from each other, the perturbation is large. As the evolutionary process reaches to the final stage, the population converges to a small region and the perturbation gets smaller. As a result, the adaptive search step benefits the evolution algorithm by performing global search with a large perturbation step at the beginning of the evolutionary process and refines the population with a small search step at the end [10].

The darker area in Figure 2.4 is for the better fitness values. The thick solid arrow is for the differential vector, and dashed arrows represent the perturbation vectors. The individual p_i creates its offspring p_i' after the reproduction operation.

The selection operator in DE compares the fitness values of the parent and offspring and chooses the better one as shown in Equation 2.2.

$$p_i^{t+1} = \begin{Bmatrix} p_i^{'(t)} if \Phi(p_i^{'(t)}) > \Phi(p_i^{(t)}) \\ p_i^{(t)} otherwise \end{Bmatrix} \tag{2.2}$$

For more details about the DE approaches a PhD thesis work on multi-objective differential evolution [10] can be checked.

3. PERFORMANCE METRICS AND TEST CIRCUITS FOR THE ALGORITHMS IMPLEMENTED

To compare the results of different methods for multi-objective optimization, some performance metrics and some analog circuitry to be sized are needed. The goal here is to try different benchmark problems and analog circuits to make the comparison fair and to use some metrics fairly comparing the distribution, range and especially dominance quality of a Pareto Front with another one.

3.1. Performance Metrics

The metrics used to compare Pareto Fronts are about the distribution quality, dominance and the range of the objective functions.

3.1.1 Schott's Spacing Metric

Schott describes the following spacing metric:

$$\sqrt{\frac{1}{n-1}\sum_{i+1} n.\left(\bar{d}-d_i\right)^2} \tag{3.1}$$

where $d_i = \min_j \left(\left|f_1^i(\vec{x}) - f_1^j(\vec{x})\right| + \left|f_2^i(\vec{x}) - f_2^j(\vec{x})\right|\right)$, $i, j = 1...n$, $\bar{d}$ is the mean of all d_i and $n = |Z|$.

Schott's Spacing Metric tries to find how evenly the points are distributed. It is an independent metric, induces a complete ordering, and is cardinal. It exhibits neither monotony nor relativity, since Z^* may be non-uniform. Used in conjunction with other metrics (as it is designed to be) it provides information about the distribution of vectors obtained. It has low computational overhead. It can be generalized to more than two

dimensions by extending the definition of d_i. Schott's definition of d_i does not specify the use of normalized distances, which may be problematic [13].

3.1.2 IGD Metric

The inverted generational distance (IGD) is also used to determine the performance of the algorithms. Let P^* be a set of uniformly distributed points in the objective space along the PF. Let A be an approximation to the PF, the inverted generational distance from P^* to A is defined as:

$$IGD(A, P^*) = \frac{\sum_{v \in P^*} d(V, A)}{\left|P^*\right|} \tag{3.2}$$

where $d(V, A)$ is the minimum Euclidean distance between *v* and the points in A. This method has been used as the main performance metric in the CEC09 conference [23].

3.1.3 Number of Dominated Points

To find the true pareto (PF) from all possible pareto optimal solutions of lots of tests of different methods, first of all, the dominated points from the whole possible PF points are found out and then eliminated. So, the points left are used for the true Pareto which is also called Pareto Front to use with metrics like IGD in order to compare the performance of the methods.

The number of the dominated points for a single test is also a good method especially for the dominance quality. If there are lots of dominated points in the objective space of a method, then this method can be considered to be easily dominated by the other methods.

Since m is the number of the objectives in a solution space with n individuals, for a PF with lots of solution points on it, for $i = 1:m$ if $\forall(f_i > P_i)$ is true for a minimization problem than the related individual of the solution space inquired is dominated. For example, if Method1 has 20 individuals dominated comparing to a PF1, Pareto Front, and if Method2 has just 3 individuals like that, for fair conditions (like same seed number of initialization) it will mean that Method2 is much better in terms of dominance quality.

3.2. Circuits to be Optimized

In this part two analog circuits used for the optimization during the thesis work are presented. For the first part of the thesis work a W/L based method is used and for the tests of this method the W and L values of the transistors are the design variables. For the second part of the thesis an operating point driven (OPD) based method has been implemented. The design variables for this method are the DC voltage and current values of the transistors. For the analog circuits optimized these values have been shown in Table 3.1 and Table 3.2.

For 90nm and 180nm (TSMC) and 250nm (UMC) fabrication models, folded cascode and gain boosted amplifier, have been simulated and the dimensions of the transistors have been optimized during the project.

The variables are dependent on the algorithm used. If the MOEAD-DE algorithm based on the W-L search space is used then (first part of the project) then the variables to be optimized are W and L values. If the DC root solving method (OIOPD) is added, then the optimization variables are basically V (voltage) and I (current) values of the nodes.

For different technology files (90nm, 180nm, 250nm), the W/L values and the V/I values to be optimized are given in Table 3.1. and Table 3.2. respectively. It should be noted that the values are for the upper and lower boundaries for the related technology parameters.

Table 3.1. W-L Limits for the Optimization Variables

Technology:	90nm	180nm	250nm
Wmin	120nm	240nm	360nm
Wmax	200um	800um	800um
Lmin	90nm	180nm	250nm
Lmax	5um	10um	20um

Table 3.2. V-I Limits for the Optimization Variables

Technology:	90nm	180nm	250nm
Vmin	-0,6V	-0,9V	-1.25V
Vmax	0,6V	0,9V	1.25V
Imin	0,1 mA	0,1 mA	0,1 mA
Imax	10mA	10mA	10mA

3.2.1 Folded Cascode Amplifier

The folded cascode circuit seen below has been used as the main analog circuit to try the sizing algorithm. In the circuit there are 13 transistors and 1 current source whose values have to be altered. Because of the design properties (like symmetry), some values are equalized to each other.

During the W-L optimization of the folded cascode amplifier there are 11 values to be optimized. W1, W3, W5, W8, W10, L1, L3, L5, L8, L10 and ib. The ib value has been searched in a range 0,5uA to 2,5mA. The W and L values are searched in the range according to Table 3.1. The choice of values is dependent on the technology restrictions (for the minimum values) and also experimental (for the maximum values).

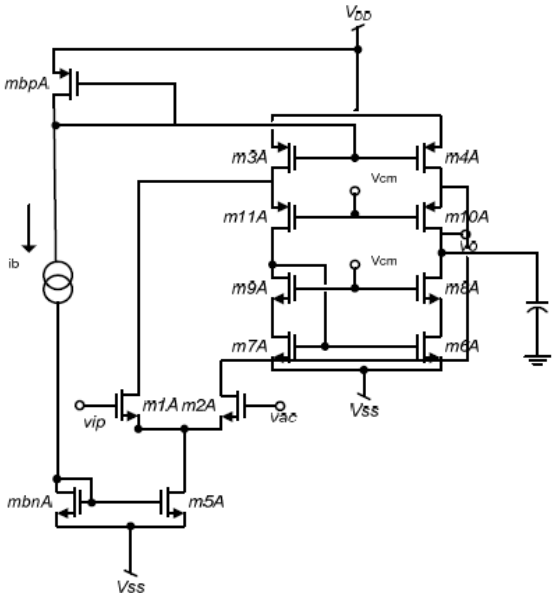

Figure 3.1. The Folded Cascode Amplifier

Table 3.3. W/L components to be optimized

Transistor	Width	Length
m1A	W1	L1
m2A	W1	L1
m3A	W3	L3
m4A	W3	L3
m5A	W5	L5
m6A	W5	L5
m7A	W5	L5
m8A	W8	L8
m9A	W8	L8
m10A	W10	L10
m11A	W10	L10
mbnA	W3/1.1	L3
mbpA	W5	L5

OIOPD method, based on optimizing the DC variables have 7 voltage and 3 current values to be optimized as seen below. Since this method is based on the guessing W process (that will be explained on the related part of the thesis) L values also have to be optimized. So there are 5 different transistor length values also (they were given before). As a result, a total of 15 variables need to be altered to find the best Pareto Optimal solutions. The DC values need to be altered are given in Figure 3.2.

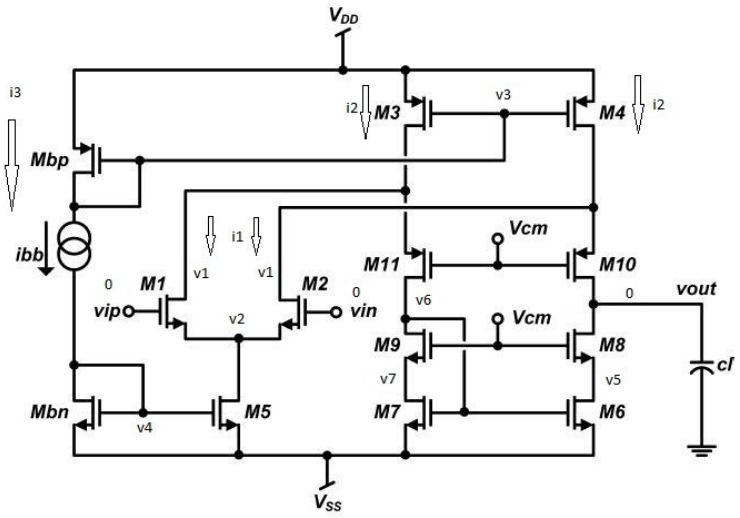

Figure 3.2. V/I variables for the Folded Cascode Amplifier

3.2.2 Gain Boosted Amplifier:

The gain boosted amplifier circuit given in Figure 3.3 has been used as the main analog circuit to try the sizing algorithm. In the circuit there are 39 transistors and 4 capacitors whose values have to be altered. Since the N amplifier and P amplifier wil be used 2 times then the number of the transistors to be optimized decrease to 25 and number of the capacitors decrease to 2. Also because of the design properties (like symmetry), some values are equated to each other.

During the optimization of the gain boosted amplifier, there are 38 values (18 W, 18 L, 2 C) to be optimized.

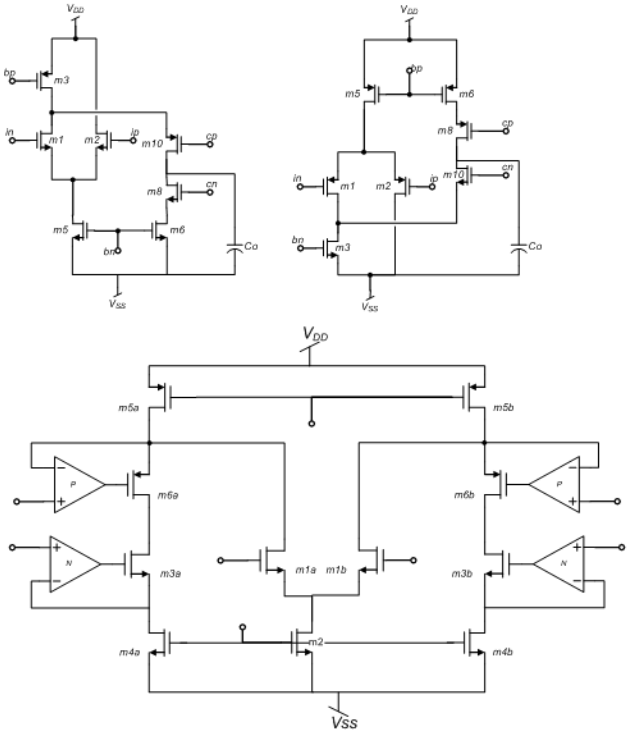

Figure 3.3. The Gain Boosted Amplifier

Table 3.4. The W/L values to be optimized for the Main Block of the Gain Boosted Amplifier

Transistor	Width	Length
Main Block:		
m1a	fw1a	fl1a
m1b	fw1a	fl1b
m2	fw2	fl2
m3a	fw3a	fl3a
m3b	fw3a	fl3a
m4a	fw4a	fl4a
m4b	fw4a	fl4a
m5a	fw5a	fl5a
m5b	fw5a	fl5a
m6a	fw6a	fl6a
m6b	fw6a	fl6a

Table 3.5. The W/L values to be optimized for the P-Amplifier of the Gain Boosted Amplifier

P-Amplifier:	W	L
m1	sw1	sl1
m2	sw1	sl1
m3	sw3	sl3
m5	sw5	sl5
m6	sw6	sl6
m8	sw8	sl8
m10	sw10	sl10

Table 3.6. The W/L values to be optimized for the N-Amplifier of the Gain Boosted Amplifier

N-Amplifier:	W	L
m1	tw1	tl1
m2	tw1	tl1
m3	tw3	tl3
m5	tw5	tl5
m6	tw6	tl6
m8	tw8	tl8
m10	tw10	tl10

The capacitor value has been searched in the 0,5pF to 20pF range. The W and L values have been searched in the range according to Table 3.1.

OIOPD method, based on optimizing the DC variables for the gain boosted amplifier has 16 voltage and 10 current values to be optimized. Again L values are also supposed to be altered. There are 18 different transistor length values also (they were given before). When 2 capacitor values to be altered are added, a total number of optimization

variables is found to be 46. The DC values of the gain boosted amplifier which need to be altered are as given in Figure 3.4.

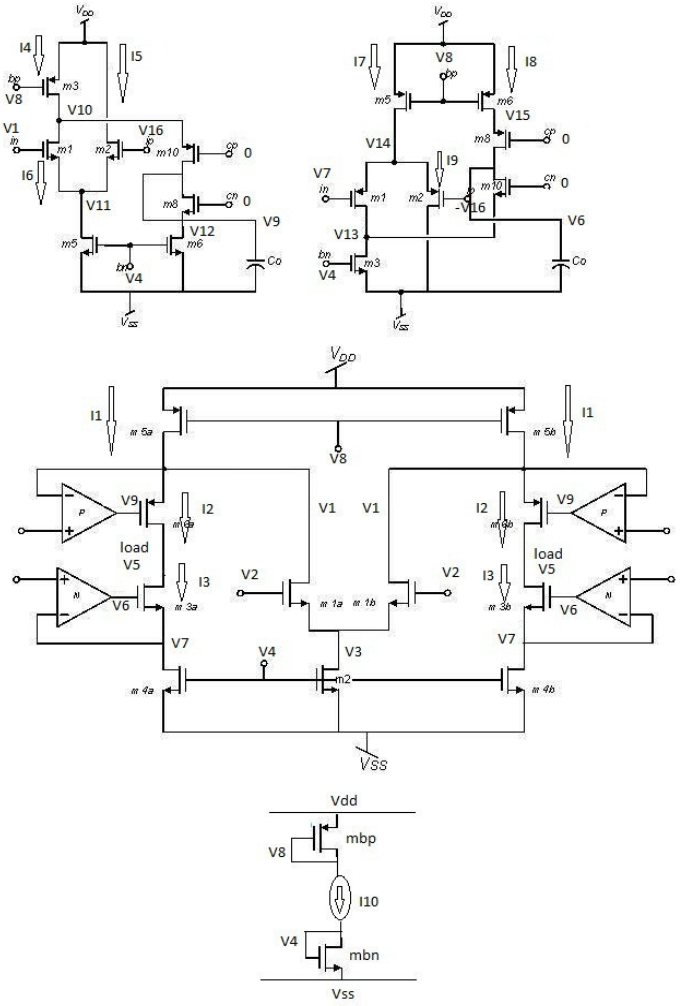

Figure 3.4. The V/I variables for the Gain Boosted Amplifier

The constraint handling has also been implemented during the evaluation of some of the folded cascode tests. The HSpice output, .sp file, gives dm values which refer to the saturation conditions of the transistors. If a transistor is not in saturation, these dm values for the related transistors are added to the performance values of the optimization algorithm, so the algorithm tries to find some new solutions which will lead to a transistor in saturation, since the goal of minimization is not satisfied. As a result, handling constraints is satisfied by using the dm values given by the simulation output file of the HSpice. For the OIOPD method the generation of the V and I values are already being

determined with the saturation conditions, so there is no need for extra saturation constraint handling.

Another trick on the performance value evaluation mechanism has been used for the failed values of the HSpice output. If the HSpice simulator fails to find a result, the code generates a feedback value 1000, which is a high cost value, to the main algorithm. Since the algorithm works on the minimization problem, then this solution is chosen as a bad result and is not used.

4. MULTIOBJECTIVE EVOLUTIONARY ALGORITHM WITH DECOMPOSITION (MOEA/D)

4.1. Definition of MOP

A multi-objective optimization problem (MOP) can be given as follows:

$$\text{maximize } F(x) = \left(f_1(x), ..., f_m(x)\right)^T \text{ subject to } x \in \Omega \tag{4.1}$$

Ω is the decision (design variable) space while $F : \Omega \to R^m$ consists of m real-valued objective functions and R^m is the objective space. The reachable objective set is defined as the set $\{F(x) \mid x \in \Omega\}$ [14].

Ω is described by Equation 4.2 if all the objectives are continuous and $x \in R^n$;

$$\Omega = \{x \in R^n \mid h_j(x) \le 0,\ j = 1, ..., m\} \tag{4.2}$$

h_j here are continous functions, (4.1) is called a continuous MOP. Very often, since the objectives in (4.1) contradict each other, no point in Ω maximizes all the objectives simultaneously. They have to be balanced. The best tradeoffs among the objectives can be defined in terms of Pareto optimality [14].

Let $u, v \in R^m$, it is said to dominate v if and only if $u_i \le v_i$ for every $i \in \{1, ..., m\}$ and $u_j > v_j$ for at least one index $j \in \{1, ..., m\}$. A point $x^* \in \Omega$ is Pareto optimal to (4.1) as long as there is no point $x \in \Omega$ which satisfies *F(x)* dominates *F(x*). F(x*)* is then called a Pareto optimal vector. In other words, any improvement in a Pareto optimal point in one objective must lead to deterioration in at least one other objective, which is the trade-off

concept. Pareto set (PS) is known as the set of all pareto optimal points, and the set of all the pareto optimal objective vectors is the Pareto front (PF) [14].

4.2. The Concept of Decomposition

In this part, implementation of the decomposition method on Evolutionary Algorithms has been introduced. Different techniques for decomposition of the multi-objective problems are also tested and compared to each other.

4.2.1 Implementation of Decomposition on Evolutionary Algorithms

Analog sizing problem is a pareto based problem, a MOP, whose objective function is a function of all the $f_i's$, which are the performance values like gain etc, and this objective function could be an optimal solution of a scalar optimization problem. That means PF approximation can be decomposed into a number of scalar objective optimization subproblems. There are several methods for constructing aggregation functions, $f_i's$, the most popular ones are the weighted sum approach and Tchebycheff approach. Boundary intersection methods have also got lots of attention recently [14].

Most of the MOEA's do not use the concept of decomposition. Instead of associating each individual solution with any scalar optimization problem, these algorithms consider a MOP as a whole. In a scalar objective optimization problem, all the solutions can be compared based on their objective function values and the task of a scalar objective evolutionary algorithm (EA) is often to find one single optimal solution. In MOPs, however, domination does not define a complete ordering among the solutions in the objective space and MOEAs aim at producing a number of Pareto optimal solutions as diverse as possible for representing the whole PF. As a result, conventional selection operators, which were designed for scalar optimization, can not be directly used in nondecomposition MOEAs. If there is a fitness assignment scheme for assigning an individual solution a relative fitness value to reflect its utility for selection, then scalar optimization EAs can be extended for dealing with MOPs, although other techniques such as mating restriction, diversity maintaining etc. Some properties of MOPs, and external

populations may also be needed for increasing the performances of these extended algorithms. Because of that, fitness assignment has been a major issue in current MOEA research. Popular fitness assignment strategies include alternating objectives-based fitness assignment such as the vector evaluation genetic algorithm (VEGA), and domination-based fitness assignment such as Pareto archived evolutionary strategy (PAES), non-dominated sorting genetic algorithm (NSGA-II [8]) and strength Pareto evolutionary algorithm (SPEA-II [9]) [14].

4.2.2 Different Decomposition Methods:

There are several approaches for converting the problem of approximation of the PF into a number of scalar optimization problems. In the MOEA/D algorithm proposed 3 of these decomposition methods are offered. These methods are Weighted-Sum Approach, Tchebycheff Approach and Boundary Intersection Approach. Detailed information about these methods will be given in Chapter 5.

The three approaches above can be used to decompose the approximation of the PF into a number of scalar optimization problems. A reasonably large number of evenly distributed weight vectors usually leads to a set of Pareto optimal vectors, which may not be evenly spread but could approximate the PF very well. There are many other decomposition approaches in the literature that could also be used in our algorithm framework. Since the major purpose is to study the feasibility and efficiency of the algorithm framework, only the above three decomposition approaches are used [14].

