ANALOG CELL SIZING USING ENHANCED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM (MOEA/D-DE) AND FORMING A FEEDBACK LOOP INTERFACE BETWEEN SACSES AND TOLAS

by

Süha Sipahi

BS, Electrical and Electronics Engineering, Yeditepe 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 Electrical and Electronics Engineering
Boğaziçi University
2011

ANALOG CELL SIZING USING ENHANCED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM (MOEA/D-DE) AND FORMING A FEEDBACK LOOP INTERFACE BETWEEN SACSES AND TOLAS

APPROVED BY:

Prof. Günhan Dündar (Thesis Supervisor)	.....
Prof. A. C. Cem Say	.....
Assoc. Prof. Arda D.Yalcinkaya	.....

DATE OF APPROVAL:

ACKNOWLEDGEMENTS

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Günhan Dündar, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the office to work in my own way. One simply could not wish for a better or friendlier supervisor.

I would like to thank my family members, especially my mother, and my father for supporting and encouraging me to pursue this degree.

I warmly thank Prof. Georges Gielen, for his valuable advice and friendly help. His extensive discussions around my work and interesting explorations in operations have been very helpful for this study.

During this work I have collaborated with many colleagues for whom I have great regard, and I wish to extend my warmest thanks to Bou Liu and Murat Pak who have helped me with my work in the Department of Electrotechnical Engineering at the Katholieke Universiteit Leuven, Belgium.

I would also like to thank Utku Kuscu ,my house mate, with whom I lived throughout the undergraduate and graduate degree and who took on my chores when I was immersed with my thesis work.

The financial support of TUBITAK is gratefully acknowledged.

ABSTRACT

ANALOG CELL SIZING USING ENHANCED MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM (MOEA/D-DE) AND FORMING A FEEDBACK LOOP INTERFACE BETWEEN SACSES AND TOLAS

This thesis investigates an evolutionary-based design system for automated sizing of analog integrated circuits(ICs). A new algorithm, called multi-objective evolutionary algorithm based on decomposition using enhanced differential evolution (MOEAD/D-DE), is proposed to design analog ICs with practical user-defined specifications. On the basis of the combination of HSPICE and MATLAB, the system links circuit performances, evaluated through electrical simulation, to the optimization system in the MATLAB environment, once a circuit topology is selected. The method has been tested through the sizing of several analog circuits and benchmark problems. The results show that design specifications are met and objective functions are highly optimized. Comparisons with available methods like NSGA-II, are also carried out, showing that the proposed algorithm has important advantages in terms of diversity and optimization quality. As a final work, an interface environment is created to link another optimization system (SACSES) to layout generator. This feedback loop lets the optimization to take into account the parasitic effects after the layout is drawn by TOLAS.

Keywords: Evolutionary algorithms, multiobjective optimization problems, pareto optimality.

ÖZET

GELİŞTİRİLMİS ÇOK OBJEKTİFLİ EVRİMSEL ALGORİTMA İLE SERİM ETKİLERİNİ DAHİL EDEREK ANALOG DEVRE SENTEZİ

Günümüz teknolojisinin geldiği nokta, büyük oranda tümdevre tasarımında gelinen seviyeyle paralellik göstermektedir. Eğilim daha cok çip üstü sistemler yönünde ilerlerken, bunların sayısal devre kısımları gelişmiş bilgisayar destekli tasarım aracları yardımıyla sentezlenebilmektedir. Analog devre tasarımının insan gücüne olan bagımlılığının daha yüksek oluşu, bilgisayar desteğinin bu alanda kullanımının gecikmesine neden olmustur. Fakat son zamanlarda evrimsel algoritmaların, yüksek güclü bilgisayarla olan isbirliği neticesinde istenilen performanstaki analog devrelerin transistör boyutları rahatlıkla elde edilebilmektedir. Tez calısması özetle analog devrelerdeki transiztör boyutlarının, evrimsel algoritmalar yardımıyla eniyilenerek elde edilmesine dayanmaktadır. Tercihen çoklu-performans fonksiyonlarının eniyilenmesinin daha alt problemlere ayrıştırılmasına dayanan yöntem kullanılmıştır. Daha sonra evrimsel algoritmanın arama ve yer değiştirme kısımlarında gerekli değişiklikler yapılarak gerek hız gerek çözüm çeşitliği açısından daha başarılı sonuçlar elde edildi. Bu amaçla katlanmıs kaskod devresi ve kazancı arttırılmıs kuvvetlendirici tasarımları test sonucu olarak sunulmustur. Elde edilen boyutlar serim üreten programa girilerek tasarımların serimleri otomatik olarak alınmıstır. Sonraki asama olarak serimler parazitik elemanlarına ayrıstırılarak, diğer bir eniyileme algoritmasına geri beslenmiştir. Amaç eniyilemeye parazitik etkileride dahil etmektir.