4.3. MOEA/D Algorithm

This algorithm has been used as the background work for the enhanced MOEA/D-DE Algorithm proposed. Background work MOEA/D can be checked from the reference number 14. In the following pages, the flow-chart of the algorithm with the details of the optimization method can be found. On 4.3.2, the features of the MOEA/D algorithm are given.

4.3.1 The Framework of the MOEA/D Algorithm

MOEA/D is in a need of decomposing the MOP under consideration. There are several methods that can be used for this. Tchebycheff approach is used for the following descriptions. When the other decomposition methods are used it is very easy to modify the following MOEA/D [14].

Let $\lambda^1,...,\lambda^N$ is a set of even spread weight vectors and z^* is the reference point for the objective functions. The problem of approximation of the PF of (4.1) can be decomposed into scalar optimization subproblems, with number *N*, by using the Tchebycheff approach. The objective function of the k^{th} subproblem is:

$$g^{te}\left(x\middle|\lambda, z^*\right) = \max_{1\le i\le m}\{\lambda_i \left| f_i(x) - z_i^* \right|\} \tag{4.3}$$

for $\lambda^k = (\lambda_1^k,...,\lambda_m^k)^T$. In a single run, the algorithm optimizes (minimizes in this case) all these objective functions simultaneously.

Note that g^{te} is continuous in λ, and that optimal solution of $g^{te}\left(x\middle|\lambda^i, z^*\right)$ should be close to that of $g^{te}\left(x\middle|\lambda^k, z^*\right)$ if λ^i and λ^k are close to each other. This is the neighborhood concept. So, any information about these g^{te}'s with weight vectors close to λ^i should be helpful for optimizing $g^{te}\left(x\middle|\lambda^i, z^*\right)$. This is the basic behind MOEA/D [14].

In MOEA/D, the neighborhood of weight vector λ^i is defined as its several closest weight vectors in $\{\lambda^1,...,\lambda^N\}$. The neighborhood of the i^{th} subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. The best solution of any subproblem is composing the population. Only the current solutions to its neighboring subproblems are used for optimizing a subproblem in MOEA/D. MOEA/D with the Tchebycheff approach includes the following features at each generation.

- A population with size N whose components are $x^1,...,x^N \in \Omega$ since x^i is the current solution to the subproblem i;
- Since z_i is the best value found for the objective f_i, $z=(z_1,...,z_m)^T$.
- $FV^1,...,FV^N$, FV^i is the fitness value for x^i, i.e., $FV^i = F(x^i)$ for all $i=1,...,N$;
- Storing the non-dominated individuals to an extra population which is also known as EP, external population. The algorithm works as follows:

Inputs are:

- MOP as given in Equation 4.1;
- The number of subproblems;
- Stopping criteria;
- N spread of weight vectors $\lambda^1,...,\lambda^N$ which has been generated uniformly;
- The neighborhood number of each weight vector, T.

Output: External Population (*EP*)

First of all, an initialization step exists. In this step, the external population is set to zero. Later, the Euclidean distances between weight vectors are calculated in order to find the T closest weight vectors to each weight vector and the neighborhood $B(i)=\{i_1,...,i_T\}$ is set for the T closest weight vectors λ^i. After that, an initial population, which is the set of solutions, is randomly generated and the objective functions are evaluated for these solution individuals. The minimum and maximum values of $z=(z_1,...,z_m)^T$ for each objective function is set to infinite and minus infinite.

Secondly, the algorithm starts a loop of N turns to realize the updates. First of all the randomly selected two indexes of *B(i)* are used to generate a new solution by using the genetic operators. Later an improvement is applied on the solution. The improved, new solution is used to calculate the objective functions in order to update the z values. Later if

$g^{te}\left(y^{'}\left|\lambda^{j},z\right.\right)\leq g^{te}\left(x^{j}\left|\lambda^{j},z\right.\right)$ is satisfied (*y'* is the new solution), the solution set and the fitness values are updated. After that, *EP* is updated by removing all the dominated vectors by $F(y')$ and including $F(y')$ to the external population if no vectors in *EP* can dominated $F(y')$.

At the last step if the stopping condition is satisfied (it may be max number of iterations) the algorithm stops and outputs the *EP.* Otherwise the update loop goes on.

During the initialization $B(i)$ is determined by the closest *T* vectors of λ^{i}. The closest weight vectors are found out by the Euclidean distance formula. As a result the index *i* will be the first index of $B(i)$. The following *T-1* indexes are determined by the Euclidean distance to i^{th} vector and if an index j is a member of $B(i)$ then it can be told that *j* is a neighbor of *i* [14].

As mentioned before, an initial population is randomly generated and then the reference points of the optimization functions are updated. With this information, the update loop starts. From the $B(i)$ neighborhoods x^{k} and x^{l} are used to generate a new solution and if the solution is better than the parents, then it is copied into the all the neighbors of the related sub-problem. Different indexes have different neighbors, so the information is varied in a parallel (fast) and effective way. The fitness values are also used to update the reference points. When the stopping criteria (which is the maximum number of iterations) is met, then the algorithm stops and outputs the obtained population as a Pareto Set and Pareto Front [14].

4.3.2 The Features of the MOEA/D

The proposed work for the thesis is an enhanced version of the multiobjective evolutionary algorithm based on decomposition (MOEA/D) [14]. The goal of the MOEA/D is decomposing the multi-objective optimization problem into *N* scalar problems which are the subproblems and then solving these subproblems (scalar aggregation function) simultaneously. The solution is satisfied by evolving the indiviuals of the

population. At each generation (iteration in terms of mathematical programming) and for each subproblem the population is composed of the best solution found so far. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors which can be calculated with Euclidean distance. The optimal solutions of two neighboring subproblems should be very similar. Each subproblem is optimized in MOEA/D by using information from its neighboring subproblems. The following features belong to MOEA/D:

• This algorithm is a good method for including decomposition technique into multi-objective optimization problem solving. With the help of this method the decomposition based methods are expanded from mathematical programming to evolutionary algorithms.

• The MOEA based methods try to solve the MOP directly so it can get harder to solve the issues like fitness assigment, diversity maintenance etc. However, by introducing decomposition into MOEA, the optimization is satisfied for N scalar optimization problems and this makes things easier to handle in the framework of MOEA/D.

• Comparing to popular methods like NSGA-II and MOGLS, MOEA/D has lower computational complexity at each generation. Also, it has been proven that MOEA/D outperforms, in terms of solution quality, MOGLS on 0–1 multiobjective knapsack (for more information, check [3]) test instances when both algorithms use the same decomposition approach. MOEA/D with the Tchebycheff decomposition approach performs similarly to NSGA-II on a set of continuous MOP test instances for 2 objective cases. For 3 objective cases MOEA/D is fairly better than NSGA-II. This criteria has been used during the thesis work by comparing NSGA-II with novel methods implemented on Enhanced MOEA/D-DE. MOEA/D using a small population is able to produce a small number of very evenly distributed solutions.

• For large range of objective function values, normalization can be realized on MOEA/D [14].

4.4. Discussions on MOEA/D

There are several subjects to consider about the MOEA/D algorithm in order to understand its quality. The use of finite number of sub problems is an important issue. Also the satisfaction of the diversity is critical in order to find the global result. The neighborhood concept and the complexity of the algorithm compared to other algorithms has to be considered. Details of this part can be checked from [14]. Since the thesis work is based on the flowchart of MOEA/D, all of the discussions given below is also about the MOEA/D-DE algorithm proposed for the thesis work.

I. Why a finite number of subproblems are used in MOEA/D?

The weight vector used in MOEA/D is a previously selected *N* sized one. MOEA/D spends about the same amount of effort on each of the *N* objective functions, while MOGLS (a method used for comparison in this work; for more detailed information see [4]) randomly generates a weight vector at each iteration, which aims to optimize all the possible aggregation functions. What a decision maker needs is taht a finite number of evenly/fairly distributed Pareto optimal solutions; optimizing a finite number of selected scalar optimization subproblems. Since the computational resource is always limited, optimizing all the possible aggregation functions would not be very practical, and also that may waste some computational effort.

II. How is the diversity maintained in MOEA/D?

A multi-objective evolutionary algorithm needs to satisfy diversity in its population for producing a set of representative solutions. The MOEAs which do not use decomposition, like NSGA-II (for detailed information see [2]) and SPEA-II (for detailed information see [5]), use crowding distances among the solutions in their selection to maintain diversity, but it is not always easy to generate a uniform distribution of pareto optimal objective vectors in such algorithms. As mentioned several times, a MOP is decomposed into a number of scalar optimization subproblems in MOEA/D. Different solutions in the current population are related to different subproblems. The “diversity”

among these subproblems will of course create the diversity in the population. As long as the decomposition method and the weight vectors are properly/uniformly chosen, and thus the optimal solutions to the resultant subproblems are evenly distributed along the PF, this algorithm will have a good chance of producing a uniform distribution of pareto solutions if it can optimize all these subproblems well.

III. Role of neighborhood size *T* in MOEA/D and mating restriction:

As mentioned before, *T* is known as the size of the neighborhood in a population of *N* individuals. Only current solutions to the closest neighbors of a subproblem are used for being optimized in MOEA/D. In other words, two solutions have a chance to mate just when they are for two neighboring subproblems. This is known as mating restriction. Setting of *T* is an important subject. If this value is too small, two solutions (x^k and x^l) chosen for undergoing genetic operators may be very similar (close to each other) since they are for very similar subproblems, as a result, the solution generated could be very close to their parents. As a result, the algorithm loses the ability to explore new areas in the search space. On the other hand, a too large *T* can make the chosen two solutions poor for the subproblem under consideration, and so a too large *T* can also make their offspring poor. Finally, the benefiting ability of the algorithm gets weaker. It should also be noted that a too large *T* will increase the computational overhead of updating the neighboring solutions.

IV. Comparison of computational complexity of the MOEA/D and NSGA-II:

In MOEA/D, the major computational expenses are in the update steps. In these steps MOEA/D generates *N* trial solutions, like NSGA-II does at each generation. Note that updating the reference points perform *O(m)* comparisons and assignments, and updating the neighboring solutions need *O(mT)* basic operations since its major costs are to compute the values of g^{te} for *T* solutions since the computation of one such a value requires *O(m)* basic operations. As a result, if both MOEA/D and NSGA-II use the same population size, at each generation, the ratio of their computational complexities will be:

$$\frac{O(mNT)}{O(mN^2)} = \frac{O(T)}{O(N)} \tag{4.4}$$

Since T is smaller than N, the MOEA/D algorithm has lower computational complexity than NSGA-II at each generation [14].

5. ENHANCED MOEA/D-DE ALGORITHM PROPOSED

5.1.Introduction to MOEA/D-DE

In this Part an introduction to the Enhanced MOEA/D-DE Algorithm will be given where [14] has been used as a background work. Also, the enhancements realized to make it more powerful and able to solve the analog sizing problem will be introduced [15].

5.1.1. The Background Work and Introduction to MOEA/D-DE

In the first part of the project, a multiobjective evolutionary algorithm based on decomposition (MOEA/D) [14] with its extended version by using differential evolution (DE) as the main search engine (MOEA/D-DE) has been proposed. This method outperform several widely used multiobjective evolutionary algorithms. MOEA/D-DE decomposes a multiobjective problem into a number of scalar optimization sub-problems with a neighborhood structure and optimizes them simultaneously to approximate the Pareto-optimal set. In this work, additional to MOEA/D lots of mechanisms are investigated to enhance the performance of MOEA/D-DE. Firstly, the MOEA/D algorithm code has been simplified to improve the performance in terms of software quality. Later on, a novel method for generating the weight vectors has been proposed and it has been observed that it enhances the overall quality of the Pareto Fronts. After that, different normalization methods for the objective functions have been implemented and the best solution has been determined. Later on different decomposition methods have been performed to find the best one. For the reproduction of the new individuals, a new replacement mechanism is proposed to call for a balance between the diversity of the population and the employment of good information from neighbors. Secondly, DE search algorithm has been added instead of polynomial mutation of the MOEA/D and the scaling factor in DE is randomized to enhance the search ability. Comparisons are carried out with MOEA/D-DE on ten benchmark problems, showing that the proposed method exhibits significant improvements. Finally, the enhanced MOEA/D-DE is applied to a real world problem, the sizing of a folded-cascode amplifier with four performance objectives.

Many real-world optimization applications involve several conflicting objectives. According to different goals and requirements in the decision-making process, multiobjective optimization techniques can be roughly classified into two categories: (1) a priori methods: a decision maker specifies their preferences on these objectives and so transform the multiobjective problem into a single objective one by using aggregation methods, and (2) a posteriori methods: they produce a number of well representative optimal trade-off candidate solutions for a decision-maker to check. These had been mentioned on Chapter 2. A Pareto optimal solution is a candidate solution for achieving the best trade-off. There can be many, even infinite Pareto optimal solutions to a multiobjective optimization problem (MOP). The set of all the Pareto optimal solutions is called the Pareto set (PS) and its image in the objective space is the Pareto front (PF). Most multiobjective optimization evolutionary algorithms (MOEA) aim at finding a reasonable number of solutions to approximate the PF. This means that MOEA's are members of a posteriori methods. This had been mentioned before. Most MOEAs compare solutions based on dominance. However, domination can not provide a full ranking among all the solutions. Therefore, these MOEAs need some other techniques for ranking solutions (e.g. crowding distances, fitness sharing, niching). Among these algorithms, non-dominated sorting genetic algorithm II (NSGA-II) [8] and strength Pareto evolutionary algorithm 2 (SPEA2) [9] have received much attention in real world applications. However, it is shown that these methods cannot always provide good results, especially when the MOP is complicated.

Recently, a new MOEA framework, multiobjective evolutionary algorithm based on decomposition (MOEA/D) [14], was proposed. It decomposes a MOP into a set of scalar optimization sub-problems with neighborhood relations. The neighborhood relations are defined by the distances between their aggregation coefficient vectors. In this way, the fitness assignment is the same as single objective optimization, and the diversity is maintained by the diverse search directions determined by the uniformly distributed weight vectors. The first version of MOEA/D uses simulated binary crossover (SBX) and polynomial mutation as the search engines.

5.1.2. Enhancing the Algorithm Quality?

Several studies have been carried out to enhance the performance of the MOEA/D framework. These can be classified as below:

I. Improving the algorithm in terms of software speed:

First of all the software code (Matlab) of the MOEA/D has been simplified to prevent some extra computational work. Later, the overall algorithm has been transformed into script files instead of functions to make them work faster and to be able to follow the parameters easily. Another work in this topic was deleting the structs which slows down the algorithm and writing the parameters seperately. During this phase, the speed of the software code has been increased and nothing related to algorithm improvement has been realized. After these updates, comparing the speeds of the old version MOEA/D code and new MOEA/D code is as given in Table 5.1.

Table 5.1. The Comparison of the Algorithm Speed Before and After Software Enhancement

First Version MOEA/D code	0,41 sec
Updated MOEA/D code	0,26 sec

These are the average time of 100 optimizations of a 2 objective benchmark problem. As seen the speed of the algorithm has been increased by %36 as compared to the first version.

II. A novel method generating the weight vectors:

Initialization of the weights has the utmost importance in order to have a reasonable solution range. By assigning well spread weights to each objective makes the solution space spanning both extreme points. Otherwise the solution might get stuck around a limited region. A new method has been proposed for generating the weight vectors. This method is a composition of an orthogonal array based method and a LUT. The novel

method increases the dominance and distribution quality of the MOEA/D. Detailed information about this work can be found in Part 5.2.1.

III. Finding the best Normalization Method:

For the decomposition based MOEA methods, fair optimization of each objective is an important issue. If the range of a fitness value (the result of an objective function) is smaller than the others, the optimization probability of that function gets smaller. As a result all of the objective functions have to be equalized to the same range, which is [0 1], for the normalization case. This will make the optimization of different functions fair. Different normalization methods have been performed and compared to see the best one. Detailed information about this work is given in Part 5.2.2

IV. Finding the best Decomposition Method:

When the objective is an aggregation of all the $f_i's$, it is known that a Pareto optimal solution to a MOP, under mild conditions, could be an optimal solution of a scalar optimization problem. As a result, PF approximation can be decomposed into a number of scalar objective optimization subproblems. There are several methods for constructing aggregation functions, and the most popular ones among them include the weighted sum approach and Tchebycheff approach. Recently, the boundary intersection methods have also attracted a lot of attention [14]. All of these three methods recommended by the MOEA/D authors have been performed and compared to each other in order to find the best method. More information about the methods and the tests done can be found in Part 5.2.3.

V. Enhancing the search ability:

To enhance the search ability of the MOEA/D, a DE search algorithm has been added instead of the mutation method of the MOEA/D. Also the scaling factor in the DE mutation has been randomized to achieve this. In the proposed algorithm, a new version using the mutation (DE/best/1/bin [5]) in differential evolution (DE) as the main search

engine was proposed and shown to outperform MOEA/D and NSGA-II, especially for complex problems. Detailed information is given in Part 5.2.4.

VI. A new replacement mechanism:

Another work to enhance the quality of the MOEA/D is on the population replacement. The goal is to call for a balance between information sharing and diversity maintenance. In the proposed method, when the number of parent solutions that can be replaced by a high quality child solution exceeds the maximum number, the parent solutions are ranked and those that are closer to the child solution are firstly replaced. This novel method increases the quality of the dominance and distribution of the Pareto Set. More details can be found in Part 5.2.5.

After all these improvements on MOEA/D framework, a novel method called Enhanced MOEA/D-DE is obtained. This algorithm is quite powerful in terms of the solution dominance, distribution quality, convergence speed and the range of the objective functions on the Pareto Front.

5.2.The Enhancements Realized

In this Section, the enhancements realized for improving the MOEA/D Algorithm will be discussed. They were mentioned in Section 5.1; however this time they will be explained in detail and with the tests using folded cascode amplifier and several benchmark problems.

5.2.1. A Novel Method for the Generation of the Weight Vectors

All of the decomposition methods used for MOEA optimization algorithm which are weighted sum approach, Tchebycheff approach and boundary intersection (BI) approach use a weight vector set λ_m^N where m is the number of the objective functions and N is the number of the subproblems. Basically it can be told that weight vectors are used as a method of decomposition of different subroblems into a single subproblem. Let

$\lambda=(\lambda_1,...,\lambda_m)^T$ be a weight vector, for $\lambda_i \geq 0$ and m be the number of the objective functions, $\sum_{i=1}^{m} \lambda_i = 1$ should be satisfied for each individiual of the population to set the weight matrix.

With weight matrix, each objective function to be optimized will have a different weight value which will make it easy or hard to be optimized compared to other objective functions for the related individual of the whole Pareto front. For example in a MOP with 2 objective functions to be minimized, in the final Pareto front which has N individuals there will be N weight vectors which include 2 weight matrix values whose sum is equal to 1. As seen below for the left side of the Pareto set, objective function 2 will be minimized like a single objective optimization case since the weight vector λ^1 is just taking care of the second objective function. λ^N will be working with the same logic to minimize objective function 1. An internal weight vector like $\lambda^2 = [0.1, 0.9]$ will be helping the MOEA to minimize both of the objective functions. However, for the fair conditions of the trade-off of these objective functions, objective function 2 will have more chance to be minimized since its weight value is higher.

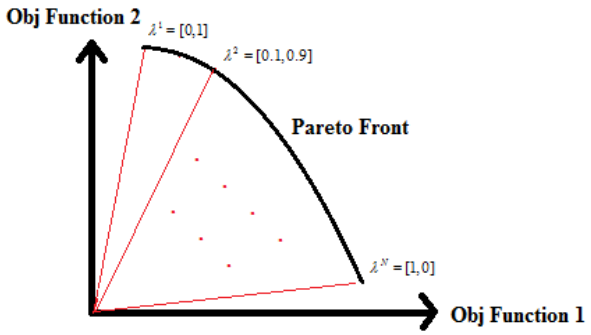

Figure 5.1. The Effects of the Weight Matrix on Fitness Functions

As seen above, evenly distributed weight matrix will let the Pareto front to be evenly distributed and will let the limit conditions of the Pareto set (like smallest obj function 1 value or obj function 2 value for the figure) to have a higher range. Good Pareto optimal solutions can be obtained by weight vectors and if the weight vectors are altered, the Pareto optimal results will also change.

As mentioned above, initialization of the weights has the utmost importance in order to have a reasonable solution range. By assigning well spread weights to each objective, the solution space extends to the both extreme points. Otherwise the solution might get stuck around a limited region.

In the previous work used as a background work [14], the weight matrix is just assigned for 2 objective optimization problem. The weight matrix initialization logic is very simple. For N individuals, the weight matrix is in the form of $\lambda^a = [1-(a-1)/(N-1),(a-1)/(N-1)]$ where a is the number individual which can vary between 1 and N, the population size.

$$\begin{aligned}
&\lambda^1 = [1,0] \\
&\lambda^2 = [1-1/(N-1),1/(N-1)] \\
&\lambda^3 = [1-2/(N-1),2/(N-1)] \\
&. \\
&. \\
&\lambda^{N-1} = [1-(N-2)/(N-1),(N-2)/(N-1)] \\
&\lambda^N = [0,1]
\end{aligned} \tag{5.1}$$

Since this is the best distribution which can be obtained for the case of two objective functions, this weight matrix initialization method has been used in the algorithm proposed for two objective function cases.

For more than two objectives there are several ways for creating the weights. First of all, the orthogonal genetic algorithm offered in [16] is implemented. It has been proved that the orthogonal design is optimal for additive model and quadratic model, and the selected combinations are good representatives for all the possible combinations. This method proposes to create weight matrices by randomly distributing the possible candidates. The basic algorithm of the method includes a selection of the levels which will lead to the weight values and number of the factors which mean the number of the objective functions. In the following Figure 5.2 it is shown a $L_9(3^4)$ orthogonal array with 3 levels and 4 factors.

Combination	Factor 1	Factor 2	Factor 3	Factor 4
1st	1	1	1	1
2nd	1	2	2	2
3rd	1	3	3	3
4th	2	1	2	3
5th	2	2	3	1
6th	2	3	1	2
7th	3	1	3	2
8th	3	2	1	3
9th	3	3	2	1

Figure 5.2. Orthogonal Array Example with Different Factors and Combinations [16]

The level number means how good the related objective function will be optimized. For a combination, lets say with level values 2,1,2,3 of the four factors, the weight vector will be 0.25, 0.125, 0.25, 0.375 (since the sum should be equal to 1) which is the ratio of each weight to the overall weight. This is kind of a normalization of the levels to the weight vector form.

The basic idea behind the orthogonal array is selecting a level limit and distributing these levels between objective functions with a method which decreases the number of the overall possibilities. In Figure 5.2 the combinations are decreased to 9, which means with a level of 3 and 4 objective functions, a solution is offered with 9 weight vectors, making the population size. Increasing the level number to 10 will increase that population size to 100 and it will create a larger search space and better distribution.

As mentioned above, by applying orthogonality to an array, the total number of combinations can be decreased to an acceptable number. For example, in Figure 5.2 above, 9 points are used instead of all 81 possibilities. The orthogonality of an array means that 1) for the factor in any column, every level occurs the same number of times; 2) for the two

factors in any two columns, every combination of two levels occurs the same number of times; and 3) the selected combinations are uniformly distributed over the whole space of all the possible combinations [16].

Figure 5.3 below shows the decrease in the number of all possibilities to an acceptable number of possibilities. The orthogonal array tries to generate fairly distributed vectors in the whole space.

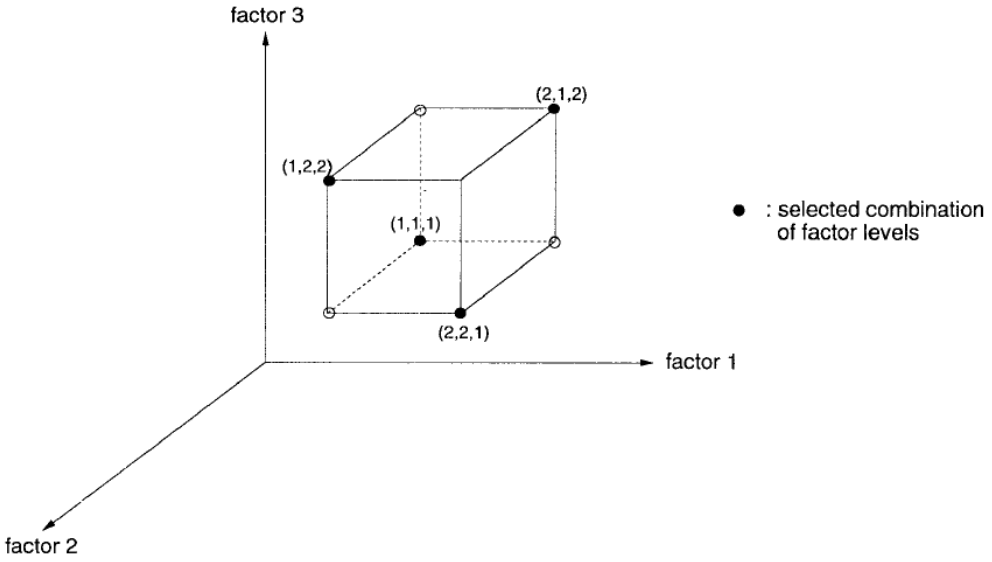

Figure 5.3. Orthogonality of the Orthogonal Array $L_4(2^3)$ where 4 Refers to Final Number of the Vectors, 2 Refers to Number of the Levels and 3 is for the Number of the Sub-problems [16]

However, this random process does not concern the trade-offs very well. In other words, by unfair distribution, some significant objectives might be overlooked and its search space might be limited. For example, in Figure 5.3, for the first objective function the highest weight values will be obtained at 7^{th} and 8^{th} combination which are $\frac{3}{9} = 0.33$. This is not enough for the optimization range of the first function since the avarage weight value it has is already 0.25. Thus, even if the number of the levels is increased, it is clear that the limit conditions of the Pareto optimal solutions will get stuck due to the weak weight matrix initialization.

In order to avoid this situation, the weights for more than two objectives problems are initialized manually as a look-up table by concerning both fair distribution of the weights and the three rules mentioned for orthogonal array which leads to good distribution.

The weight initialization part of the final algorithm first checks the number of the objectives. If there are just two objectives, the previously mentioned weights matrix is exploited. In other cases , the manually initialized weight matrices which support up to seven objectives are called. If it is more than seven objectives, the rest are created by the orthogonal array method. Also the look-up table is prepared for up to 150 population size. For more than 150 population size the rest of the weight matrix is filled with an orthogonal array initialization. As a result the final version of the weight initialization may be a combination of a look-up table and orthogonal array implentation.

The four objective case is illustrated below for a population size of 20. While creating this matrix manually, some points are taken into account. First, the importance of the objectives are equally distributed. Each objective must be dominant over others once as seen at first 4 rows of the matrix. Then this dominance is shared equally by making them same like the 5th row. The rest of the rows are written by changing the importance level of the objectives by making sure that almost every possible value is assigned.

As a result the algorithm for initalizing the weight matrix works as follows:

First check the number of the objective functions, if it is equal to 2, then use the method given in Equation 5.1. If the number of the objectives are more than 3 then first load the LUT for the related number of the objectives. This LUT stores a data till 150 population size. If the population size is x which is less than 150, then the first x lines of the LUT are used. If the population size is more than 150, first of all the whole LUT is used, and the remaining lines are generated by the orthogonal array method.

Illustrative Example :

Table 5.2. An Example for a 4 Objective Weight Matrix Initialization

Obj1	Obj2	Obj3	Obj4
1,00	0,00	0,00	0,00
0,00	1,00	0,00	0,00
0,00	0,00	1,00	0,00
0,00	0,00	0,00	1,00
0,25	0,25	0,25	0,25
0,70	0,10	0,10	0,10
0,10	0,70	0,10	0,10
0,10	0,10	0,70	0,10
0,10	0,10	0,10	0,70
0,30	0,30	0,30	0,10
0,10	0,30	0,30	0,30
0,30	0,10	0,30	0,30
0,30	0,30	0,10	0,30
0,40	0,40	0,10	0,10
0,40	0,10	0,40	0,10
0,40	0,10	0,10	0,40
0,10	0,40	0,10	0,40
0,10	0,40	0,40	0,10
0,10	0,10	0,40	0,40
0,30	0,30	0,20	0,20

The results comparing the orthogonal method with the proposed method for 10 different tests of global and local normalization methods are as given in Table 5.3. The comparison metric is the IGD value. It can be seen that the proposed method is much better than the orthogonal array method.

Table 5.3. Comparisons between the Orthogonal Array Method and the Proposed Method

Method	Normalization	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Average
Orthogonal	Global	0.3022	0.3094	0.2049	0.3629	0.2542	0.4101	0.3742	0.3068	0.3548	0.3310	0.3211
Proposed	Global	0.1483	0.1504	0.1439	0.2103	0.1394	0.1663	0.1236	0.1971	0.1343	0.1373	0.1551
Orthogonal	Local	0.0328	0.0349	0.0413	0.0411	0.0367	0.0458	0.0395	0.0464	0.0335	0.0416	0.0394
Proposed	Local	0.0285	0.0357	0.0324	0.0351	0.0282	0.0376	0.0276	0.0550	0.0311	0.0362	0.0347

For a four objective analog sizing problem of folded cascode amplifier, some of the comparisons of two objectives are given in Figure 5.4. and Figure 5.5.

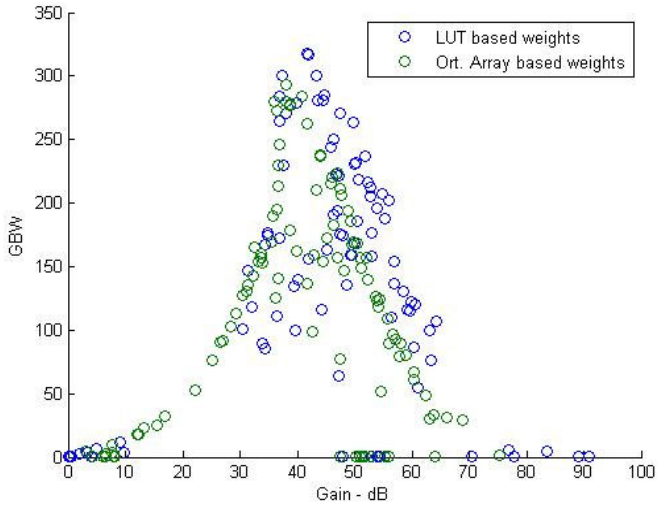

Figure 5.4. Comparison of Two Weight Matrix Initialization Methods for 4 objective Analog Sizing Problem, Geometric Projections of the Gain and GainBandwith Objective Functions

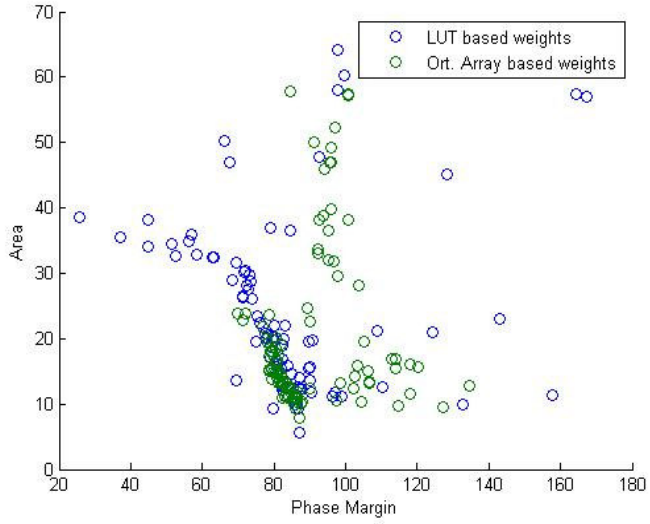

Figure 5.5. Comparison of Two Weight Matrix Initialization Methods for 4 objective Analog Sizing Problem, Geometric Projections of the Phase Margin and Area Objective Functions

As seen from the Figure 5.4 and Figure 5.5, the results with LUT based method dominates the ones with orthogonal array method since all of the objective functions have been optimized much better, like higher gain value reached for the same gain-bandwidth product etc.

Also the basic goal of the new method which is increasing the range of the objectives is also satisfied. The ranges of the objective functions for the figures above are given in Table 5.4.

Table 5.4. The Ranges of the Objective Functions with 2 Different Methods of Weight Matrix Initialization

	LUT based	**Ort. Array based**
GAIN	0,211 - 91,001	3,0879 - 75,197
GBW	0,0588 - 318,12	0,267 - 292,85
PM	25,518 - 167,41	70,005 - 134,4
AREA	5,5275 - 64,145	7,845 - 57,757

As seen above all limit values (min or max) for all of the objective functions are better for the new method.

The novel method described is basically using an hand-made LUT data which especially shows the effects of the extreme points (for the goal of single objective optimization) on the weight vectors. However such a LUT should be implemented with respect to some sampling methods from the literature. A good example for such a sampling technique is known as Latin Hypercube Sampling (LHS).

Latin Hypercube sampling, LHS, is an option which is now available for most simple risk analysis simulation software programs. It uses a technique known as “stratified sampling without replacement” [35]. The probability distribution is divided into n intervals of equal probability, where n is the number of samples that are to be performed on the model. As the simulation runs, each of the n intervals is sampled once.

The advantage of the LHS is that, it generates a set of samples that more precisely reflect the shape of a sampled distribution than pure random (Monte Carlo) samples. The general effect is that, the mean of a set of simulation results more quickly approaches the "true" value, particularly for models that are simply subtracting or adding a number of variables.

Extreme values of the weight vectors of the objective functions for the Latin Hypercube Sampling and Orthogonal Array is given in Table 5.5.

Table 5.5. Extreme values of the weight vectors for Latin Hypercube Sampling and Orthogonal Array Method

Method / Number of Objectives	Max.Weight of Obj1	Max.Weight of Obj2	Max.Weight of Obj3	Max.Weight of Obj4	Max.Weight of Obj5
Latin Hypercube - 3 objectives	0,7579	0,7983	0,7663		
Latin Hypercube - 4 objectives	0,6242	0,6775	0,7228	0,7179	
Orthogonal Array - 4 objectives	0,6667	0,5833	0,5625	0,6429	
Latin Hypercube - 5 objectives	0,4267	0,4649	0,5774	0,4749	0,4875
Orthogonal Array - 5 objectives	0,4615	0,4545	0,5263	0,5625	0,4545

Weight vectors generated for 3 objective case with Latin Hypercube Sampling and Orthogonal Array are given in Figure 5.6 and Figure 5.7.

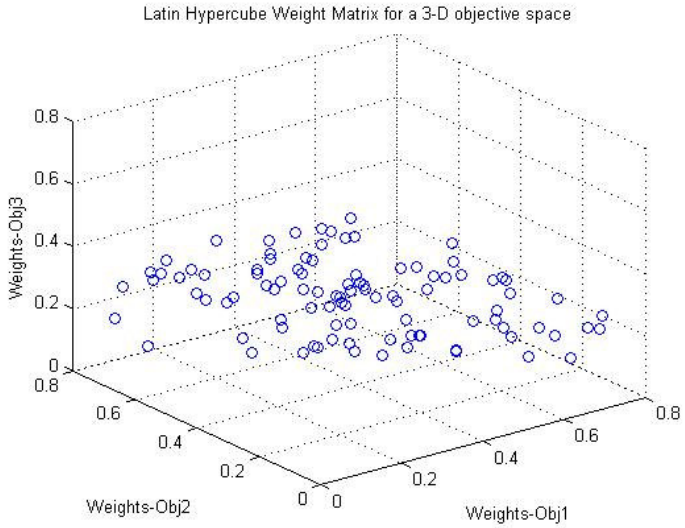

Figure 5.6. Latin Hypercube Sampling Weight Matrix for 3 objectives

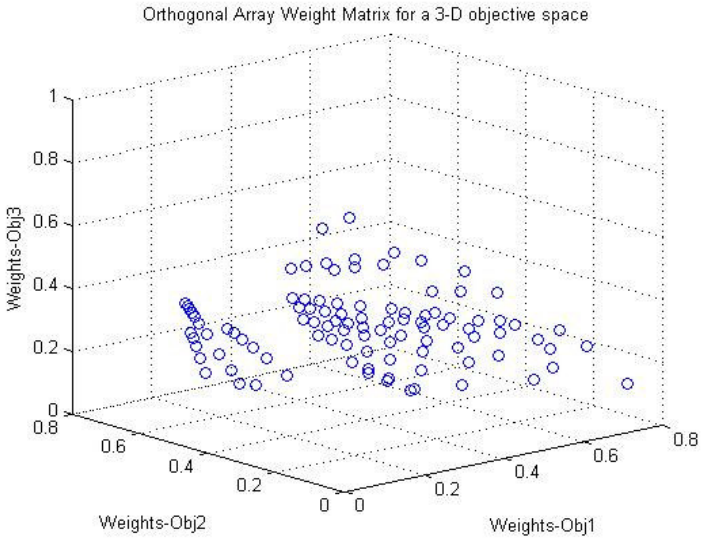

Figure 5.7. Orthogonal Array Weight Matrix for 3 objectives

As a result, a novel method for generating the weight matrix has been proposed. This method uses the combination of a LUT and an Orthogonal Array based method. The test results show the effects of the new method on the dominance, distribution quality and range of the objectives on PFs are better than the methods which already exist.

The weight values for the first and second objectives are distributed as given in Figure 5.8, Figure 5.9 and Figure 5.10 for all three methods.

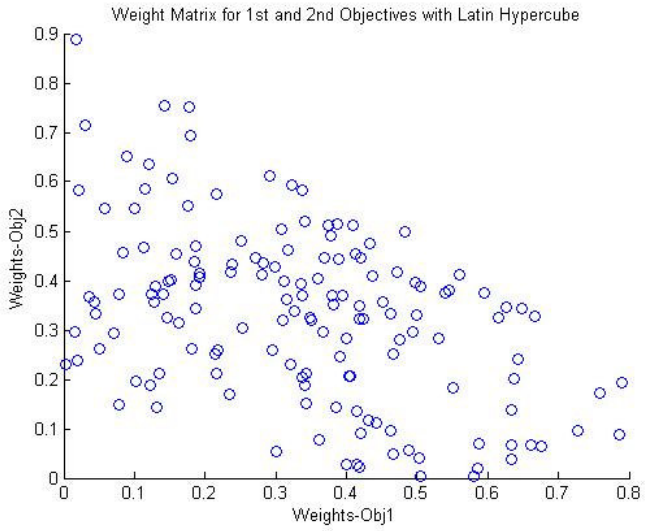

Figure 5.8. Latin Hypercube Sampling for the first two objectives

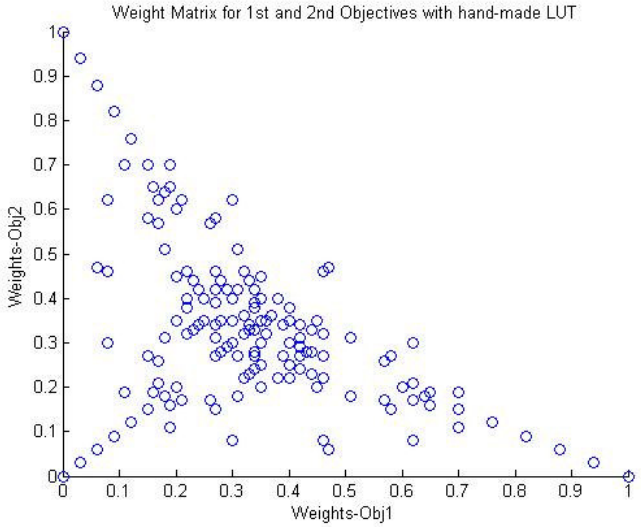

Figure 5.9. Hand-made LUT for the first two objectives

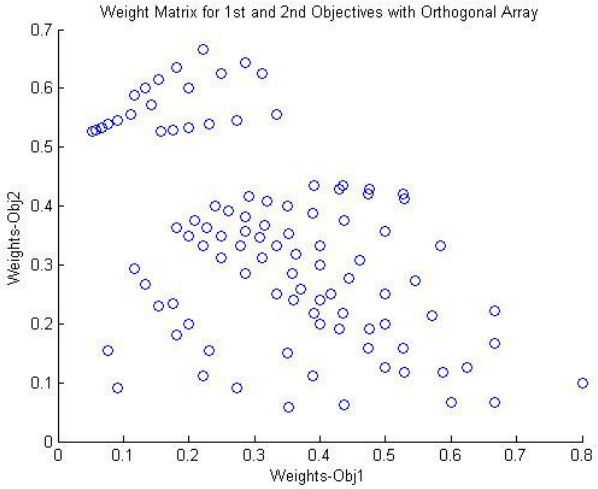

Figure 5.10. Orthogonal Array Method for the first two objectives

5.2.2. Finding the Best Normalization Method

For the decomposition methods of the MOEA-D, the function to be minimized is linearly dependent on the values of the objective functions. For example, in the case of Tchebycheff Approach, it is as follows:

$$\text{minimize } g^{te}\left(x\middle|\lambda, z^*\right) = \max_{1\le i\le m}\{\lambda_i \left|f_i(x) - z_i^*\right|\} \text{ subject to } x \in \Omega \qquad (5.2)$$

where Ω is the decision space which is also known as the population of Pareto optimal points, $z^* = \left(z_1^*, ..., z_m^*\right)^T$ is the reference point since $z_i^* = \max\left(f_i(x)\middle|x \in \Omega\right)$ for each $i = 1, ..., m$

As seen, the scalar optimization function is directly dependent on the minimum values (z) obtained in the population and also the objective function values. Let's assume that for an analog sizing problem, the objective function gain changes in the range of 0-100 dB and power changes in the range of 0.1-10 mW. An average value of gain which is 50 will be having a value of 50 for $\left|f_i(x) - z_i^*\right|$ function form. Here it was assumed that the objective function was turned into a minimization problem (by multiplying the function values with a minus) case, so z value will be -100 dB and f_i will be -50 dB. For the power calculation, 5 mW is an average value of f_i and for $z = 0.1$, $\left|f_i(x) - z_i^*\right| = 4.9$ will be obtained. For the fair weight values, between these two objectives the minimization possibility of the gain will be much higher than the power. Even if the power has a weight value of 0.9 and gain will have 0.1 the minimization will be focusing on the gain. In conclusion, different ranges and limit values of different objective functions will be creating an unfair minimization enviroment.