Anahtar Sözcükler: Evrimsel algoritma, çoklu hedefli optimizasyon, pareto optimizasyon kriteri

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

d	Distance between the objective functions
P	Population
N	Population size
F	Scaling factor for DE
Z^*	True pareto set
m	Dimension of the objective Functions
f	Fitness function
T	Number of the individuals in the neighborhood
B	The neighborhood space
g	Optimization Problem
α	Description of α
λ	Weight Vector Matrix
Ω	Solution Space for the optimization variables
γ	Greediness of the DE

LIST OF ACRONYMS/ABBREVIATIONS

CAD	Computer Aided Design
CR	Crossover Rate
DE	Differential Evolution
DM	Decision Maker
EA	Evalutionary Algorithm
EP	External Population
GBW	Gain Bandwidth Product
IGD	Inverted Generational Distance
LUT	Look-up Table
MODE	Multi-objective Differential Evolution
MOEA	Multi-objective Evolutionary Algorithm
MOP	Multi-Objective Problem
NSGA	Non-dominated Sorting Genetic Algorithm
PF	Pareto Front
PM	Phase Margin
PS	Pareto Set
TICER	Time Constant Equilibration Reduction

1. INTRODUCTION

In recent years, there has been an increasing tendency in the electronics market to integrate complete systems, which before occupied one or more boards, onto a single-chip or multichip module. This is the evolution toward systems on a chip (SoC) or systems on a package. Technologically, this integration has been made possible because of the increasing miniaturization of very large scale integration technology. Most of the functions in such an integrated system are performed with digital circuitry, which perform digital signal processing. Analog circuits, however, are always needed at the interface between the electronic system and the outer world. Nature is analog and interaction with nature or transportation of signals is, therefore, inevitable through analog interface circuits. Although the analog circuits occupy only a small part of the area in these mixed-signal ICs, they require an inversely large part of the design time and cost and are often responsible for design errors and expensive redesign iterations. Most steps in an analog design are basically still handcrafted, ranging from extensive and repeated SPICE simulation runs through manual place and route with the assistance of parameterized device generators. All this does not fit well with the short design cycles of time-to-market critical applications. Clearly, there is an industrial need to increase analog design productivity and to lower the design risk. The key to managing this increased design complexity while meeting the shortening time-to-market factor is the use of computer-aided design (CAD) and verification tools. Todays high-speed computers provide more than enough power to make large and detailed computations possible. What is needed to expedite the analog and mixed-signal design process is a structured methodology and supporting CAD tools to manage the entire design process and design complexity. CAD tools are also needed to assist or automate many of the routine and repetitive design tasks, taking away the tedium of manually designing these sections and providing the designer with more time to focus on the creative aspects of design. In addition, CAD tools can increase the productivity of designers, even for nonrepeatitive analog blocks.[7] Therefore, analog CAD and circuit design automation are likely to play a key role in the design process of the next generation of mixed-signal ICs and ASICs. And although the design of mixed-signal ASICs served as the initial

impetus for stepping up the efforts in research and development of analog design automation tools, the technology trend toward integrating complete systems on a chip in recent years has provided yet another driving force to support analog CAD efforts.
Unfortunately, the story is quite different on the analog side. There are not yet any robust commercial CAD tools to support or automate analog circuit design apart from circuit simulators and layout editing environments and their accompanying tools. Some of the main reasons for this lack of automation are that analog design in general is perceived as less systematic and more heuristic and knowledge-intensive in nature than digital design, and that it has not yet been possible for analog designers to establish a higher level of abstraction that shields all the device-level and process-level details from the higher level design. Analog IC design is more sensitive to nonidealities and all kinds of higher order effects and parasitic disturbances.
Research in analog design automation (DA) has been relatively slow. By the year 1985 only a handful of analog DA systems were reported and only few institutions worldwide demonstrated interest in analog design automation. Basically these can be classified into three main group.

(i) Optimization-Based Design Approach: Historically, the first attempts towards design automation were optimization-based. They consider the sizing of transistors of a user-given circuit topology as an optimization problem. Typically, these systems employ optimization algorithms to iteratively adjust transistor sizes in order to meet user input constraints and objectives.A simulator is used within the optimization loop to assess the performance of the circuit at each iteration.

(ii) Layout-Based Design Approach: Several systems have been developed that follow a layout-based design approach. In essence, this approach is an adaptation of the extensively used standard cell, gate-array,and parameterized cell methods found in the digital domain. Because with this approach designs are controlled to a large extent by the layout, it is also referred to as semi-custom bottom-up approach. Analog arrays are pre-designed and layed-out blocks of different sizes, configurations and levels of complexity, varying from single-component arrays to circuit arrays. The required func-

tions are designed by appropriately programming one or more levels of interconnect. Designing with analog arrays has several serious drawbacks. First of all, they do not provide the necessary design flexibility required for high performance analog circuits. Not only do they restrict the designer to the limited range of available active and passive components, but also to a limited range of component values.

(iii) Knowledge -Based Design Approach: Knowledge-based systems exploit domain knowledge to design analog ICs, and they address the design task in a full-custom way, thereby allowing for maximum flexibility and a potentially better coverage of the circuit's performance space.