To avoid this problem, the objective functions are supposed to be scaled into the same range, for example [0,1], which is also known as normalization. This normalization has been implemented on the fitness values of the analog sizing problem. A general function of the normalization of f_i is as given in Equation 5.3.

$$\overline{f^i} = \frac{f_i - z_i^*}{z_i^{nad} - z_i^*} \qquad (5.3)$$

z_i^{nad} is used for the nadir points for the related objective and $\overline{f^i}$ is for the updated objective function values.

According to z^* values chosen, the normalization methods can be generalized into two main groups which are global and local normalization.

I. Global Normalization:

For the selection of the new objective functions which will be scaled to [0,1] range, the z^* reference values can be chosen prior to algorithm run. For this case a single objective optimization algorithm, MSOEA [17] was run to find the limit values of the objective functions to be used for the analog sizing problem. For the 7 objectives of the analog sizing of folded cascode amplifier the extreme values found by MSOEA with 100 population size, 1000 iterations and chosen values for the global normalization method are as given in Table 5.6 below.

Table 5.6. Limit Values of the Objective Functions

	Max obtained	Min obtained	Max used	Min used
Gain	110	-44	120	0
Gain Bandwidth	460	-7,55	380	0
Phase Margin	172	7	180	0
Output Swing	7,8	0,025	2,5	0
Slew Rate	324	-61	324	0
Power	6,9	-1,2	6,9	0
Area	109	0,038	110	0

In the multi-objective optimization trials with local or global normalization (with tens of different extreme values tried), it was found out that normalization becomes fairer with the values selected above. For example, for a multi-objective optimization problem even a GBW value is set to 460 MHz, it never goes over 380 MHz. It was also experimented that setting the minimum values to 0 is not only logical but also much better in terms of global normalization based MOEA-D. A method for this reason has been

realized as evolving new individuals for the negative evalulation results of objective functions. This is kind of a direct way of constraint using.

II. Local Normalization:

Another method for the normalization is that, the z^* reference values can be chosen during the algorithm run. It is also same for the nadir (worst case) points. This can be satisfied by updating the extreme values obtained by the simulation results (which are the objective function evaluation values). This update can be realized after a generation or inside a generation. According to that, local normalization can be classified into two sub groups. The first one is called "local so far normalization" which updates the extreme values after each generation and uses the updated value for the next generations. If it is updated again, a new value will be used for the following generations. Another method called "local normalization in current population" was also performed. In this method the z* reference values and nadir points are updated and used inside the current population. For the new generation the extreme values are starting with $\pm\infty$ and being updated for the first evaluation of objective functions. These updated values are used until they are updated again inside the current population till the next generation starts.

First, global normalization and "local so far normalization" were compared in terms of dominance and distribution quality. Later on local normalization methods were compared with each other. For the more realistic test of each method and for better comparison, 10 tests were run with each having 150 population size, 60 niches and 150 iterations. Each method uses MOEA-D algorithm to optimize four analog objectives which are gain, gain bandwidth product, phase margin and area of the folded cascode amplifier. Later on, a true Pareto front (a final pareto front with non-dominated points) was obtained by these 30 tests (10 for each method) so IGD values and number of dominated points have been calculated. The results are as follows:

Table 5.7. Average IGD Values for Global and Local Normalization

	Global Norm.	**Local Norm.**
Average IGD	0,3211	0,0394

Table 5.8. Number of Non-dominated Points for Global and Local Normalization

(max 10*150 = 1500)	**Global Norm.**	**Local Norm.**
Non-Dominated Points	367	1239

The comparison of methods by the projection of the objectives on 2D space are as shown in Figure 5.11, Figure 5.12 and Figure 5.13.

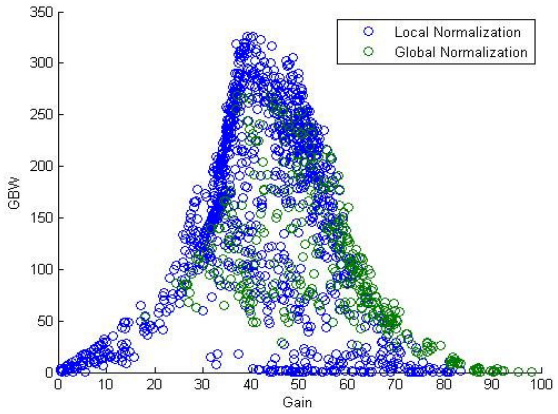

Figure 5.11. Gain-GBW objectives for 4-objective Optimization with Local and Global Normalization Methods

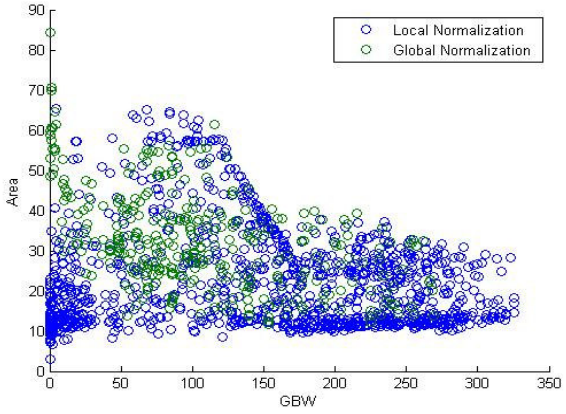

Figure 5.12. GBW-Area objectives for 4-objective Optimization with Local and Global Normalization Methods

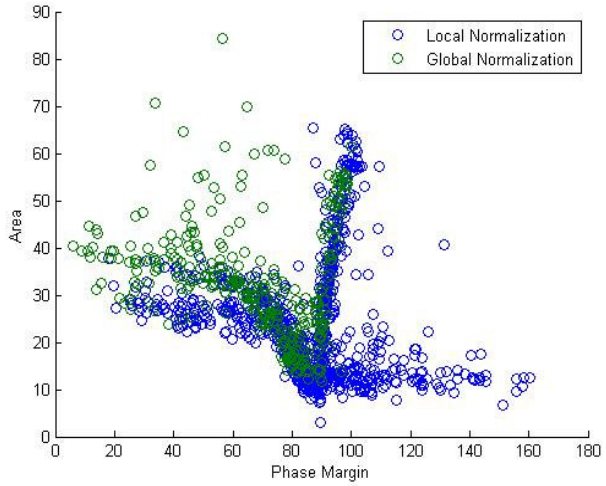

Figure 5.13. Phase Margin-Area objectives for 4-objective Optimization with Local and Global Normalization Methods

As seen from the three figures above, local normalization generally dominates the results of global normalization. The goal is to maximize the phase margin, gain, gbw and minimize the area objectives. Also the ranges of the Pareto optimal results differ a lot from local to global normalization. Local normalization is much better in terms of range and thus distribution quality.

Since the “local so far normalization” yielded much better results than the global normalization, a further method based on “local normalization in current population” was also implemented. In this method the ideal points (minimum fitness values) and the maximum points of the objective functions are updated inside the population. In other words, the reference points of the scalar optimization function starts from infinite values and update themselves on each generation.

Figure 5.14, Figure 5.15 and Figure 5.16 show the comparisons between two methods of the local normalization. It can be easily told that “local normalization in current population” is not even comparable with “local so far normalization” for distribution quality, range and dominance quality.

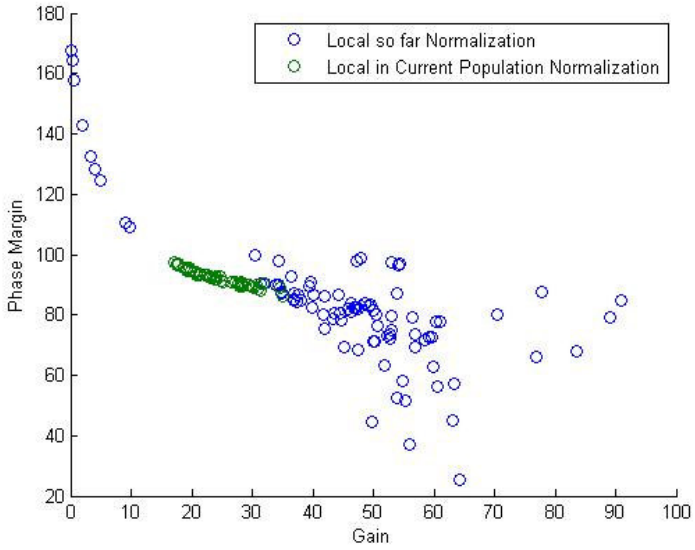

Figure 5.14. Gain-Phase Margin objectives for 4-objective Optimization with 2 different Local Normalization Methods

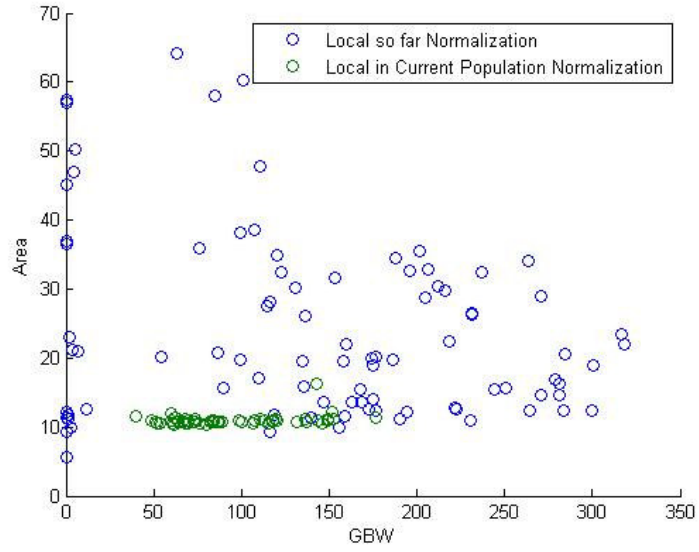

Figure 5.15. GBW-Area objectives for 4-objective Optimization with 2 different Local Normalization Methods

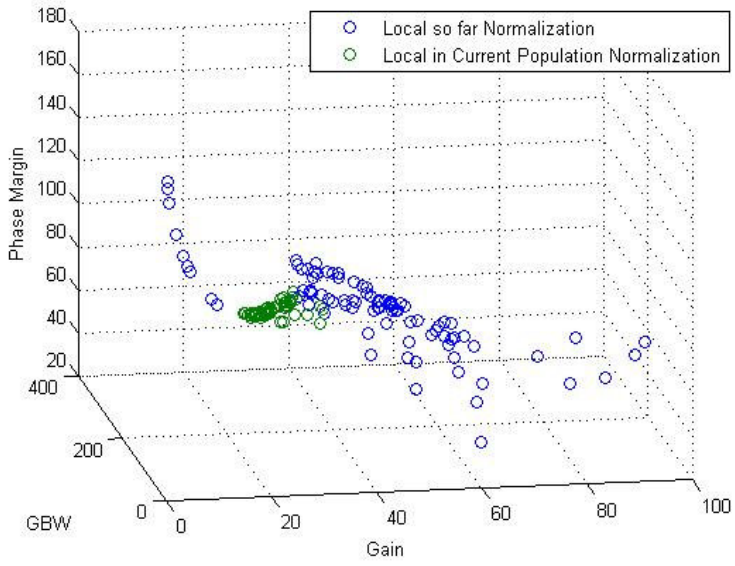

Figure 5.16. Gain-Phase Margin objectives for 4-objective Optimization with 2 different Local Normalization Methods

As a result, normalization is an important subject for fair optimization of the objective functions in such a decomposition based algorithm. Different methods were implemented and compared to each other to see the effects on the Pareto front about the solution's distribution quality, range and dominance quality. The local normalization based on the update of extreme values for all generations, performed better than the other methods and that was proved by a large number of tests.

5.2.3. Finding the Best Decomposition Mehod

There are several approaches for converting the problem of approximation of the PF into a number of scalar optimization problems. In the MOEA-D algorithm [14], 3 different ways of decomposition methods has been mentioned.

I. Weighted Sum Approach:

A convex combination of the different objectives is considered. If $\lambda = (\lambda_1, ..., \lambda_M)^T$

is a weight vector, i.e., $\lambda_i \geq 0$ for all $i = 1,...,m$ and $\sum_{i=1}^{m} \lambda_i = 1$. As a result, the optimal solution to the following scalar optimization problem is as given in Equation 5.5.

$$\text{maximize } g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i . f_i(x) \text{ subject to } x \in \Omega \tag{5.5}$$

$g^{ws}(x|\lambda)$ is used to mention that λ is a coefficient vector in this objective function, since x are the design variables to be optimized. To generate a set of different Pareto optimal vectors, different weight vectors λ in the above scalar optimization problem can be used [14].

The biggest disadvantage of the approach is that the optimization function g will focus on optimizing the f_i with highest value. There is no minimum/maximum reference for the optimization algorithm to converge so lots of efforts have not been shown for this approach.

II. Tchebycheff Approach:

In this approach, the scalar optimization problem is as given in Equation 5.6:

$$\text{minimize } g^{te}(x|\lambda, z^*) = \max_{1 \leq i \leq m} \{\lambda_i | f_i(x) - z_i^*|\} \text{ subject to } x \in \Omega \tag{5.6}$$

here Ω is again the decision space, $z^* = (z_1^*,..., z_m^*)^T$ is the reference point since $z_i^* = \max(f_i(x)|x \in \Omega)$ for each $i = 1,...,m$. For each Pareto optimal point x* there exists a weight vector λ such that x* is the optimal solution of g^{te}. Consequently, one is able to obtain different Pareto optimal solutions by changing the weight vector. One weakness with this approach is that its aggregation function is not smooth for a continuous MOP. However, it can be used in the EA framework proposed since the algorithm does not need to compute the derivative of the aggregation function [14].

III. Boundary Intersection (BI) Approach:

Several recent MOP decomposition methods such as Normal-Boundary Intersection Method and Normalized Normal Constraint Method [18] can be classified as the BI approaches. They were designed for a continuous MOP. The PF of a continuous MOP is part of the most top right boundary of its attainable objective set under some conditions. Geometrically, these BI approaches aim to find intersection points of the most top boundary and a set of lines. If these lines are evenly distributed, it can expected that the resultant intersection points provide a good approximation to the whole PF. These approaches are able to deal with nonconcave PFs. In this work, a set of lines emanating from the reference point are used. As a result, the following scalar optimization subp roblem is considered [14]:

$$\begin{aligned} &\text{minimize } g^{bi}\left(x|\lambda, z^*\right) = d \\ &\text{subject to } z^* - F(x) = d.\lambda \\ &x \in \Omega \end{aligned} \tag{5.7}$$

As shown in Figure 5.17, the constraint $z^* - F(x) = d.\lambda$ guarantees that $F(x)$ is always in line L, the line with direction λ and passing through z^*. The goal is to push $F(x)$ as high as possible so that it reaches the boundary of the attainable objective set. One of the drawbacks of the above approach is that it has to handle the equality constraint. To cope with the constraint handling problem, using a penalty factor can be considered as a good method.

One of the drawbacks of the above approach is that it has to handle the equality constraint. Using a penalty method to deal with the constraint is a good method for it. It is given in Equation 5.8.

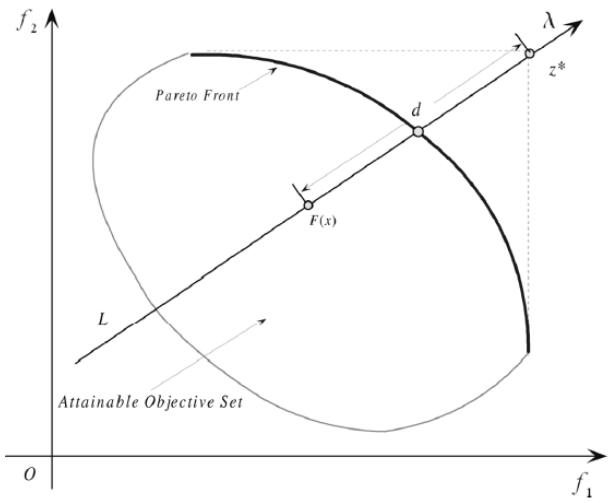

Figure 5.17: Illustration of boundary intersection approach [14]

$$\text{minimize } g^{bi}\left(x\middle|\lambda, z^*\right) = d_1 + \theta d_2$$

$$\text{subject to } x \in \Omega \text{ where;} \tag{5.8}$$

$$d_1 = \frac{\left\|(z^* - F(x))^T . \lambda\right\|}{\|\lambda\|} \text{ and } d_2 = \left\|F(x) - (z^* - d_1 . \lambda\right\|$$

$\theta > 0$ is a previously set penalty parameter. If y is the projection of $F(x)$ on the line L, as shown in Figure 5.18, d_1 will be the distance between z^* and y. d_2 is the distance between $F(x)$ and L. If θ is set appropriately, the solutions to (5.7) and (5.8) should be very close. Hereafter, this method is called the penalty-based boundary intersection (PBI) approach [14].

The advantages of the PBI approach (or general BI approaches) comparing to the Tchebycheff approach are as follows:

• In the case of more than two objectives, let both the PBI approach and the Tchebycheff approach use the same set of evenly distributed weight vectors, the resultant optimal solutions in the PBI should be much more uniformly distributed

than those obtained by the Tchebycheff approach, particularly when the number of weight vectors is not large.

• If x dominates y, it is still possible that $g^{te}(x|\lambda, z^*) = g^{te}(y|\lambda, z^*)$, while it is rare for g^{bip} and other BI aggregation functions [14].

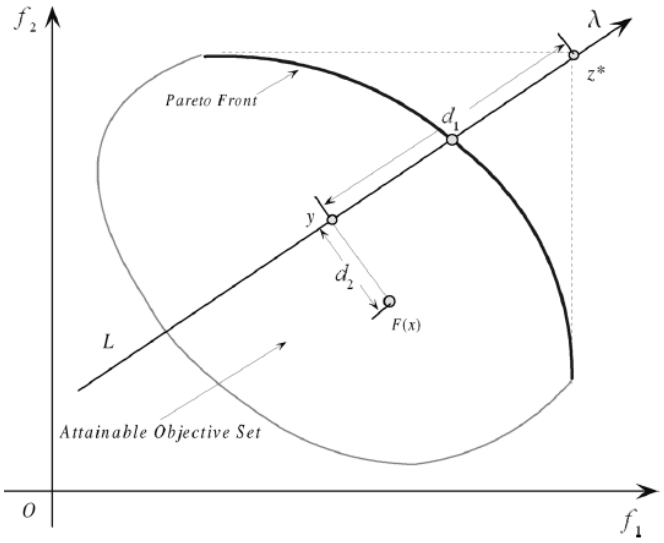

Figure 5.18. Illustration of penalty-based boundary intersection approach [14]

However, these benefits of course have a price which is that, one has to set the value of the penalty factor. It is well-known that a too large or too small penalty factor will even decrease the quality of the method [14]. It has also been experimented that best penalty factor is problem dependent.

The above approaches can be used to decompose the approximation of the PF into a number of scalar optimization problems. A reasonably large number of evenly distributed weight vectors usually leads to a set of Pareto optimal vectors, which may not be evenly spread but could approximate the PF very well [14].

There are also other decomposition approaches in the literature which can be used in EA algorithms. Since the main goal is to study the feasibility and efficiency of the algorithm framework the above three decomposition approaches were considered [14].

For the reasons given above, Weighted-Sum approach has not even been performed and compared with the other methods. However, lots of effort have been shown on the qualities of Tchebycheff approach and Penalty-Based Boundary Intersection Point approach.

The tests have been realized with two objectives of folded cascode analog amplifier, 100 population size, 40 niche and 100 generations. Tchebycheff approach has been implemented with both local and global normalization. The results of TE (Tchebycheff Approach) and PBI (Penalty-Based Boundary Intersection Point Approach) have also been compared with NSGAII algorithm.