- Hierarchical Approach: The underlying idea involves the breaking of the required circuit (or system) into smaller distinct parts or blocks. Each of these parts is assigned a set of specifications so that, if met, the combination of the performance of these parts will yield the desired circuit performance. The same procedure is repeated in a similar manner for the smaller blocks at lower hierarchical levels. The number of hierarchical levels depends upon the complexity of the circuit as well as the sophistication of the design system.
- Fixed Topology Approach: The fixed topology approach is the second knowledge-based design philosophy that has emerged and differs widely from the first one. At their most fundamental level, systems following this approach employ a sizing method to compute appropriate sizes for the devices within a given fixed circuit topology. These fixed, unsized, device level circuit topologies are stored in a knowledge base together with the necessary domain knowledge for dimensioning the devices. The nature of this domain knowledge depends on the method of computing the device sizes.
- Combined Hierarchical and Fixed Topology Approach: This is a system that fits into this class of design systems, since it puts together a circuit topology in a hierarchical manner whereas the design (i.e., sizing) of the topology is performed in a manner that resembles fixed topology systems. Systems that combine features from the hierarchical and fixed

topology approach provide an additional degree of design flexibility-topology modification. This can lead to a smaller circuit library and a wider coverage of circuit performances. However, they are not as flexible as fully hierarchical systems [8].

The CAD tool realized for the thesis work is based on optimization-based design approach. The aim of the thesis work is finding optimal dimensions for the transistors of an user-given topology in order to get the pareto optimal performance points in terms of gain, gain-bandwidth, offset, phase margin etc. The computation of these performance metrics are done by evaluation of performance function which corresponds to HSpice in the thesis work. After required performance values have been reached, the transistor sizes at hand are sent to layout-generation tool. As next step the layout parasitics are extracted by Calibre and post layout spice files are fed back to the optimizer to redesign the circuit by considering parasitic effects. The details of the optimization tool will be given in following chapters. Basically it exploits evolutionary algorithms because of its powerful capability for rapid convergence after enough number of iterations.

2. BACKGROUND

2.1. Multi-objective Optimization Problems

The Multiobjective Optimization Problem can be defined as a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually a part of a trade-off. Hence, the term optimize means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.

Typically, at the beginning of a design process, it is impossible to specify the relative importance of each objective function until the best capabilities are determined (e.g circuit capability). Thus the design process is necessarily an interactive one. The area of multiple objective optimization attempts to provide some results and geometrical interpretations which can server the designer as a guide in this process.

(i) Decision Variables

The decision variables are the numerical quantities for which values are to be chosen in an optimization problem. These quantities are denoted as x_j , $j = 1, 2, ..., n$. The vector x of n decision variables is represented by:

$$x(n) = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}$$

(ii) Constraints

In most optimization problems, there are always restrictions imposed by the particular characteristics of the environment or available resources (e.g. technological constrains for our case). These restrictions must be satisfied in order to consider a certain solution acceptable. All these restrictions in

general are called constraints, and they describe dependences among decision variables and parameters involved in the problem. These constraints are expressed in form of mathematical inequalities:

$$g_i(\mathrm{x}) \leq 0 \; i = 1, \ldots, m$$

(iii) Criteria, Objectives, and Goals

Criteria generally denote evaluative measures, dimensions or scales against which alternatives may be gauged in a value or worth sense. Objectives are sometimes viewed in the same way, but may also denote specific desired levels of attainment or vague ideals. Goals usually indicate either of the latter notions. A distinction commonly made in OR is to use the term goal to designate potentially attainable levels, and objective to designate unattainable ideals. The convention used in the thesis is using the terms objective, criteria, and attribute interchangeably to represent an MOP' s (multiobjective optimization problem) goals or objectives to be achieved. The terms objective space or objective function space are also used to denote the coordinate space within which vectors resulting from evaluating an MOP' s solutions are plotted.

The objective functions are designated:$f_1(x), f_2(x), ..., f_k(x)$, where k is the number of objective functions in the MOP being solved. Therefore, the objective functions form a vector function $f(x)$ which is defined by:

$$f(x) = [f_1(x), f_2(x), ..., f_k(x)]^T \tag{2.1}$$

(iv) Multiobjective Optimization

The single objective formulation is extended to reflect the nature of multiobjective problems where there is not one objective function to optimize, but many. Thus, there is not one unique solution but a set of solutions. This set of solutions are found through the use of Pareto Optimality Theory which is explained in following sections. Note that multiobjective problems require

a decision maker to make a choice of x_i values. The selection is essentially a tradeoff of one complete solution x over another in multiobjective space. More precisely, multiobjective problems (MOPs) are those problems where the goal is to optimize k objective functions simultaneously. This may involve the maximization of all k functions, the minimization of all k functions or a combination of maximization and minimization of these k functions.

A general MOP is defined as minimizing (or maximizing) $F(x) = (f_1(x), ..., f_k(x))$ subject to $g_i(x) \leq 0$, $i = 1, ..., m$, and $h_j(x) = 0$, $j = 1, ..., p$ $x \in \Omega$

An MOP solution minimizes (or maximizes) the components of a vector $F(x)$ where x is a n -dimensional decision variable vector $x = (x_1, ..., x_n)$ from some universe Ω.

It is noted that $g_i(x) \leq 0$ and $h_j(x) = 0$ represent constraints that must be fulfilled while minimizing (or maximizing) $F(x)$ and Ω contains all possible x that can be used to satisfy an evaluation of $F(x)$.

2.2. Evolutionary Algorithms

Evolutionary algorithms (EAs) are based on the Darwinian theory that de- scribes the evolution of nature. The Darwinian theory of evolution explains the adaptive change of species by the principle of natural selection, which favors those species for survival and further evolution that are most suitable to their environmental conditions. However, modern biochemistry and genetics have extended the Darwinian theory by microscopic findings concerning the mechanisms of heredity. The major applications of evolutionary algorithms are in optimization, although they have also been used to build classifier systems and finite-state machines. In comparison with traditional optimization techniques, such as calculus-based methods, 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 applied to solve those problems that are characterized by chaos, chance, and nonlinear interactivity which tend to be intractable to traditional methods. In the thesis, the problem in question is quite nonlinear and suitable for evolutinary algorithms [9].

2.2.1. Evolutionary Operators

The majority of the current implementations of evolutionary algorithms de- scend from three strongly related but independently developed approaches, namely, Genetic Algorithms (GAs), Evolutionary Programming (EP), and Evolution Strategies (ESs). Genetic programming is particularly developed to deal with tree structures such as artificial logic and biological computing. The main streams of evolutionary algorithms have been developed independently over the past thirty years, although they are closely related with each other in terms of their underlying principles. The operators of one evolutionary approach are often different from those of another. Thus, for example, genetic algorithms usually strongly emphasize recombination. Evolution strategies and evolutionary programming, however, concentrate on mutation, although sometimes evolution strategies also incorporate recombination as an operator.Although these differences exist among the different forms of evolutionary algorithm, they all rely on the concept of a population of individuals, which undergoes the actions of probabilistic operators such as mutation, selection, and recombination to evolve toward better fitness values of individuals. The fitness of an individual rejects its objective function value with respect to a particular objective function to be optimized. The mutation operator introduces new information into the population by randomly generating variations to individuals; and the recombination operator, if there is such operation in the evolutionary algorithm, typically performs an information exchange between different individuals from a population. The selection operator imposes a driving force on the process of evolution by preferring individuals with high fitness values. There is usually a main loop in evolutionary algorithms, which consists of such evolutionary operators as recombination, mutation, fitness evaluation, and selection. This loop is iterated for a number of generations until the computing time is exhausted, a sufficiently good solution is found, or some other termination criterion is fulfilled.

2.3. Pareto Optimality

A solution $x \in \Omega$ is said to be Pareto Optimal with respect to Ω if and only if there is no $x' \in \Omega$ for which $v = F(x') = (f_1(x'), ..., f_k(x'))$ dominates $u = F(x) =$

$(f_1(x), ..., f_k(x))$ The phrase Pareto Optimal is taken to mean with respect to the entire decision variable space unless otherwise specified.

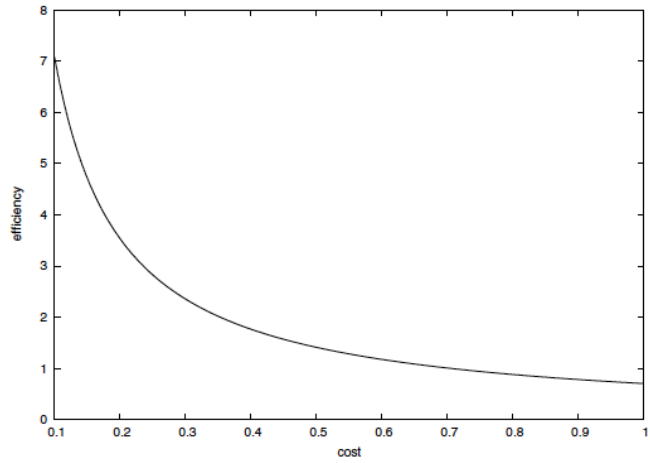

Figure 2.1. An example of a problem with two objective functions.

In words, this definition says that x^* is Pareto optimal if there exists no feasible vector x which would decrease some criterion without causing a simultaneous increase in at least one other criterion (assuming minimization).

Pareto optimal solutions are those solutions within the decision space whose corresponding objective vector components cannot be all simultaneously improved. These solutions are also termed non-inferior or efficient solutions, with the entire set represented by P^*. Their corresponding vectors are termed nondominated; selecting a vector(s) from this vector set (the Pareto front set PF^*) implicitly indicates acceptable Pareto optimal solutions or decision variables. These solutions may have no apparent relationship besides their membership in the Pareto optimal set. They form the set of all solutions whose associated vectors are nondominated; Pareto optimal solutions are classified as such based on their evaluated functional values.

When plotted in objective space, the nondominated vectors are collectively known as the Pareto front. Again, P^* is a subset of some solution set. Its evaluated objective vectors form PF^*, of which each is nondominated with respect to all objective vectors produced by evaluating every possible solution in Ω. In general, it is not easy to find an analytical expression of the line or surface that contains these points and in most cases, it turns out to be impossible. The normal procedure to generate the Pareto front is to compute many points in Ω and their corresponding f(Ω). When there is a

sufficient number of these, it is then possible to determine the nondominated points and to produce the Pareto front. A sample Pareto front is shown in Figure 2.1 is an example of a problem with two objective functions: cost and efficiency. The Pareto front or trade-off surface is delineated by a curved line [10].