Schotts metric has been used to compare the quality of the pareto front obtained by different theta (θ) penalty parameters. The θ values have been searched in a space of 0-10. For the gain-gbw trade-off the best θ has been found to be 3,2. In Figure 5.19, the good result range of theta for gain-gbw 2 objective problem has been shown.

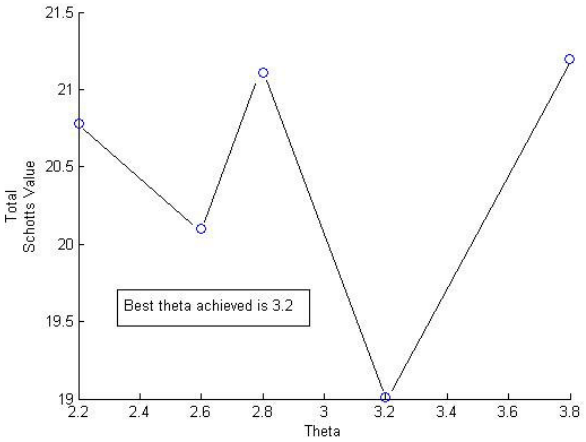

Figure 5.19. Theta Optimization Tests for Gain-GBW Problem

For $\theta = 3,2$ the PBI decomposition method has been implemented. The algorithm has also been run for TE approach with local and global normalization. NSGAII has also been performed. To make the comparison fair same number of population size and generations has been chosen. Moreover, to equalise the initial values of the individuals seed numbers of the random number generator has been set to a same value. The results are as seen below:

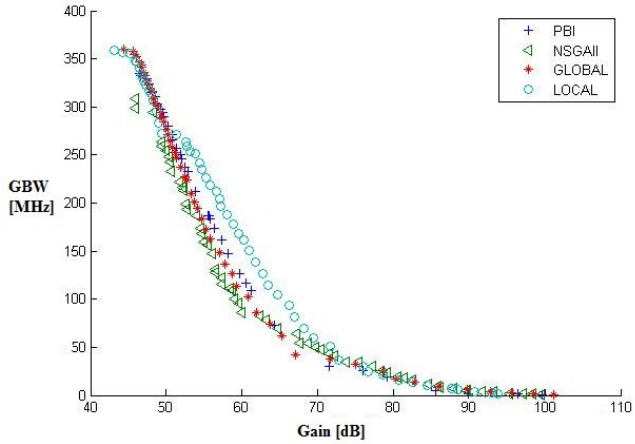

Figure 5.20. Comparisons of PBI with different TE methods for Gain-GBW Pareto Front

Figure 5.20 shows that the TE method with local normalization performs the best in terms of dominance and range of the pareto optimal solutions. TE with global normalization performs as good as NSGAII method; however PBI method has a smaller range than the other ones, also the distribution quality it has seems to be worse than the other methods.

Same trials have been realized on gbw-area trade-off. $\theta = 4,4$ seems to be the best option for this two objective optimization problem. It should be noted that best theta value is changing from problem to problem which means extra computational effort needs in order to alter and find the best theta for each problem.

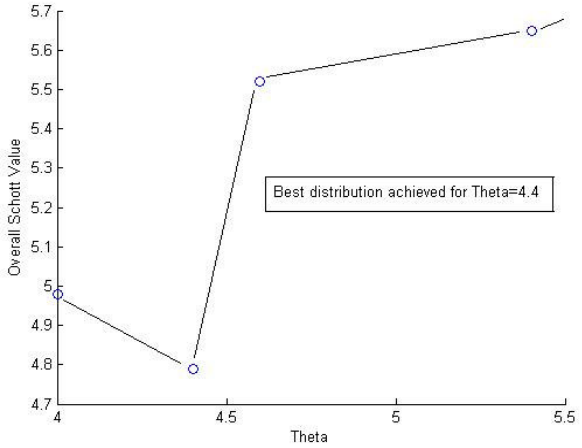

Figure 5.21. Theta Optimization Tests for GBW-Area Problem

PBI approach with $\theta = 4,4$ has again been compared with TE approach and NSGAII algorithm. The results show that TE method performs almost as good as NSGAII. For the PBI approach with the best theta value obtained the dominance seems to best but not far away from other methods. The critical point is that, PBI method has a small range in the pareto front also with a bad distribution.

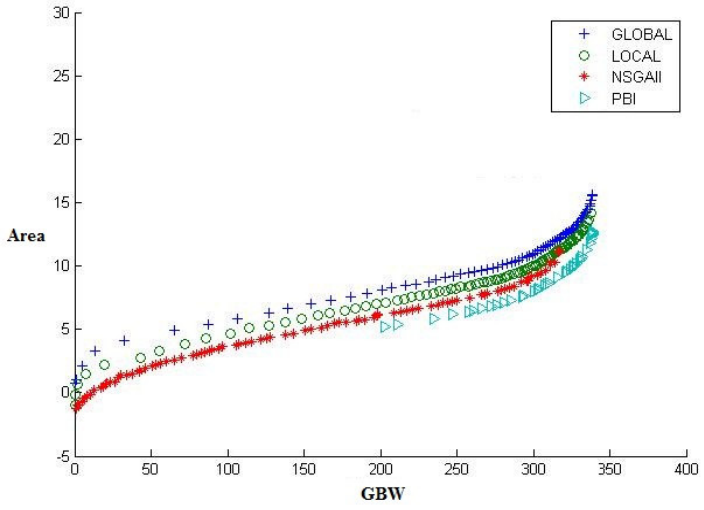

Figure 5.22. Comparisons of PBI with different TE methods for GBW-Area Pareto Front

Some decomposition methods which convert a Pareto Front to scalar optimization problems have been performed and compared in terms of distribution quality, range of the objective functions and dominance.

Weighted-Sum approach is not recommended for the optimization problems with very different ranged objective functions (For example, phase margin can vary between 0 to 180 degrees since power can just change from 0 to 6,9 mW). PBI approach has the disadvantage of obtaining a penalty factor which changes for every optimization problem. It is hard to obtain this value. Also, it has been observed that, the range and distribution of the functions to be optimized are not as good as other methods. TE approach seems like the best method for MOEA-D algorithm. It has no range, distribution or dominance problem, and no parameters need to be tuned prior to run.

As a result Tchebycheff approach has been chosen for the decomposition method of the main algorithm implemented.

5.2.4. Enhancing the Search Ability with the Use of DE

DE is without doubt, a very powerful search engine for single objective optimization. But when it comes to multiobjective problems, it seems to converge very fast to the vicinity of the true PF, but somtimes it may have some problems to actually reach it. What has been studied so far shows that for F scaling values between 0.5 and 1 is more close to overcome this problem [19].

DE mutation used, DE/best/1/bin [20], is as follows:

$$y' = x^{i}(t) + F(x^{r1}(t) - x^{r2}(t)) \tag{5.9}$$

where the indices $r1$ and $r2$ are randomly chosen and mutually different, and also different from the current index i. $F \in (0,1]$ is the scaling factor that controls the amplification of the differential variation $x^{r1}(t) - x^{r2}(t)$.

An extended version of the DE mutation above has also been performed and compared with the other DE mutation method. This mutation equation uses the best solution as an extra parameter:

$$y' = x^i(t) + F(x_{best}(t) - x^i(t)) + F(x^{r1}(t) - x^{r2}(t)) \tag{5.10}$$

Besides two different equations given, also a novel method randomizing the scaling factor (instead of setting it to a value between 0 and 1) has been used. In this work, a Gaussian distributed random scaling factor with mean value μ and variance σ has been used.

$$F_{i,k} = norm(\mu, \sigma) \ , \ i = 1,..,N \ \text{ and } \ k = 1,..,n \tag{5.11}$$

For each variable in the search space, scaling factor $F_{i,k}$ of each differential variation $x^{r1}(t) - x^{r2}(t)$ is different. $\hat{F}$ is continously and randomly generated in each iteration. So Equation 5.9 can be rewritten as:

$$y' = x^i(t) + \hat{F}(x^{r1}(t) - x^{r2}(t)) \tag{5.12}$$

Two different equations and two different scaling factor (F) determinations result in four different variations for the tests. In these tests, the parameter δ (it is the parameter which determines the probability that parent solutions are selected from the neighborhood) is set to 0,9. A 5^{th} test with δ=1, which means parent solutions will always be selected from the neighborhood, has also been performed. At last, the tests with the polynomial mutation, which has been used in the first version MOEAD [14], has also been performed. All these six type of methods, which have been mentioned above, are as given below in Table 5.9.

Table 5.9. Different Techniques to Find the Best DE Method

	Equation for Mutation	**Scaling factor selection**	**δ**
DE1	Equation 1	F=0.5	1
DE2.1	Equation 1	F=0.5	0.9
DE2.2	Equation 1	F=norm (μ,σ)	0.9
DE3.1	Equation 2	F=0.5	0.9
DE3.2	Equation 2	F=norm (μ,σ)	0.9
Polynomial Mutation	-	-	-

For the test conditions $popsize = 50$, the maximum number of solutions replaced by a child solution $n_r = 0.1 * niche$ and $niche = 20$, 30 tests were performed to generate the true PF for the test problem. 10 tests of each method (DE1, DE2.1 etc.) has been run for 100 iterations. For the same number of iterations, population size and probablity of polynomial mutation and with a SBX distribution index set to 20, NSGAII was also run for the same test problem.

The benchmark test problem is an extended version of MOP-C3/Viennet4 [21] problem. The original version is for 3 objectives and with similar rules a 4^{th} objective function has been added.

$$
\begin{aligned}
F &= (f_1(x,y), f_2(x,y), f_3(x,y), f_4(x,y)) \\
f_1(x,y) &= \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3, \\
f_2(x,y) &= \frac{(x+y-3)^2}{175} + \frac{(2y-x)^2}{17} - 13, \\
f_3(x,y) &= \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} + 15, \\
f_4(x,y) &= \frac{(y-x-3)^2}{16} + \frac{(x-y+2)^2}{8} + 5
\end{aligned}
\tag{5.13}
$$

The performance metric IGD has been used to compare the methods. The results for 10 tests are given in Table 5.10.

Table 5.10. IGD Values of Different DE tests for a 3-objective Benchmark Problem

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10		Average
Poly. Mut.	0.0028	0.0017	0.0021	0.0034	0.0045	0.0022	0.0038	0.0031	0.0025	0.0024		**0.0029**
DE1	0.0017	0.0018	0.0029	0.0013	0.0029	0.0020	0.0017	0.0039	0.0015	0.0017		**0.0021**
DE2.1	0.0017	0.0018	0.0016	0.0024	0.0017	0.0017	0.0014	0.0014	0.0015	0.0015		**0.0017**
DE2.2	0.0017	0.0018	0.0012	0.0020	0.0018	0.0021	0.0013	0.0017	0.0014	0.0017	(best)	**0.0016**
DE3.1	0.0018	0.0011	0.0036	0.0017	0.0011	0.0021	0.0022	0.0016	0.0016	0.0020		**0.0019**
DE3.2	0.0018	0.0012	0.0031	0.0020	0.0017	0.0019	0.0018	0.0017	0.0015	0.0017		**0.0018**
NSGAII	0.0053	0.0122	0.0089	0.0035	0.0089	0.0056	0.0081	0.0068	0.0070	0.0095		**0.0076**

From the results, it can be seen that, DE mutation enhanced the search ability comparing to the polynomial mutation which was used in the first version of MOEAD [14]. Also, including a δ parameter (the probability that parent solutions are selected from the neighborhood) with a value of 0,9 resulted much better than the case with δ=1. NSGAII has easily been beaten by any of the MOEAD-DE methods. For the selection of the DE mutation equation, the first equation (Equation 1) seems to be working than the second one (Equation 2).

According to the test results, DE2.2 has been chosen. The reason for that choice is that it has the best IGD value. This method is using Equation 5.9 for the DE mutation and it is also randomizing the scaling factor method. The true PF obtained by DE2.2 has 1427 points which means just 73 (since 50*30 = 1500) points have been dominated by all other methods (For all 30 tests per method). This is a quite good point to note, especially in terms of the dominance quality of the optimization algorithm. The projection of the true PF obtained by DE2.2 on the 2-D space is as given in Figure 5.23. The Pareto Set points are the points from the True Pareto of the related method (The non-dominated points among all of the tests)

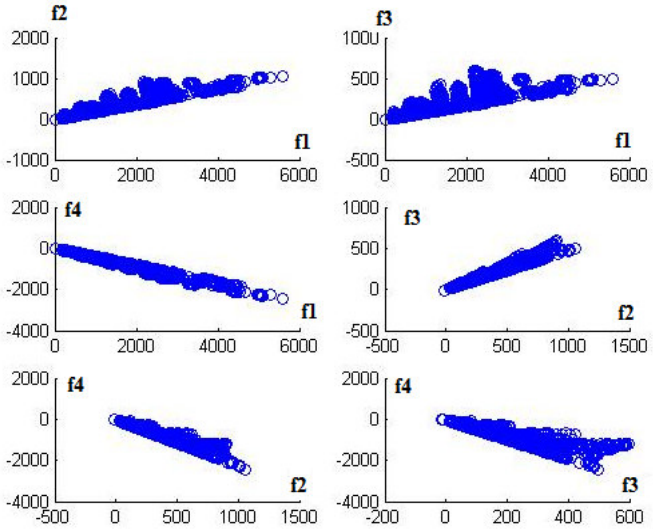

Figure.5.23. Pareto Front's 2-D projections obtained by DE2.2 for the test problem

Most important comparison is on the scaling factors, since it is a novel method proposed. The results show that, the method randomizing the scaling factor performs better than the one setting the scaling factor to a constant value. Figure 5.24 shows the effect of randomizing *F*. It can be seen that a cloud of potential points centered around the mutant vector could be generated.

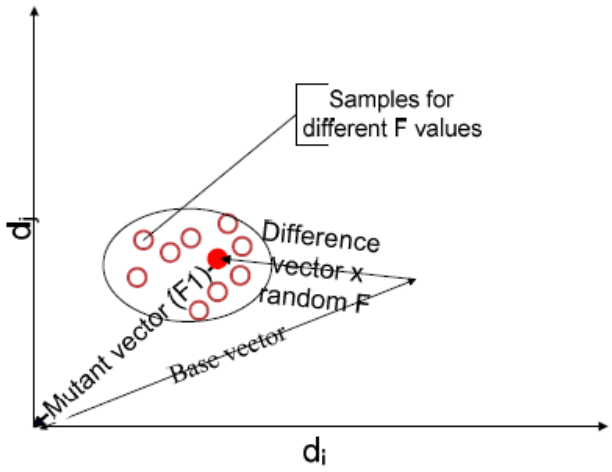

Figure 5.24. Illustration of Mutant vectors obtained by the random-scale operator

The random amplification induces two advantages: (1) The algorithm has a lower probability of providing premature solutions because of the reasonable diversity; (2) The vicinity of the mutant vector is investigated by the randomized amplification of the differential variation $x^{r1}(t)-x^{r2}(t)$. Even when stagnation appears, a new trial vector has fair chances of pointing at an even better location on the multimodal functional surface.

As a result DE mutation enchances the search quality for the multi-objective optimization. For the mutation function DE/best/1/bin [20] has been used. Also a novel method based on randomizing the scaling factor has been tried and successfull results have been obtained.

5.2.5. A New Replacement Mechanism

Besides DE mutation, to enhance the performance of the MOEA/D-DE framework, another work on population replacement has been realized. The goal here is to call for a balance between information sharing and diversity maintenance. In the method proposed, when the number of parent solutions that can be replaced by a high quality child solution exceeds the maximum number, the parent solutions are ranked and those that are closer to the child solution are replaced first.

Previous method for the reproduction was based on the replacement of each index of the neighborhood in the population. It was basically as follows:

For each index $j \in B(i)$, if $g^{te}\left(y' \middle| \lambda^j, z\right) \le g^{te}\left(x^j \middle| \lambda^j, z\right)$ then set $x^j = y'$ (5.14)

In the proposed method the replacement is realized under some conditions, instead of replacing the whole neighborhood. The new replacement mechanism first calculates $g\left(y \middle| \lambda^j, z\right)$ and $g\left(x^j \middle| \lambda^j, z\right)$ for each j in P.

Secondly, $c = 0$ is selected. In case $g\left(y \middle| \lambda^j, z\right) \le g\left(x^j \middle| \lambda^j, z\right)$, this *c* parameter is increased by one ($c = c + 1$) in order to control the number of the replaced solutions.

Thirdly, the comparison of c with n_r is realized. If $c \le n_r$ is true for each j with $g\left(y\middle|\lambda^j, z\right) \le g\left(x^j\middle|\lambda^j, z\right)$, all the improved solutions (the children) are replaced ($x^j = y$) with the old ones. If $c > n_r$, for each j with $g\left(y\middle|\lambda^j, z\right) \le g\left(x^j\middle|\lambda^j, z\right)$, then euclidean distances between $f(y)$ and $f(x^j)$ are calculated and and ranked. After the ranking, n_r solutions with the smallest distances are chosen for the replacement.

In MOEAs, the replacement mechanism is intended to improve the quality (in terms of domination) of the population and maintain the diversity. Although in decomposition-based methods, search in different directions according to different weight vectors can "naturally" help the diversity, diversity maintenance is also affected by the replacement mechanism. A high quality child solution may replace most of the current solutions to its neighboring sub-problems. Consequently, diversity decreases significantly. In MOEA/D [14], the maximum number of solutions that can be replaced by a child solution is the size of the neighborhood, *T*, whose disadvantage is shown in [22]. MOEA/D-DE improves the replacement mechanism by adding a bound n_r, which is much smaller than *T*. A high quality child solution can replace n_r current solutions at most, which helps the diversity maintenance.

However, setting the value of n_r is not a trivial problem. n_r controls the balance of information sharing and diversity maintenance. If n_r is large, the information of a good solution can be shared by more current solutions, but the risk of diversity reduction is higher. In contrast, if n_r is small, the information can be shared by less solutions, but the diversity is maintained.

An empirical rule is proposed by setting $T = 0.1N$, $n_r = 0.01N$, and the n_r current solutions which will be replaced by a high quality child solution are randomly chosen if the bound is exceeded. Generally, this rule is reasonable. Nevertheless, both conditions, *c* (the number of current solutions with $g\left(y\middle|\lambda^j, z\right) \le g\left(x^j\middle|\lambda^j, z\right)$ much smaller than $0.01N$ and *c* much larger than $0.01N$ may appear in the evolution process. When *c* is much larger than

$0.01N$, randomly selecting $0.01N$ individuals to be replaced may not always be a good solution.

It can be seen that n_r is approximately 10% of T, that is, for one segment with 10 points that can be replaced by a high quality child solution, only one of them can be updated. Such n_r is small to share the good information. On the other hand, n_r cannot be larger to keep the diversity. Hence, selecting which points should be replaced in order to make the sharing more effectively is a significant problem. It can be argued that in the objective space, points that are near to the newly generated high quality child solution can be benefit more compared with the ones that have longer distance from it if the replacement is performed. The reason is that for neighbors which have similar fitness landscapes, their optimal solutions should be close to each other in the decision space. This is the principle of MOEA/D which can be seen as "neighbor's neighbor". In Figure 5.25, the bottom point with a coordinate (1,1) is the high quality child solution, and can replace all of the 6 points with '*' symbol. If only one can be selected, then the two points in the circle will benefit more than the other 4 points if the schema of the bottom point is used.

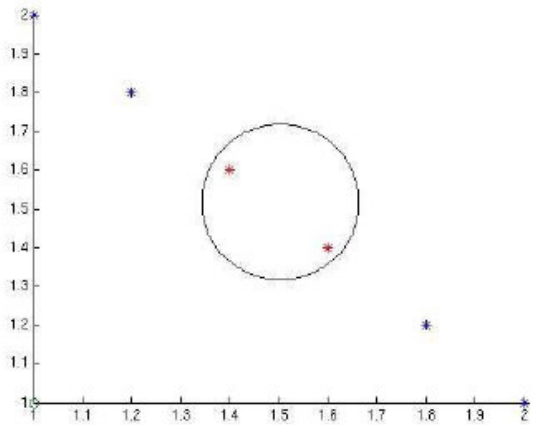

Figure 5.25. Illustration of the replacement mechanism

Therefore, the mechanism is that when a high quality child solution, which has the ability to replace most of the current solutions in T, appears, instead of randomly choosing n_r current solutions, their distances are ranked to the high quality child solution in the objective space and the n_r solutions are replaces with the smallest distances.