2.4. Multi-objective Evolutionary Algorithm

The exploration of Evolutionary Multi-Objective Optimization to solve the class of multi-objective problems has increased in recent years. The ideal solution for a multi-objective problem is the one that optimizes all criteria simultaneously. However, such an ideal solution can never be obtained in practical applications where outcome criteria may be fundamentally inconsistent. Optimal performance according to a single objective, if such an optimum exists, often implies unacceptably low performance in one or more of the other objective dimensions, creating the need for compromise to be reached. As mentioned above, it is advantageous to obtain the Pareto optimal set before evoking the preferences of DM(decision maker). Evolutionary algorithms inherently explore a set of possible solutions simultaneously. This characteristic enables the search for an entire set of Pareto optimal solutions, at least approximately, in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of traditional mathematical programming techniques. Additionally, evolutionary algorithms are less susceptible to problem dependent characteristics, such as the shape of the Pareto front (convex, concave, or even discontinuous).
The first actual implementation of what it is now called a multi-objective evolutionary algorithm (or MOEA, for short) is credited to David Schaffer, who proposed the Vector Evaluation Genetic Algorithm (VEGA), in 1984.

Definition Given a function $f : \Omega \subseteq R_n \rightarrow R_k$, $\Omega \neq \phi$, $k \geq 2$, for $x \in \Omega$ the set $PF^* \triangleq f(x_i^*) >$ (-∞, ..., -∞) is called global minimum if and only if

$$\forall\ x \in \Omega : f(x_i^*) \leq f(x)$$

Then, x_i^*, $i = 1, \ldots, n$ is the global minimum solution set (i.e., P^*), f is the multiple objective function, and the set Ω is the feasible region.

Let $\phi : I \rightarrow \Re^k$, ($k \geq 2$, a multiobjective fitness function). If this multiobjective fitness function is substituted for the fitness function in above definition then the algorithm shown in 2.2 is called a Multiobjective Evolutionary Algorithm.

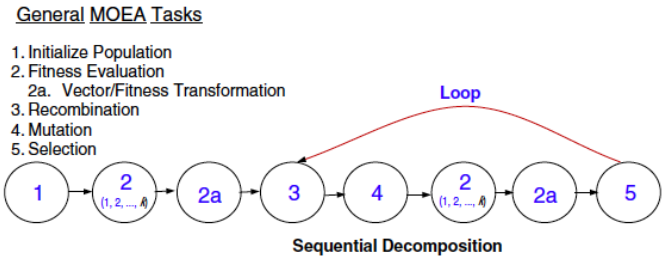

Figure 2.2. MOEA Description [1].

By definition, Task 2 in the MOEA computes k (where $k \geq 2$) fitness functions. In addition, because MOEAs expect a single fitness value with which to perform selection, additional processing is sometimes required to transform MOEA solutions' fitness vectors into a scalar (Task 2a). The major difference of MOEAs from single-objective EAs lies in the evaluation procedure that assigns fitness values to each individual based on Pareto dominance. The dominant solutions are given more credits to reflect their closeness to the ultimate Pareto optimal set. Most MOEA implementations consider the situation that no preference information is given by the DM a priori. The fitness assignment in these algorithms is based purely on Pareto dominance. Others consider to incorporate preference information from DM into fitness assignment.

The idea of using Pareto-based fitness assignment was first proposed by Goldberg [11]. He suggested the use of nondominated ranking and selection to move a population toward the Pareto front in a multiobjective optimization problem. The basic idea is to find the set of strings in the population that are Pareto nondominated by the rest of the population. These strings are then assigned the highest rank and eliminated from further contention. Another set of Pareto nondominated strings are determined from the remaining population and are assigned the next highest rank. This process continues until the population is suitably ranked. Goldberg also suggested the use of some kind of niching technique to keep the GA from converging to a single point on the front. A niching mechanism such as sharing would allow the GA to maintain indi-

viduals all along the nondominated frontier. In order to preserve population diversity within each rank, the fitness sharing techniques can be naturally applied. Fonseca and Fleming [11] implemented fitness sharing in the objective space and provided theory for estimating the necessary niche size with consideration of the Pareto properties in MOEA. The NSGA performed sharing in the decision variable space. Other than the major stream of fitness sharing in order to ensure variability in the population, mating restriction has been considered based on certain distance metric between individuals in objective space by Fonseca and Fleming [11].

MOEA approaches have been classified into three major categories. These categories are:

- Priori Techniques: By definition, these a priori techniques require a decision maker (DM) to define the MOP objective relative importance prior to search.
- Progressive Techniques: In this technique the DM normally has to define goals to bias the search, and this requires an interactive work that may be difficult and inefficient when nothing about the problem is known. This technique basically based on that the decision maker is unable to indicate preference information a priori because of the complexity of t he problem. Thus, the decision maker specifies and adjusts his or her preferences at the same time as he or she is learning more about the problem. But a high effort is required from the decision-maker during the whole search process. So that as a rule these methods are not used [12].
- Posteriori Techniques: A posteriori techniques are explicitly seeking $P_{(true)}$ and $PF_{(true)}$. Thus, the emphasis is now to perform a search as widespread as possible, as to generate as many different elements of the Pareto optimal set as possible. The decision making process will now take place after completing the search.

The algorithm which is used throughout the thesis is a population based posteriori method. As mentioned above, DM makes its preference right after the searching part is over.