Table 5.11. Effects of different n_r values on performed DE techniques

	Test 1	Test 2	Test 3	Test 4		Ave. IGD
DE1						
nr=0.05*niche	0.0060	0.0061	0.0058	0.0064		0.0061
nr=0.10*niche	0.0051	0.0055	0.0055	0.0059		0.0055
nr=0.15*niche	0.0068	0.0080	0.0059	0.0064		0.0068
nr=0.20*niche	0.0078	0.0060	0.0058	0.0067		0.0066
					Average:	**0.0063**
DE2.1						
nr=0.05*niche	0.0062	0.0066	0.0043	0.0056		0.0057
nr=0.10*niche	0.0058	0.0066	0.0047	0.0064		0.0059
nr=0.15*niche	0.0059	0.0052	0.0050	0.0061		0.0056
nr=0.20*niche	0.0076	0.0068	0.0045	0.0054		0.0061
					Average:	**0.0058**
DE2.2						
nr=0.05*niche	0.0048	0.0045	0.0056	0.0056		0.0051
nr=0.10*niche	0.0043	0.0050	0.0070	0.0057		0.0055
nr=0.15*niche	0.0064	0.0064	0.0050	0.0042		0.0055
nr=0.20*niche	0.0046	0.0057	0.0053	0.0046		0.0050
					Average:	**0.0053**
DE3.1						
nr=0.05*niche	0.0072	0.0042	0.0060	0.0053		0.0057
nr=0.10*niche	0.0097	0.0054	0.0059	0.0062		0.0068
nr=0.15*niche	0.0033	0.0048	0.0056	0.0109		0.0062
nr=0.20*niche	0.0052	0.0057	0.0063	0.0056		0.0057
					Average:	**0.0061**
DE3.2						
nr=0.05*niche	0.0052	0.0061	0.0051	0.0050		0.0054
nr=0.10*niche	0.0061	0.0040	0.0046	0.0047		0.0049
nr=0.15*niche	0.0054	0.0078	0.0043	0.0055		0.0057
nr=0.20*niche	0.0055	0.0047	0.0061	0.0046		0.0052
					Average:	**0.0053**
NSGAII	0.0253	0.0234	0.0221	0.0237		**0.0236**

The critical parameter for the quality of the proposed replacement method is clearly the value of n_r. To see the behavior of the change in n_r and to be able to see more tests on DE methods tried, the extended version of MOP-C3/Viennet4 [21] benchmark problem has been run again with different seed numbers (setting seed number makes the Comparisons fair by equalising the random numbers generated) than previous ones. 10 test optimizations were done for each DE method and different n_r to create the true PF's. 4 tests for each conditions were run to compare the results. The results are shown above in Table 5.11.

As it can be seen from the tests, DE2.2 looks like having the best results. DE3.2 can compete with DE2.2. The results are compatable with the previous tests and they show that NSGAII has been beaten very easily again.

About different n_r values, it looks like there is no big difference. Sometimes a small value of n_r can give good results and sometimes a large value. For the DE methods performed $n_r = 0.05 * niche$, $n_r = 0.15 * niche$ and $n_r = 0.20 * niche$ has the best results just for 1 methods. However $n_r = 0.10 * niche$ is the best result for 2 cases.

As a result $n_r = 0.10 * niche$ and DE2.2 methods have been chosen as the main parameters and mutation methods. The following tests are using DE2.2 mutation as the search engine and $n_r = 0.10 * niche$ parameter.

So far, it as been mentioned that, DE is a good method to enhance the search ability comparing to the previous mutation. Also with an emprical method, $n_r = 0.10 * niche$ has been decided. To see the effects of randomizing F scaling factor and adding new replacement rules, MOEA/D-DE (OD) which is using DE2.1 mutation (F has already been set to 0.5, no new replacement methods), MOEA/D-DE with new replacement rules (RD), MOEA/D-DE with stochastic scaling factor (FD) and MOEA/D-DE with both new replacement rules and stochastic scaling factor (FRD) have been compared. The test problem instances are UF1 to UF10 in CEC 2009 competition (2-3 objectives) [23] and a

real world problem, sizing of folded-cascode amplifier (4 objectives) are performed on the test problem instances UF1 to UF10 in CEC 2009 competition (2-3 objectives) [23] and a real world problem, sizing of folded-cascode amplifier (4 objectives). The performance metric is again IGD.

The test problems include benchmark problems and a four objective analog sizing problem. The benchmark problems are UF1 to UF10 in [23]. The multiobjective analog sizing is optimization of a folded-cascode amplifier, where the DC gain, GBW, phase margin and power are the 4 objectives. In the analog sizing problem, there is no analytical formulation of the optimization goals. They are based on the SPICE simulation. There are 11 design variables, 5 of which For UF1 to UF10 in [23], the number of decision variables is 30. For the analog sizing problem, the number of design variables is 11. The number of sub-problems (population size), *N*, is 300 for 2 objective problems, 500 for three objective problems and 148 for the analog sizing problem (though 4 objectives, considering the computational effort, *N* is reduced to 148) . *T* is set to 0.1*N*, $n_r = 0.01 * N$, δ is set to 0.9. In DE operators, CR is set to 1, F is a Gaussian distributed vector with a mean of 0.5 and a variance of 0.15. In GA operators, η and p_m are the same as MOEA/D-DE. For benchmark problems, the algorithm stops after 1000 generations for 2 objective problems, and 1200 generations for 3 objective problems. For the analog sizing problem, the algorithm stops after 200 iterations.

Table 5.12. The IGD statistics based on the average of 20 runs of different methods based on the New Replacement Method and DE

Tests	FRD	FD	RD	OD
UF1	0.0096	0.0064	0.0025	0.0027
UF2	0.0084	0.0072	0.0094	0.0098
UF3	0.0472	0.0311	0.0093	0.0105
UF4	0.0592	0.0788	0.0881	0.0858
UF5	0.5577	0.7650	0.8476	0.9247
UF6	0.1795	0.2726	0.2381	0.2665
UF7	0.0056	0.0063	0.0054	0.0032
UF8	0.0660	0.0611	0.0569	0.0562
UF9	0.1304	0.1299	0.1170	0.1501
UF10	0.4035	0.4370	0.4119	0.4781
Analog	9.4572	9.5344	9.5199	9.6079

For UF1 to UF10 in [23], the set $P^* \in PF$ is available (P^* is the true Pareto Front to converge). For the analog sizing problem, 30 runs are first performed using each method, whose results are combined to approximate the P^* the True PF. Table 5.11 shows the mean values of IGD results for each problem in 20 runs.

Here are some observations of the results. For each problem, the different methods are ranked according to the IGD values and Table 5.13 and Table 5.14 are obtained.

Table 5.13. Ranking of the IGD values of different methods based on the New Replacement Method and DE

Tests	FRD	FD	RD	OD
UF1	Rank 4	Rank 3	Rank 1	Rank 2
UF2	Rank 2	Rank 1	Rank 3	Rank 4
UF3	Rank 4	Rank 3	Rank 1	Rank 2
UF4	Rank 1	Rank 2	Rank 4	Rank 3
UF5	Rank 1	Rank 2	Rank 3	Rank 4
UF6	Rank 1	Rank 4	Rank 2	Rank 3
UF7	Rank 3	Rank 4	Rank 2	Rank 1
UF8	Rank 4	Rank 3	Rank 2	Rank 1
UF9	Rank 3	Rank 2	Rank 1	Rank 4
UF10	Rank 1	Rank 3	Rank 2	Rank 4
Analog	Rank 1	Rank 2	Rank 3	Rank 4

Table 5.14. Statistics of the ranking of different methods based on the New Replacement Method and DE

Methods	Rank 1	Rank 2	Rank 3	Rank 4
FRD	5	1	2	3
FD	1	4	4	2
RD	3	4	3	1
OD	2	2	2	5

It can be seen that the improvement of the new replacement mechanism is obvious. In 7 cases out of 11, the RD (MOEA/D-DE with new replacement) method ranks 1 or 2, FRD (RD plus random scaling factor) method has 6 cases with rank 1 or 2, FD (MOEA/D-

DE with random-scale F) have 5 cases with rank 1 or 2 and the original MOEA/D-DE has 4 cases. If only considering the rank 1 column, it can be seen that RD and FRD have more distinct advantages. If only adding a random scaling factor, slight improvements have been observed in high rank region (rank 1 or 2). But it is obvious that the FD method ranks 3 in 4 cases and ranks 4 in 2 case, while the original MOEA/D-DE (OD) ranks 3 in 2 cases, and ranks 4 in 5 cases. When the two mechanisms are combined together, it can be seen that FRD have 5 cases with rank 1, which has distinct advantage compared with other methods. On the other hand, it has 3 cases with rank 4. Therefore, it can be concluded as FRD is a method which can obtain very good result, and RD method is more stable.

Best results (which are the tests with the smallest IGD values from all tests) of some of the benchmark problems for FRD method are as seen in Figure 5.26. The Pareto Fronts for all benchmark problems have been plotted.

NSGA-II is also implemented for the analog sizing problem using the same population size, η and p_m. The distribution index in SBX is set to 20. The average IGD value is 15.8656, which is much larger than MOEA/D-based methods.

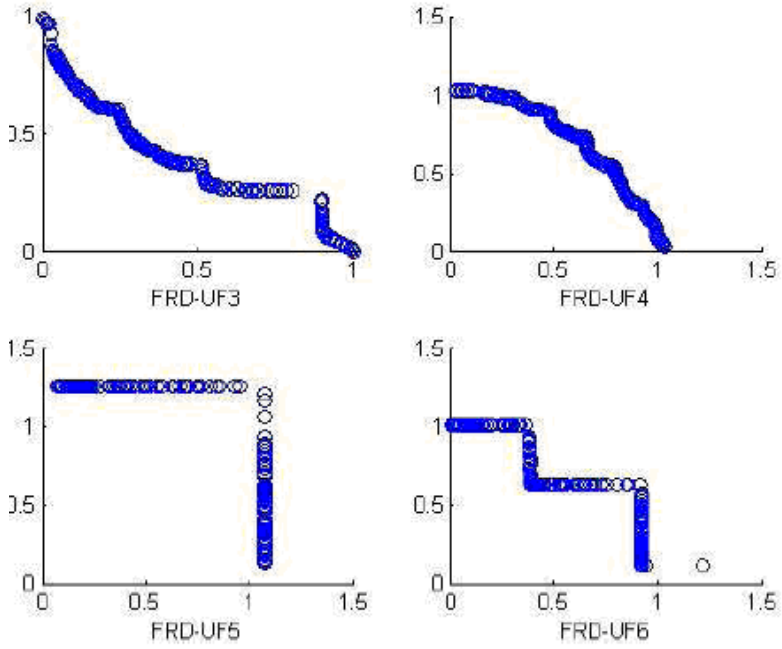

Figure 5.26. PF for some of the benchmark problems with smallest IGD by FRD method

5.3. Enhanced MOEA/D-DE Algorithm

In this Section, the Enhanced MOEA/D-DE Algorithm will be given with the Multi-objective Optimization Problem (MOP) definition and with the flow chart of the algorithm.

5.3.1. MOP for the MOEA/D-DE

A multiobjective optimization problem can be stated as follows:

$$\min\{f_1(x),...,f_m(x)\}, x \in \Omega \tag{5.16}$$

where $x=(x_1,...,x_n)$ is the decision variable vector and $f_i(x)$ are the objective functions. Ω is the decision space. A solution x is said to dominate solution y if and only if $f_i(x) \le f_i(y)$ for every $i \in \{1,...,m\}$ and $f_j(x) < f_j(y)$ for at least one index $j \in \{1,...,m\}$. A point $x^* \in \Omega$ is Pareto optimal to (1) if there is no point $x \in \Omega$ such that $f(x)$ dominates $f(x^*)$. $f(x^*)$ is Pareto-optimal objective vector. The set of all the Pareto-optimal points is called the Pareto Set (PS). The set of all the Pareto-optimal objective vectors is called the Pareto Front (PF).

5.3.2. The Working Principles of the Algorithm

For the Tchebycheff approach, the scalar function is as follows:

$$g^{te}\left(x\middle|\lambda, z^*\right) = \max_{1\le i\le m}\{\lambda_i \left| f_i(x) - z_i^* \right|\} \tag{5.17}$$

where $\lambda=\left(\lambda_1,..,\lambda_m\right)$ is a weight vector and $\sum_{i=1}^{m}\lambda_i = 1$. Ω is the solution space and $z^*=(z_1^*,...,z_m^*)$ is the reference point. If N is reasonably large and $\lambda^1,...\lambda^N$ are properly

selected, the optimal solutions to those scalar functions will provide a good approximation to the PS/PF. The major components in MOEA/D are its neighborhood concept, and its population replacement mechanism. The enhanced MOEA/D-DE, proposed works as follows:

Inputs are:

i. A MOP
ii. A stopping criterion
iii. N: the number of sub-problems
iv. T: the neighborhood size
v. δ : the probability that parent solutions are selected from the neighborhood
vi. n_r : the maximum number of solutions replaced by a child solution
vii. CR: crossover rate in DE
viii. μ,σ : the mean and variance of the scaling factor $\hat{F}$ in the DE mutation
ix. p_m : the probability to perform polynomial mutation
x. λ : weight vector

Outputs are:

i. Approximation to the PF
ii. Approximation to the PS

First of all, an initialization step exists. In this step, the external population is set to zero. Later, the Euclidean distances between weight vectors are calculated in order to find the T closest weight vectors to each weight vector and the neighborhood $B(i)=\{i_1,...,i_T\}$ is set

for the *T* closest weight vectors λ^i. After that, an initial population, which is the set of solutions, is randomly generated and the objective functions are evaluated for these solution individuals. The minimum and maximum values of $z=(z_1,...,z_m)^T$ for each objective function evaluated before.

Secondly, the algorithm starts a loop of *N* turns to realize the updates. First of all the neighborhood set is selected with respect to the δ value. Later on, randomly selected two indexes of *B(i)* are used to generate a new solution by using the genetic operators. Here, the DE mutation is used as the main search engine. Later an improvement is applied on the solution. If the solution is out of boundary then it is regenerated. The improved, new solution is used to calculate the objective functions in order to update the *z* values. Later, with respect to n_r if $g^{te}\left(y^{'}\left|\lambda^j,z\right.\right)\le g^{te}\left(x^j\left|\lambda^j,z\right.\right)$ is satisfied (*y'* is the new solution), the solution set and the fitness values are updated. It should be noted that if the number of the solutions to be replaced exceeds the n_r, then n_r number of solutions are ranked and the parents with the smallest Euclidean distance to the solutions are updated.

At the last step if the stopping condition is satisfied (it may be max number of iterations) the algorithm stops and outputs the *EP*. Otherwise the update loop goes on.

The general framework of the proposed MOEA/D-DE method works as given in Figure 5.27.

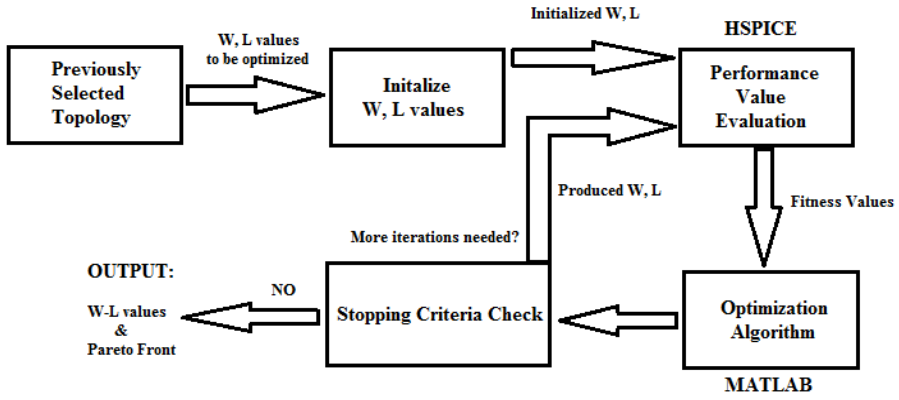

Figure 5.27. The framework of the MOEA/D-DE Optimization Algorithm

5.3.3. Conclusions

First of all, W and L values for the selected topology are randomly generated. These values are simulated by using HSPICE A. 2007-2009 simulator. The outputs are the evaluated performance values of the analog circuit. Because of the general behavior of the algorithm, all of the objectives are supposed to be minimized. This is satisfied by using a minus sign for the fitness values which are expected be maximized. For example, the "area" is a function to be minimized, so it's value is directly used as obtained from HSpice; however, the function "gain" has to be maximixed so the return value of the HSpice is multiplied by -1 and the problem turns into a minimization problem. According to that fitness values as mentioned before the algorithm generates new W, L values to enhance the performance of the circuit. This is done by the search algorithm of the MOAE/D-DE which was coded on MATLAB. New W, L values are then simulated on HSpice and new fitness functions are obtained. According to these fitness values, the algorithm generates new W, L values and so on. This loop goes until the set value for the generation (iteration) number is reached. The parameters (variables) to be optimized are not only supposed to be W or L, they can also be capacitor values (C), inductor values (L), or current source values (ib), etc. All the runs were realized on a Pentium Dual Core CPU - T4300 @ 2.10 GHz.

6. ON-LINE INTERPOLATION OPERATING POINT DRIVEN METHOD

There are two methods for chosing the variables of optimization in analog design automation. First one is direct optimization of the dimensions, W and L. The second type of methods are based on optimizing the DC values of the nodes (voltages) and the branches (currents) in an analog circuit. These algorithms have DC root solving mechanisms. DC root solving is basically guessing the W of the transistors by using DC optimization variables. This is the only way to evaluate the analog circuit and let the optimization go on. A novel method for DC root solving has been proposed, and the results show that proposed method is much more sufficient than the methods in the literature in terms of accuracy, speed, use of memory etc.

6.1 Introduction to OPD Based Methods

In recent years, analog design automation methodologies receive much attention in both literature and industrial applications. Besides the development of the cell-level analog sizing methods, yield-aware sizing, parasitic-aware sizing and hierarchical synthesis methods are developing in a high speed. On the other hand, fundamental analog sizing (cell-level optimization for analog ICs in nominal condition) remains a key problem. The reasons are that: (1) analog sizing methods considering other factors, e.g. hierarchical, also rely on fundamental cell-level optimization; (2) many advanced analog sizing methods need a good starting point generated by fundamental cell-level optimization. This work focuses on fundamental analog cell sizing.

Fundamental analog cell sizing methods can be classified to two main categories: width and length (W/L)-based methods [17,24,25] and operating-point driven (OPD)-based methods [26-28]. The former method uses the width and length as the design variables and considers the analog sizing as a constrained optimization problem. The latter method, however, uses operating point as the design variables and the device dimensions (W) are determined out of it. OPD-based method is used much less than width and length-based

method in recent literatures. However, it has been reported that OPD-based methods have advantages [26-28], which can be summarized as follows. Firstly, it highly relieves the convergence problem of electrical simulators, as a DC consistent solution cannot be found for some W/L in the search space and these W/L are difficult to be pruned beforehand [26]. Secondly, device operating constraints (e.g. $v_{gs} > 0$, transistor in the saturation region) [29] are self-contained in the generation of the candidate solutions in the OPD-based methods, which can ensure the devices to operate in the intended region. These constraints are obvious to the designer, but not to the optimization algorithm. For W/L-based methods, explicitly measurements and enforcements need to be added, which increase the number of constraints and make the optimization problem more difficult [28]. Thirdly, the design variable W often has a large range in W/L-based method, which adds high pressure to the search algorithm, especially when the circuit is complex. On the other hand, W is calculated from device biases (voltages, currents) in OPD-based methods, which decreases the search space in another way (The search area for a W may change from less than 1um to hundreds of micrometers. DC points of the circuit have lower range to search. For example a voltage value in a 0,25um technology can change from 0 to 2.5 Volts which means it is easier to converge to the optimal value). Fourthly, OPD-based methods allow the designer to reason in terms of voltages and currents and relieve him from the burden of determining device sizes [26].

OPD-based methods have such advantages, but why they seldom appear in recent literatures? An important reason is that with the scaling down of devices, the models are becoming more and more complex. This makes the available DC root solving algorithms and the look-up-table-based methods face significant challenges on accuracy, efficiency and memory requirements. Also obtaining W by solving equations loose their accuracy, which leads to the difficulty to compute an acceptable W by L and device biases. For look-up-table (LUT) methods, the trade-off between accuracy, look-up time (the time to find the corresponding data in the LUT) and memory consumption is important. The experiments in the following parts of the work show that a LUT with acceptable accuracy need a large amount of data, which consumes long look-up time and large memory.

To address these problems, a new OPD method, called on-line interpolation operating-point driven (OIOPD), is proposed. In OIOPD, the width of a transistor is computed by the interpolation of the width-current curve with a determined set of length and voltage biases. The two $(W_{\min}, I_{\min})$ and $(W_{\max}, I_{\max})$. $W_{\min}$ and $W_{\max}$ are the two extreme allowed values of W for a technology) are first simulated on-line, then the request W is computed by SPLINE interpolation. OIOPD has 10 times improvement on accuracy, 300-1100 times improvement on efficiency compared with the available methods. In addition, OIOPD need not to tune the parameters, e.g. the size of the LUT, number of neurons in NN. Also there is no need for a memory.

6.2 OPD Methods

6.2.1. Review of the OPD Methods

Figure 6.1 shows the flow of the OPD method in analog sizing. In each iteration, for each transistor of each candidate design, the input is the length and device biases. As shown by Equation 6.1, the W is computed by $\{L, I_{DS}, V_{ds}, V_{gs}, V_{bs}\}$.

$$I_{DS} - I_{DS}\left(V_{ds}, V_{gs}, V_{bs}, W^{*}, L\right) = 0 \tag{6.1}$$

where W is the independent variable and the drain source current is a function of the drain-source, gate-source, bulk-source DC voltages, W and the L. The estimated W for all the transistors are collected and SPICE simulation using the L and the estimated W is done to obtain the performance of the candidate solution. The performances are sent to the optimization algorithm to start the next iteration.

It can be seen that the DC root solving is a critical problem in this flow. Whether the W can be computed with an acceptable accuracy determines the successfulness of the OPD-based analog sizing. The methods to obtain W from Equation 6.1 can be classified to three main categories.

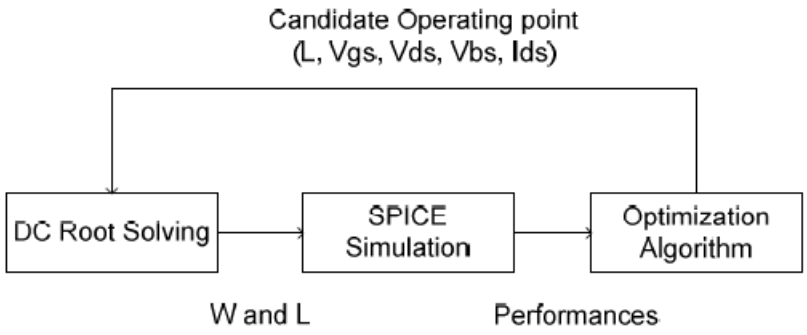

Figure 6.1. OPD based analog sizing

The first kind of method is to find W by directly solving Equation 6.1 [26,27]. First-order models can be computed easily, since an explicit equation for W exists. But it is too inaccurate. Hence, high-order models are necessary. Reference [26] suggests using the first-order solution as the starting point and then using SPICE-in the-loop method to solve the equation. It also suggests scaling the variables to linearize strongly non-linear functions, which transforms Equation 6.1 to Equation 6.2.

$$\log I_{DS} - \log I_{DS}\left(\log V, \log W^{*}, \log L\right) = 0 \qquad (6.2)$$

Because of using SPICE simulation in the equation (no analytical form), using iteration methods to solve the equation are appropriate. Newton-Raphson method [31] and Golden Section Search method [32] are tried to solve (6.2) and found that a good W is difficult to be obtain. The test results are given at the following parts.

Another method to obtain W is regression. First, a set of training data are generated, and a regression model is constructed to predict W by the inputs $L, I_{DS}, V_{ds}, V_{gs}, V_{bs}$. Neural network (NN) [30] is often considered as a powerful regressor. However, in the experiments realized, it was found that NN is difficult to achieve high accuracy and generality at the same time.

LUT is another kind of method to find W [33]. Many points are sampled in the $\left\{L, I_{DS}, V_{ds}, V_{gs}, V_{bs}\right\}$ space and stored in an LUT. When using LUT to find W, a hierarchical

look-up method is used [28]. For example, the method first finds the nearest left and right points of the given V_{ds} and two candidate V_{ds} are selected. In the LUT, there exist some points with these two V_{ds}. The algorithm again finds the nearest left and right points of the given V_{gs} in the points with the two candidate V_{ds}. This process continues until the problem becomes a one-dimensional interpolation problem, e.g. the last variable is I_{DS}. The interpolation is realized by left and right side options of these 3 voltage values and the L. For all these 16 options drain-source current value is obtained and at last interpolation method is used to obtain the W. According to the experiments, it was found that the LUT method has relatively high accuracy when a large amount of samplings are used. But it costs a long look-up time and huge memory.

6.2.2. OIOPD Method

From the review of the above, three methods using modern technologies, LUT is better than the other two methods. It can be seen that interpolation is an effective way. In the LUT method, there need 4 approximations before a one-dimensional interpolation. This process looses accuracy since all of the approximations already have their own errors. To prevent this, LUT need to include a large number of samples, which cause longer look-up time and more memory. On the other hand, for each operating point, the $\{L, I_{DS}, V_{ds}, V_{gs}, V_{bs}\}$ are determined. Therefore, selecting one variable to do the one-dimensional interpolation with W under the condition of the fixed other 4 variables (need on-line simulations) can enhance the accuracy. The reason is that by on-line simulation method, the other 4 variables are accurate (%100 accuracy), not by approximation.

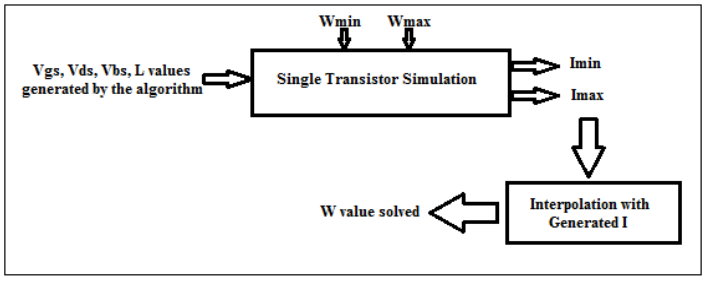

Figure 6.2. W Guessing Procedure for OIOPD Method

In OIOPD, I_{DS} is selected as the variable to do the interpolation with W, using the determined $\{L, V_{ds}, V_{gs}, V_{bs}\}$. First different samples of W are used in the technology allowed range to generate the corresponding I_{DS}. Then, the estimated W can be interpolated using the given I_{DS}. Another advantage of on-line simulation (a single NMOS and PMOS transistor is simulated for the given $L, V_{ds}, V_{gs}, V_{bs}$ values and for the limit values of *W*) is that the memory requirement problem is solved. The key problem is the necessary number of samples to achieve an accurate estimation and the comparison between the time spent to generate the samples and the look-up time in the LUT method.

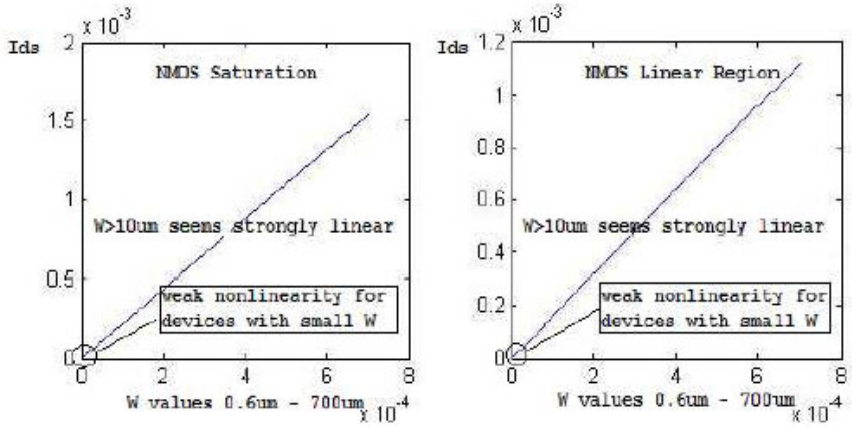

Figure 6.3. Typical case of *W*- I_{DS} curve

Figure 6.3 shows two typical cases of the curve in a 0.25um technology. It can be seen that the linearity between W and I_{DS} is strong in most part. For W smaller than 10um , the linearity decreases a little. Because of the generally strong linearity, it is reasonable to use few samples. The results of using 200 samples, 50 samples and 2 samples (the maximum and minimum width in a technology) have been compared. Some typical results are shown in Table 6.1. The tested W are uniformly distributed in the allowed range of each technology. From Table 6.1, it can be seen that the result of using 2 samples is comparable to, or even better than that of using 200 samples when the width is large than 10um. The reason is that the linearity is stronger in a global range compared with in a local range, as shown by Figure 6.3. This is also an important reason to describe why directly solving Equation 6.2 to obtain W often cannot receive good results. For the transistors with

width smaller than 10um, using 2 samples have larger error. Using 50 samples or 200 samples all have good accuracy, and the result of using 200 samples is only a little better than using 50 samples, but it costs 4 times of the simulations.

Table 6.1. Typical errors with different number of samples for OIOPD Method

technology	region	W	200 samples	50 samples	2 samples
250nm	Saturation	>10um	1.90%	1.91%	0.88%
250nm	Saturation	<10um	0.41%	0.44%	6.31%
250nm	Linear	>10um	2.13%	2.13%	1.86%
180nm	Saturation	>10um	2.84%	2.83%	1.71%
180nm	Linear	>10um	1.50%	1.52%	0.99%
180nm	Linear	<10um	0.77%	0.77%	6.45%
90nm	Saturation	>10um	1.81%	1.84%	1.86%
90nm	Saturation	<10um	0.05%	0.06%	4.32%
90nm	Linear	>10um	0.24%	0.26%	0.08%

Figure 6.4 shows the errors of using 200, 50 and 2 samples from 0.6um to 50um under typical device biases and L in a 0.25um technology.

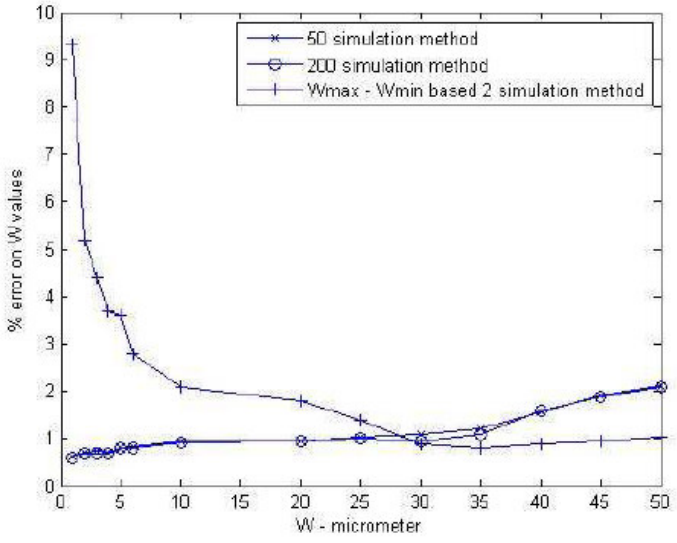

Figure 6.4. Errors for different number of samples for OIOPD Method

According to the experimental results, the OIOPD method uses the following rules to set the number of samples: (1) First uses the maximum and minimum W as the samples. (2) If the estimated transistor width is smaller than 10 um uses randomly distributed 50 samples within the minimum allowed W and 10um. On the other hand, it can be seen from Figure 6.4 that even when the width is smaller than 10um the error is within 10% in most cases. It will be shown on the experiment results. Hence, it is also reasonable to use 2 samples for all the transistors.

According to experiments, it was found that SPLINE interpolation [33] performs better than linear and cubic interpolation. Hence, OIOPD use SPLINE interpolation.

6.2.3. Selection of the LUT for OPD Method

There exists a trade-off between the accuracy, the look-up time and the memory requirement for different sizes of LUT. To make a fair comparison, a good LUT is necessary to be selected. The example below shows the selection of the LUT for a 0,18um technology, and the selection of 90nm technology is done in the same way. The accuracy, look-up time for each transistor and memory requirement of 3 LUT are shown in Table 6.2. LUT 1 uses uniformly distributed 54 points within the range of W of the technology (0,24um to 800um). LUT 2 uses 104 points of W, and LUT 3 uses 206 points of W. L has a range of 0,18um to 10um, V_{ds}, V_{gs}, V_{bs} have a range of 0 to 1,8 Volts. LUT1 has 810,000 points, LUT2 has 1,560,000 points and LUT3 has 3,090,000 points. In the test process, 278 test points are selected, 40% of which are in the linear region, 60% of which are in the saturation region, and they are uniformly distributed. The experiments are run on a PC with Xeon processor and 8GB RAM memory in Linux system.

Table 6.2. Different LUT for 0,18um technology

Look-up Tables	Error	Look-Up Time	Memory
LUT1	31%	0.62 sec	55.6 MB
LUT2	23%	1.19 sec	108.4 MB
LUT3	20%	2.24 sec	205.4 MB

It can be seen that LUT 1 (the sparsest table) has the fastest speed to find the data in the LUT but with the lowest accuracy. LUT 2 spends longer time, but renders a remarkable improvement (8%) by doubling the points in the LUT. Although LUT 3 provides the best accuracy, it consumes twice the time of LUT 2 and the improvement in accuracy is not impressive (only 3%) compared with LUT 2. Note that the look-up time for one transistor is not long to each LUT. But in the analog sizing, thousand times of this estimation are necessary, so the look-up time is an important factor. Hence, LUT 2 is chosen to compare with OIOPD.

6.2.4. Comparisons of OIOPD with Different Methods

Table 6.3 shows the comparisons of the Golden Section Search method, NN, LUT and OIOPD. Traditional DC root solving methods include gradient-based ones and direct search ones (using no gradients). A typical method of gradient-based equation solving method is Newton-Raphson method. According to experiments, Newton-Raphson method always causes SPICE unconvergence problem and cannot give a result. Hence, Table 6.3 only includes the solution of Golden Section Search method. NN is selected in the same way as LUT. Tens of NN with different training data (at least 8000 training data has been used for each network) have been compared, different number of neurons and different training methods. However, their performances are all not good. Table 6.3 shows the best result obtained from these NNs.

Table 6.3. Comparisons of different method in a 0,18um technology

Methods	Error	Look-Up Time	Memory
Golden Section	872%	4.39 sec	X
NN	143%	0.30 sec	5 KB
LUT	23%	1.19 sec	108.4 MB
OIOPD	2.23%	0.003 sec	X

From Table 6.3, it can be seen that the accuracy of Golden Section Search and NN are far from anticipation. OIOPD has 10 times improvement on accuracy compared

with LUT and 400 times improvement on efficiency. In addition, no extra memory is needed. Comparison of LUT and OIOPD in a 90nm technology is shown in Table 6.4.

Table 6.4. Comparison of LUT and OIOPD in a 90nm technology

Methods	Error	Look-Up Time	Memory
LUT	23%	2.32 sec	475.6 MB
OIOPD	1.92%	0.0021 sec	X

From Table 6.4, it can be seen that OIOPD has approximately 12 times improvement on accuracy and 1100 times improvement on efficiency.

6.2.5. Comparisons of OIOPD with Extreme Cases of LUT

Since the best method competing with the proposed OIOPD method is LUT's, To see the most accurate cases of LUT method than LUT 2, more extreme (in terms of sampling points, so the memory) table options were used. Since the step sizes for V_{ds}, V_{gs}, V_{bs} are equal to 20, step size of L is equal to 50 and W is equal to 200 and 800 for 2 different tests. The errors on W for 0,25nm folded cascode example test points are:

Table 6.5. W errors of 11 transistors for the LUT with 200/800 samples of W

W deviations:	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	max	ave
W=200	7,5287	7,529	5,833	5,833	5,833	0,985	0,985	10,61	10,61	13,73	13,73	13,73	7,564
W=800	6,5774	6,577	5,241	5,241	5,241	0,761	0,761	8,463	8,463	11,49	11,49	11,49	6,3923

The second LUT above has 32,000,000 samples to find W. Since the accuracy reached up to the quality of % 6,39 more tests on LUT have been tried. In Table 6.6 below there are 3 different extremely large LUTs. Memory issue is given on Table 6.7.

Table 6.6. Different options for the large LUTs

Method	**Step Num**	**Step Num**	**Step Num**	**Step Num**	**Step Num**
Option A	L=50	Vgs=30	Vgs=30	Vgs=30	W=200
Option B	L=80	Vgs=40	Vds=40	Vds=40	W=200
Option C	L=100	Vgs=80	Vbs=80	Vbs=80	W=200
OIOPD	fixed	fixed	fixed	fixed	W=2 or W=50

Table 6.7. Memory needed by 3 LUTs A,B and C

	for generating every file seperately	overwriting the unneeded table files	Just the needed data
Method	**memory needed during data generation**	**memory needed during data generation**	**memory needed for data store**
Option A	125,8 Gbyte	8,16 Gbyte	6,12 Gbyte
Option B	473,6 Gbyte	30,72 Gbyte	23,04 Gbyte
Option C	4, 736 Tbyte	307,2 Gbyte	230,4 Gbyte
OIOPD	0 bit - online simulation		0 bit - online simulation

For 3 options of LUT and for the proposed method, the time issue related to the data acquisition for interpolation (for LUT it means reading and interpolating the data from that tree structure, for proposed method it means online simulation of single transistor) and the time consumed by the whole W guess process for both of the methods is as follows:

Table 6.8. Interpolation time for the 3 LUTs A,B and C

	Pentium Dual Core CPU-T4300@ 2.10GHz	Pentium Dual Core CPU-T4300@ 2.10GHz
	For entire Circuit (Folded Cascode)	**For entire Circuit (Folded Cascode)**
Method	**Look-up time**	**overall time of W guessing (with interpolation)**
Option A	0.35 sec * number of transistor = 3.85 sec.	0.39 sec * number of transistor = 4.29 sec.
Option B	0.35 sec * number of transistor = 3.85 sec.	0.39 sec * number of transistor = 4.29 sec.
Option C	0.35 sec * number of transistor = 3.85 sec.	0.39 sec * number of transistor = 4.29 sec.
OIOPD	0.42 sec.	0.48 sec.

The accuracy results for these 3 different LUT and proposed method are as follows:

Table 6.9. W errors for the 3 LUTs A,B and C

W errors (%)	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	max	ave
Option A	14,017	14,017	3,328	3,328	3,328	29,548	29,55	0,601	0,601	12,037	12,037	29,5482	11,127
Option B	7,4182	7,4182	0,135	0,135	0,135	11,08	11,08	2,903	2,903	0,278	0,278	11,0795	3,9786
Option C	6,4175	6,4175	0,242	0,242	0,242	5,946	5,946	3,837	3,837	2,1884	2,1884	6,4175	3,4094
OIOPD	1,9015	1,9015	0,1	0,1	0,1	1,8088	1,809	3,111	3,111	2,1355	2,1355	3,1114	1,6559

As seen from all the extreme LUTs, there is a huge need for the memory to store the tables. For an accurate LUT the need for the table is around hundreds of Gbytes. Also the look-up time in order to obtain the W value even for a single transistor is a lot larger than OIOPD for the LUTs performed. Finally the best W value convergence obtained by a LUT method is more than % 3,4 (for the average of the whole circuit) since it is less than % 1,7 for the OIOPD method.

It can be seen from Tables 6.10 and Table 6.11 that for all 3 different technology files (90nm, 180nm, 250nm) the errors on W guessing of the LUT methods is larger than % 6. This looks like an acceptable error; however, the problems including a large memory, long look-up time etc. still exist. Also the results show that W errors of all 11 transistors for OIOPD method is less than % 3,12 for the folded cascode amplifier example and again less than % 2,37 for the shown transistors of the gain boosted amplifier. Also the results show that for the gain boosted amplifier average error on W for the transistors is less than % 0,5 for any technology. This shows how powerfull the OIOPD is to converge to the real W values.

Errors on the W change the performance values of the circuits (fitness values). As a result, during the optimization flow, the objective functions with high error can occur if the

error on W is large. This will result in that, the optimization algorithm works less accurate while generating new individuals by using genetic operators. The performance value (gain, phase margin etc.) error comparison for the W's generated by the methods have also been tested for the gain boosted amplifier.