2.5. Differential Evolution

Differential Evolution (DE) has recently proven to be an efficient method for optimizing real-valued multi-modal objective functions. Besides its good convergence properties and suitability for parallelization, DE's main assets are its conceptual simplicity and ease of use, having only a few control variables which remain fixed throughout the entire minimization procedure. DE is a simple yet powerful evolutionary algorithm by Price and Storn [13] that has been successfully used in solving single-objective optimization problems. Hence, several researchers have tried to extend it to handle MOPs. DE is a simple evolutionary algorithm that creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness. This is a rather greedy selection scheme that often outperforms traditional EAs. Basically, DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. If the resulting vector yields a lower objective function value than a predetermined population member, the newly generated vector replaces the vector, with which it was compared, in the next generation; otherwise, the old vector is retained. This basic principle, however, is extended when it comes to the practical variants of DE. For example an existing vector can be perturbed by adding more than one weighted difference vector to it. In most cases, it is also worth to mix the parameters of the old vector with those of the perturbed one before comparing the objective function values. There exists several variants of DE which will explained in the following sections. The pseudocode of the DE algorithm can be seen Figure 2.3

2.5.1. Scheme DE/rand/1

For each vector $x_{i,G}$, $i = 0, 1, 2, ..., NP-1$, a perturbed vector $v_{i,G+1}$ is generated according to Equation 2.2

$$v_{i,G+1} = x_{r1,G} + F(x_{r2,G} - x_{r3,G}) \tag{2.2}$$

Differential Evolution

1. Evaluate the initial population $\mathcal{P}$ of random individuals.
2. While stopping criterion not met, do:
   2.1. For each individual P_i ($i = 1, \ldots, popSize$) from $\mathcal{P}$ repeat:
      (a) Create candidate C from parent P_i.
      (b) Evaluate the candidate.
      (c) If the candidate is better than the parent, the candidate replaces the parent. Otherwise, the candidate is discarded.
   2.2. Randomly enumerate the individuals in $\mathcal{P}$.

Figure 2.3. DE's main procedure [2].

with r_1, r_2,$r_3 \in$ [0, $NP - 1$], integer and mutually different and F > 0.
The randomly chosen integers r_1, r_2,r_3 are also chosen to be different from the running index i. F is a real and constant factor $\in$ [0, 2] which controls the amplification of the differential variation ($x_{r2,G}$ - $x_{r3,G}$). Figure 2.4 shows a two- dimensional example that illustrates the different vectors which play a part in the vector-generation scheme. The notation: $DE/rand/1$ specifies that the vector to be perturbed is randomly chosen, and that the perturbation consists of one weighted difference vector.

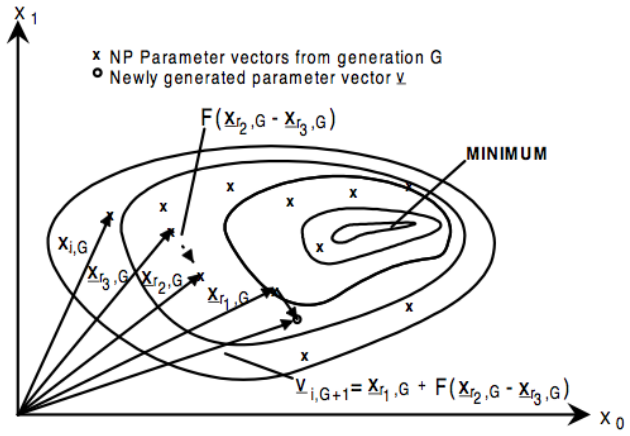

Figure 2.4. The process for generating $v_{i,G+1}$ in scheme $DE/rand/1$ [2].

2.5.2. Scheme DE/best/1

Basically, this scheme works the same way as $DE/rand/1$ except that it generates the vector $v_{i,G+1}$ according to Equation 2.3 :

$$p_i = \lambda.p_{best} + (1-\lambda).p_i + F.\sum_{k=1}^{K}(p_{i_a^k} - p_{i_b^k}) \tag{2.3}$$

where p_{best} is the best individual in the parent population, λ represents greediness of the operator, and K is the number of perturbation vectors, F is the scale factor of the perturbation, $p_{i_a^k}$ and $p_{i_b^k}$ are randomly selected mutually distinct individual pairs in the parent population, and p_i' is the offspring that is generated; λ, K, and F are the parameters associated with the algorithm. This time, the vector to be perturbed is the best performing vector of the current generation. In order to increase the potential diversity, crossover is introduced. In the thesis work this version of DE is exploited instead of using the simulated binary crossover (SBX) method of the MOEAD/D.

2.6. Decomposition Methods

There is no decomposition involved in the majority of the current state-of-the-art multiobjective evolutionary algorithms. These algorithms treat a MOP as a whole. They do not associate each individual solution with any particular scalar optimization problem. In a scalar objective optimization problem, all the solutions can be compared based on their objective function values and the goal of a scalar objective evolutionary algorithm (EA) is often to find a 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. Therefore, conventional selection operators, which were originally designed for scalar optimization, cannot 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 readily extended for dealing with MOPs. The idea of decomposition

has been used to a certain extent for MOPs. For example, the two-phase local search (TPLS) [14] considers a set of scalar optimization problems, in which the objectives are aggregations of the objectives in the MOP under consideration, a scalar optimization algorithm is applied to these scalar optimization problems in a sequence based on aggregation coefficients, a solution obtained in the previous problem is set as a starting point for solving the next problem since its aggregation objective is just slightly different from that in the previous one. The multiobjective genetic local search aims at simultaneous optimization of all aggregations constructed by the weighted sum approach or Tchebycheff approach. At each iteration, it optimizes a randomly generated aggregation objective.