For the Folded Cascode Amplifier:

Table 6.10. W errors for the folded cascode amplifier with the best LUT and OIOPD Method

	OIOPD	OIOPD	OIOPD	LUT	LUT	LUT
W errors	90nm	180nm	250nm	90nm	180nm	250nm
(%)	1,8099	2,8429	1,9015	1,7148	28,8545	7,5286
	1,8099	2,8429	1,9015	1,7148	28,8545	7,5286
	0,1719	0,5017	0,1	8,1508	5,3437	5,8329
	0,1719	0,5017	0,1	8,1508	5,3437	5,8329
11 transistors	0,1719	0,5017	0,1	8,1508	5,3437	5,8329
	0,0968	0,2113	1,8088	21,9032	2,4734	0,9851
	0,0968	0,2113	1,8088	21,9032	2,4734	0,9851
	1,196	0,2251	3,1114	9,1056	3,2113	10,61
	1,196	0,2251	3,1114	9,1056	3,2113	10,61
	0,2487	1,5038	2,1355	2,4815	8,0672	13,728
	0,2487	1,5038	2,1355	2,4815	8,0672	13,728
average	**0,656227**	**1,006482**	**1,655855**	**8,623873**	**9,203991**	**7,563827**
max	**1,8099**	**2,8429**	**3,1114**	**21,9032**	**28,8545**	**13,728**

For the Gain Boosted Amplifier, the results are as given on Table 6.11 which can be seen below:

Table 6.11. W errors for the gain boosted amplifier with the best LUT and OIOPD Method

	OIOPD	OIOPD	OIOPD	LUT	LUT	LUT
W errors	90nm	180nm	250nm	90nm	180nm	250nm
(%)	0,0514	0,2987	0,0401	7,5701	7,3771	16,1057
	0,0514	0,2987	0,0949	7,5701	7,3771	1,0395
	0,0952	0,0015	0,0949	45,0067	0,7006	1,0395
	0,05	0,0015	0,108	23,0833	0,7006	14,6427
10 transistors	0,05	0,0452	0,108	23,0833	16,8199	14,6427
	0	0,0452	0,0187	127,1	16,8199	12,028
	0	1,6871	0,0187	127,1	2,4083	12,028
	0,0108	1,6871	0,0517	93,9189	2,4083	2,931
	0,0447	0,0228	0,0064	86,7337	0,8648	13,6703
	0	0,1683	2,3609	11,6032	8,8335	17,8758
average	**0,03535**	**0,42561**	**0,29023**	**55,2769**	**6,43101**	**10,60032**
max	**0,0952**	**1,6871**	**2,3609**	**127,1**	**16,8199**	**17,8758**
	OIOPD	OIOPD	OIOPD	LUT	LUT	LUT
Performance	90nm	180nm	250nm	90nm	180nm	250nm
errors	0,0087	0,142	0,0426	25,8735	6,0212	1,1837
(%)	0,0125	0,2907	0,0333	1,2912	22,5079	9,2834
	0,0511	0,0418	0,0333	13,5786	7,412	7,5435
	0,1522	0,2704	0,02	45,9572	49,531	2,6891
	0,0114	0,1546	0,0056	28,2165	7,4226	1,7755
	0,0539	0,1895	0,0093	10,6972	8,9543	4,8323
10 objectives	0,0295	0,0112	0	5,9866	2,9612	3,1823
	0,026	0,3114	0,0198	26,1336	0,9043	3,7862
	0	0	0,0545	0,0469	11,9212	11,6996
	0,0316	0,4187	0,1963	63,451	6,9753	8,4953
average	**0,03769**	**0,18303**	**0,04147**	**22,1232**	**12,4611**	**5,44709**
max	**0,1522**	**0,4187**	**0,1963**	**63,451**	**49,531**	**11,6996**

To mention again, the results show that, OIOPD is much powerfull even than a large LUT. Also it can be seen that large deviations on W result in the large errors on the performance values of the circuit (it goes up to % 63 for the LUT for 90nm) which inconveniences the work of the optimization algorithm.

From the results, it is also proven that for the technology scaling down, an old method (LUT) has more problems about the DC root solving. However the proposed method OIOPD does not suffer from the scale down of the technology and works perfect even for 90nm. This is an extra advantage of the proposed method.

6.3 Single-objective Optimization Tests for Gain-Boosted Amplifier

An example of using OIOPD to size complex analog circuit is provided. This example is very complex and needs more than 800 iterations. Because it is shown that the LUT method is much better than the traditional DC root solving methods and the regression-based methods, OIOPD is compared with a selected LUT . The gain-boosted which was given in the previous parts of the work has been optimized by the OIOPD and LUT-based analog sizing method. This circuit has 23 constraints (including those necessary to ensure the proper operating region for all transistors). 0.25um CMOS technology with 2.5V power supply s used. The transistor lengths are allowed to vary between the minimum value allowed by the technological process, 0.25um, up to 10um. The transistor widths are changed between the minimum technology value, 0.6um, up to 700um. The capacitor values could change from 100fF to 20pF. The design specifications are DC gain>135dB, GBW>180MHz, phase margin>70, gain margin<1, output swing>3.5V. All transistors should work in the saturation region. The optimization goal is the power consumption. For the search algorithm MSOEA [17] has been chosen since the optimization is a single optimization case and since it is easy to implement DC Driven methods on.

The population size is chosen as 80, the DE step size F is 0.8, the crossover probability CR is 0.8, δ is 0.2, OT is 0.9, and η in SBX is 10 [3]. Because of the influences of random numbers in evolutionary computation algorithms, 5 times are run for each

method and the average results are shown in Table 6.12. The experiments are run on a PC with Xeon processor and 8GB RAM memory in Linux system.

Table 6.12. Results of the OIOPD and LUT method on MSOEA Optimizer

Look-up Tables	Objective Function	Constraint Satisfaction	CPU Time
OIOPD	6.51 mW	23/23	4654.5 sec
LUT	9.20 mW	21.4/23	1.31 e+6 sec

From Table 6.12, it can be seen that the OIOPD method has better objective function values and satisfy all the 23 constraints in all the 5 runs. By LUT method, only 2 runs satisfy all the constraints. This is because the OIOPD method can provide more accurate estimations of W. The CPU time of OIOPD is also much smaller than that of using LUT. In order to get higher accuracy, the LUT need to include sufficient samples, and this increase the look up time, which is much longer than the time of simulations of 2 extreme points. Because this example needs a large number of iterations, it can be seen that the time cost of the LUT method is impractical, but the OIOPD method also performs well (according to experiments, normally for a less complex circuit, the time cost of the LUT method is long but practical).

6.4 Multi-objective Optimization Tests for Gain-Boosted Amplifier

The novel OIOPD based method has been implemented on the Enhanced MOEA/D-DE Algorithm, which was represented in Chapter 5, in order to see the enhancements in the results. As mentioned before, OIOPD is a very fast and accurate model to guess the W from DC points of the circuit. With this way the DC variables are used as the optimization variables which are easier to be optimized. This has lots of reasons which have been mentioned before but most importantly, again, it should be noted that a W can vary from 0,24um to 800um, since a voltage value can change from 0 Volts to 1,8 Volts for a 0,18um technology. This will lead to a faster convergence to the ideal Pareto Front.

The OIOPD based MOEA/D-DE has been compared with the original one for 2 objective optimization cases of gain boosted amplifier. The optimization functions have been chosen as phase margin-gm for the first test, gain-power for the second test and gain-phase margin for the third test. All of the runs have been realized for 100 iterations, 100 population size and 40 niche. Other parameters and the seed number of the random number generator are equalized to make it a fair comparison.

It should be noted that the reason for the small ranges of some of the objective functions are because small number of iterations. It was experimented that an optimization whose results will be used for a real design should have at least 300 iterations (this is an approximation for the global optimum points) for 2 objective optimization. However 100 iterations for 2 objective are good enough for the comparison cases. The results for 2 objective optimization of gain boosted amplifier with OIOPD based MOEA/D-DE and with Original MOEA/D-DE are as follows:

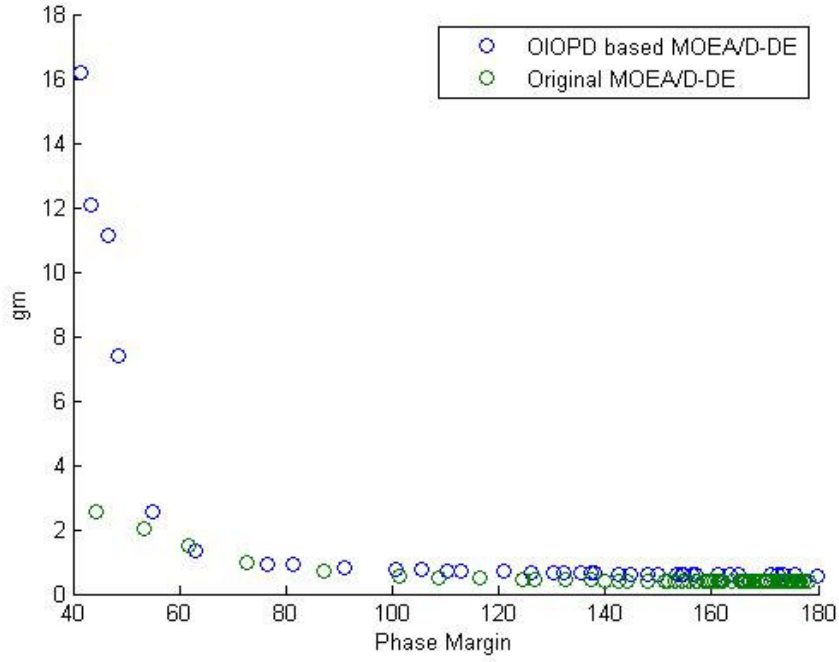

Figure 6.5. Phase Margin – Gm Optimization of Gain-boosted Amplifier with OIOPD based MOEA/D-DE and Original MOEA/D-DE

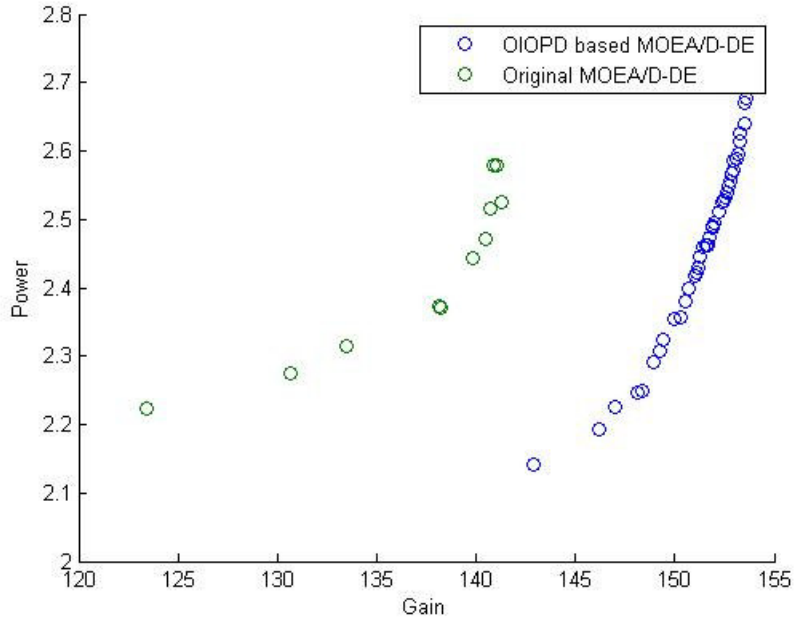

Figure 6.6. Gain – Power Optimization of Gain-boosted Amplifier with OIOPD based MOEA/D-DE and Original MOEA/D-DE

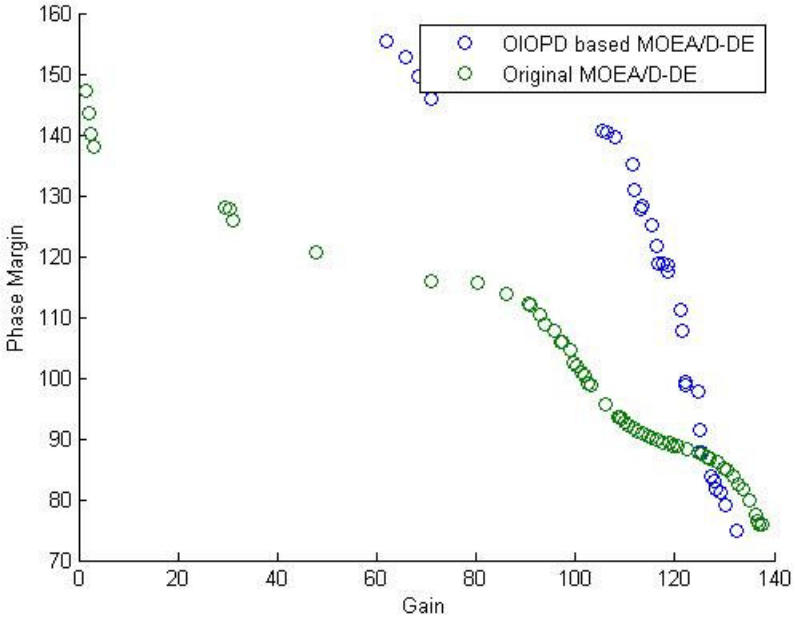

Figure 6.7. Gain – Phase Margin Optimization of Gain-boosted Amplifier with OIOPD based MOEA/D-DE and Original MOEA/D-DE

The results of the optimization show that OIOPD based DC root solving method implementation on MOEA/D-DE has increased the dominance quality and the solution ranges of the Pareto Fronts, which can be obtained by the original MOEA/D-DE.

7. CONCLUSIONS AND FUTURE WORK

Since there are several numbers of well designed CAD tools for digital design, this number is so limited for the analog CAD tools. It is clear that analog circuit design will always remain so important due to the fact that the nature we live in is analog. Increasing needs of complex analog designs, especially for the integrations on System on Chips (SoCs), are coming up with the needs of good analog CAD tools. One of the most important automation methods of analog EDA is the sizing optimization of the analog circuits.

During the thesis, several optimization methods have been discussed and it was mentioned that the Evolutionary Algorithms are used so often for analog sizing problem, because of their several advantages. A background work based on the decomposition of the whole problem into different scalar problems has been chosen and several enchancements have been realized to improve that algorithm. First of all some software enchancements have been realized to make it work faster. Later some new methods have been proposed in order to increase the quality of the convergence, dominance and distribution on Pareto Fronts etc. All these works leaded us to a new method called Enhanced MOEA/D-DE.

There are two choices for the selection of the optimization variables for analog sizing problem. The transistor dimensions can be directly optimized, or the DC operating points of the transistors can be optimized. The optimization with second type of variables are called OPD based methods. In these methods the critical point is finding the proper W from the DC points with a high accuracy. This will let the algorithm evaluate the circuit by using a SPICE simulator. In the second part of the project a DC root solving algorithm has been proposed. The novel method can guess W from the DC points in a fast and highly accurate way. This method has been compared with other OPD methods and it was seen that the results for the novel method OIOPD are much better. This novel method has been implemented on the Enhanced MOEA/D-DE to see the improvements. The results are given on Chapter 6. It can be seen that the proposed method enchanced the quality of the MOEA/D-DE algorithm proposed on Chapter 5. As a result the last version of the analog sizing optimization algorithm is the OIOPD based Enhanced MOEA/D-DE.

There are several works to be done in order to increase the quality of the algorithm and to make it more user friendly. First of all, the ranges of the objective functions on the Pareto Fronts can be more fair. For example, as it can be seen from Figure 6.6, the power objective is changing in a small range. To fix that problem some further works should be done on the normalization of the objective functions.

Also, an adaptive method for the determination of the iteration numbers is something to work on. A method like comparing the previous pareto front (what previous means here is the previous iteration's PF) with the present one and checking the improvement in a systematic way can be used to achieve that.

More future works can also be performed in order to improve the quality of the optimization algorithm.

Finally, all considered, it can be said that the OIOPD based Enhanced MOEA/D-DE Algorithm is an effective algorithm for optimizing the dimensions of the transistors on analog circuits. The results of the implemented methods building up the overall algorithm have been compared with the other works from the literature and sometimes slight, sometimes significant improvements have been obtained.

REFERENCES

1. Toumazou, C. and C. A. Makris, "Analog IC Design Automation: Part I-Automated Circuit Generation: New Concepts and Methods", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* Vol. 14, no. 2, February 1995

2. Van der Plas, G., G. Gielen, W. Sansen, *A Computer-Aided Design and Synthesis Enviroment for Analog Integrated Circuits*, Kluwer Academic Publishers, 2002.

3. Martens, E. S. J. and G. G. E. Gielen, *High-Level Modeling and Synthesis of Analog Integrated Systems*, Springer, 2008.

4. Martens, E. and G. G. E. Gielen, "Classification of analog synthesis tools based on their architecture selection mechanisms", *Integration – the VLSI journal* 41, pp. 238-252, 2008.

5. Gielen, G. G. E. and R. A. Rutenbar, "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits", *Proceedings of the IEEE,* vol. 88, no. 12, December 2000.

6. Branke, J., K. Deb, H. Dierolf and M. Osswald, *Finding Knees in Multi-objective Optimization,* KanGAL Report No: 2004010.

7. Subbotin, S. and A. Oleynik, "The Multi Objective Evolutionary Feature Selection", *TCSET 2008*, 19-23 February.

8. Deb, K., A. Prapat, S. Agarwal and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGAII", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, April 2002.

9. Zitzler, E., M. Laumanns and L. Thiele, *"SPEA2: Improving the Strength Pareto Evolutionary Algorithm"*, ETH Zurich, May 2001.

10. Xue, F., "*Multi-Ojective Differential Evolution: Theory and Applications"*, Rensselaer Polytechnic Institute, PhD Thesis, September 2004.

11. Smith, E., "What is an Evolutionary Algorithm", *Introduction to Evolutionary Computation,* Chapter 2, pp. 15-24.

12. Pohlheim, H., "Introduction Evolutionary Algorithms: Overview, Methods and Operators", *Genetic Algorithms Toolbox for Matlab*, December 2006.

13. Knowles, J. and D. Corne, *"On Metrics for Comparing Non Dominated Sets",* IRIDIA, Free University of Brussels Belgium, Dept. of Computer Science, University of Reading, December 20, 2001.

14. Zhang, Q. and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition", *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, December 2007.

15. Liu, B., V. Fernandez, Q. Zhang, M. Pak, S. Sipahi and G. Gielen, "An Enhanced MOEA/D-DE and Its Application to Multiobjective Analog Cell Sizing", *IEEE, Congress on Evolutionary Computation*, 23 July 2010.

16. Zhang, Q. and Y. W. Leung, "An Orthogonal Genetic Algorithm for Multimedia Multicast Routing", *IEEE Transactions On Evolutionary Computation*, vol. 3, no. 1, April 1999.

17. Liu, B., F. V. Fernandez, G. Gielen, R. C. Lopez and E. Roca, "A Memetic Approach to the Automatic Design of High-Performance Analog Integrated Circuits", *ACM Transactions on Design Automation of Electronic Systems*, vol. 14, no. 3, May 2009.

18. Messac, A., A. I. Yahaya, C. A. Mattson, *The normalized normal constraint method for generating the Pareto Frontier*, Springer, June 2003.

19. Das, S., A. Konar and U. Chakraborty, “Two improved differential evolution schemes for faster global search”, *Genetic and Evolutionary Computation Conference*, 2005. pp. 991-998.

20. Price, K., R. Storn and J. Lampinen, *Differential Evolution. A Practical Approach to Global Optimization*, Springer, 2005.

21. Coello Coello, C. A., G. B. Lamont, D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, p. 189.

22. Li, H. and Q. Zhang, “Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II”, *IEEE Transactions on Evolutionary Computation*, 2008. pp. 1-19.

23. CEC 2009, Benchmark Problems and Test Conditions, *“CEC 09 MOEA Competition”*, http://dces.essex.ac.uk/staff/qzhang/moeacompetition09.htm.

24. Agarwal, A. et al., “Fast and accurate parasitic capacitance models for layout-aware synthesis of analog circuits”, *Proc. of DAC*, pp. 145-150, 2001.

25. Mueller, D. et al., “Trade-off Design of Analog Circuits Using Goal Attainment and “Wave Front” Sequential Quadratic Programming”, *Proc. Of DATE*, pp. 16-20, 2007.

26. Leyn, F. et al., “An Efficient DC Root Solving Algorithm with Guaranteed Convergence for Analog Integrated CMOS Circuits”, *Proc. of ICCAD*, pp. 304-307, 1998.

27. Leyn, F. et al., “Analog Circuit Sizing with Constraint Programming Modeling and Minimax Optimization”, *Proc. Of ISCAS*, pp. 1500-1503, 1997.

28. McConaghy, T. et al., “Trustworthy, variation-aware structural synthesis of analog circuits via structural homotopy”, *IEEE TCAD*, pp. 1281-1294, 2009.

29. Graeb, H., *Analog Design Centering and Sizing*, Spring, 2009.

30. Wasserman, P. D., *Neural computing: theory and practice*, New York: Van Nostrand Reinhold, 1988.

31. Kelley, C., "Solving Nonlinear Equations Using Newton's method", *SIAM*, 2003.

32. Press, W. et al., *Numerical Recipes in C, The Art of Scientific Computing (2nd edition)*, Cambridge University Press, 1999.

33. Boor, C., *A Practical Guides to Splines*, Springer-Verlag, 1978.

34. Sönmez, Ö. S., *Circuit Level Analog Design Automation*, PhD Thesis, Boğaziçi University, Istanbul, 2010.

35. Iman, R. L., J. M. Davenport, and D. K. Zeigler, "Latin Hypercube Sampling (A Program User's Guide)", *Technical Report SAND79-1473*, Sandia Laboratories, Albuquerque, 1980.