2.6.1. Weighted Sum Approach

In the weighting method, the idea is to associate each objective function with a weighting coefficient and minimize the weighted sum of the objective functions are transformed into a single objective function. We suppose that the weighting coefficients w_i are real numbers such that $w_i \geq 0$ for all $i = 1,2,...,k$. It is also usually supposed that the weights are normalized that is, $\sum_{i=1}^{k} \lambda_i = 1$. To be more exact, the multiobjective optimization problem is modified into the following problem [15]:

$$\begin{gathered} \text{minimize } g^{ws}(\mathrm{x}\lambda) = \sum_{i=1}^{k} \lambda_i \cdot f_i \\ \text{subject to } x \in S, \\ . \end{gathered}$$

where $\lambda_i \geq 1$ for all $i = 1,..., k$ and $\sum_{i=1}^{k} \lambda_i = 1$.
where we use $g^{ws}(\mathrm{x}\lambda)$ to emphasize that λ_i is a coefficient vector in this objective function, while x is the variables to be optimized. To generate a set of different Pareto optimal vectors, one can use different weight vectors λ_i in the above scalar optimization problem. If the PF is concave (convex in the case of minimization), this approach could work well. However, not every Pareto optimal vector can be obtained by this approach in the case of nonconcave PFs.

2.6.2. Tchebycheff Approach

In this approach, the scalar optimization problem is in the form:

$$\text{minimize } g^{te}(x|\lambda,z^*) = \max(\lambda_i f_i(x) \text{ - } z^*)$$

$$x \in \Omega \tag{2.4}$$

where $z^* = (z_1^*,...,z_m^*)^T$ is the reference point i.e $z_i^* = \max(f_i(x) \mid x \in \Omega)$ 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 Equation 2.4 is a pareto optimal solution. Therefore, one is able to obtain different Pareto optimal solutions by altering the weight vector.
In the thesis work performances of both methods will be presented by revealing the result of real-world problem optimization.

2.6.3. Boundary Intersection (BI) Approach

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 be 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 subproblem is considered [14]:

$$\begin{aligned} &\text{minimize } g^{bi}(x|\lambda, z^*) = d \\ &\text{subject to } z^* - F(x) = d \cdot \lambda \end{aligned}$$

As shown in Figure 2.5, the constraint $z^* - F(x) = d \cdot \lambda$ 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.

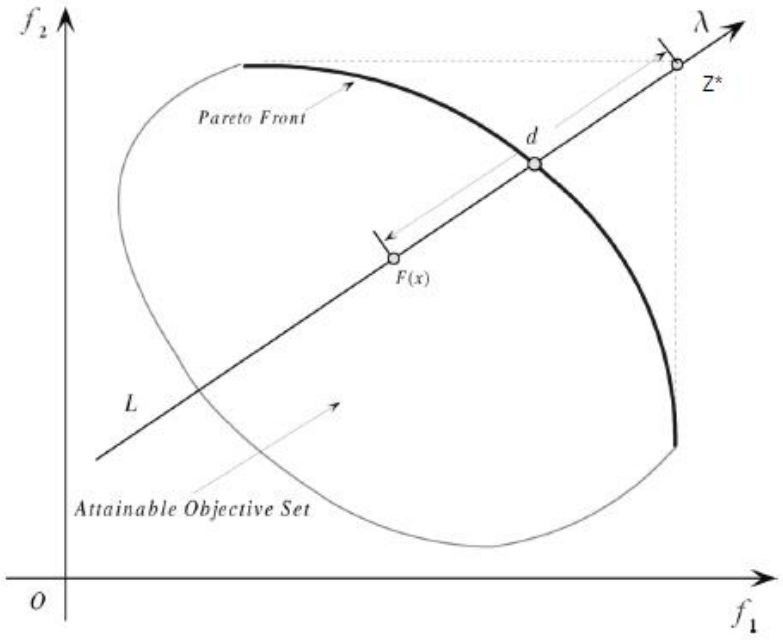

Figure 2.5. Boundary intersection approach [1].

$$\text{minimize } g^{bi}(x|\lambda, z^*) = d_1 + \theta \cdot d_2$$
$$\text{subject to } x \in \Omega \text{ where}$$

$$d_1 = \frac{||(z^* - F(x))^T \cdot \lambda||}{||\lambda||} \tag{2.5}$$

$$z^* - F(x) = d \cdot \lambda \tag{2.6}$$

θ is an previously set penalty parameter. If y is the projection of $F(x)$ on the line L, as shown in figure 2.5, d_1 will be the distance between z^* and y. d_2 is the distance between $F(x)$ and L. If θ is set set appropriately, the solutions to equation 2.5 and 2.6 should be very close. Hereafter, this method is called the penalty-based boundary intersection (PBI) approach [1].

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

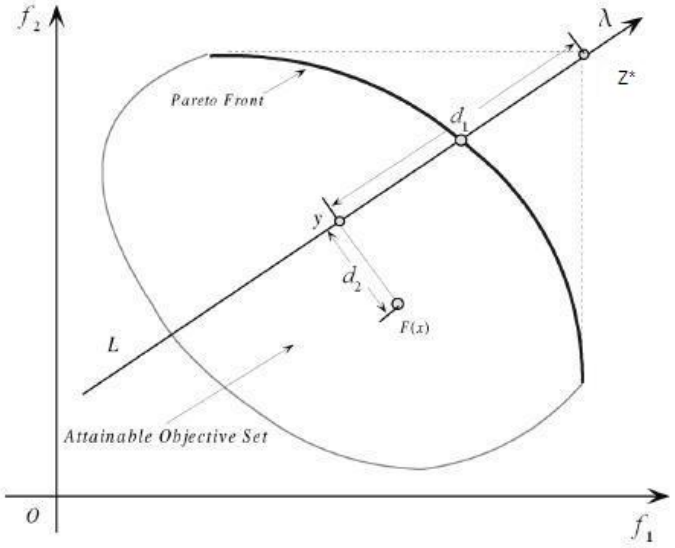

Figure 2.6. Illustration of penalty-based boundary intersection approach [1].

- 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 [1].

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.

3. PERFORMANCE METRICS REGARDING PARETO FRONTS

Comparing two evolutionary multiobjective algorithms hence requires comparing the nondominated sets they produce. If we call the true pareto front Z^*, then how do we compare a result which produces a single point $a \in Z^*$ with another result which yields a widespread set of nondominated points B, none of which dominates the discovered solution. But in the latter case, although every discovered point can be dominated, we have perhaps a very wide representation of the shape of the trade-off surface absent in the former case and perhaps some of the points are not very far from true Pareto optimal. Until recently, it has been popular to be indicated simply by graphic plot. Results are compared clearly in this form revealing which algorithm is better. Lately, several metrics have been proposed for comparing non-dominated sets, each of which attempts to represent the quality of such a set by a single number. Such a measure can then allow statistical comparison between different algorithms in terms of diversity, convergence, etc.

3.1. Schott's Spacing Metric

This metric uses the following formula to measure how evenly the points are distributed. It is an independent metric, gives rise a complete ordering and is very fundamental. It exhibits neither monotony nor relativity, since Z^* may be non-uniform. As an advantages, first it can be used in conjunction with other metrics, it provides information about the distribution of vectors obtained. Second, it has low computational overhead. Third, it can be generalized to more than two dimensions by extending the definition of d_i. However it has a couple of disadvantages such as d_i does not specify the use of normalized distances, which may be problematic.

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

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

This metric is designed to measure the distribution of the computed non-dominated front. All of them involve calculating a certain form of distance among the solutions in the computed non-dominated front.

3.2. IGD Metric

The inverted generational distance (IGD) is also used to evaluate the performance of the algorithms. To quantify this information, a large set of evenly spaced points on the Pareto-optimal front is generated. Let the size of this set be H. The minimum Euclidean distance of each point in this set from the obtained solution set is computed. Let this distance be l_i for the i^{th} element of the Pareto-optimal set. Then the IGD metric is given by

$$\text{IGDmetric} = \frac{\sum_{i=1}^{H} l_i}{H} \tag{3.2}$$

This measure of convergence indicated how far is the true Pareto-optimal front from the obtained front by each of the algorithms. Algorithm A is better than algorithm B in terms of convergence (diversity) if IGD of algorithm A is less than algorithm B. The IGD metric for the case of two objectives is pictorially depicted in Figure 3.1 [16]. The IGD metric measures both the convergence and the spread of the obtained solutions.

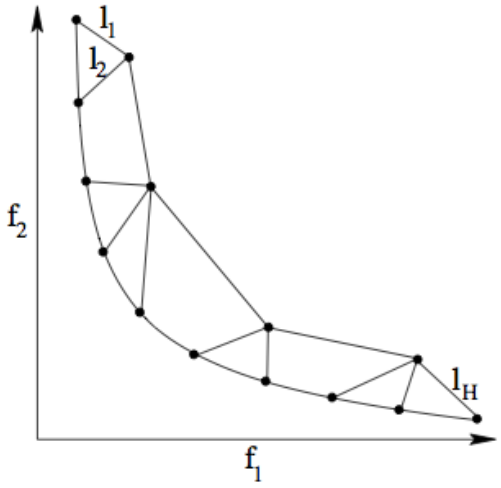

Figure 3.1. Inverted Generational Distance Metric.

4. MULTI-OBJECTIVE OPTIMIZATION BASED ON DECOMPOSITION (MOEA/D)

A multiobjective optimization problem (MOP) can be stated as follows:

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

Ω is the decision space where F: $\Omega \rightarrow R^m$ consists of m real-valued objective functions and R^m is called objective space. Very often, since the objectives in the function above contradict each other, no point in Ω maximizes all the objectives simultaneously. One has to balance them. The best tradeoffs among the objectives can be defined in terms of Pareto optimality. Analog cell sizing problem is a great example of MOP whose objective functions are the performance metrics of the circuit such as power, gain, area etc. and each of these objective functions could be an optimal solution of a scalar optimization problem. In other words PF approximation can be decomposed into a number of a scalar objective optimization subproblems. Several methods serve a purpose for constructing aggregation functions. The most popular ones among them include the weighted sum approach and Tchebycheff approach and both are explained in Chapter 2. Majority of the multiobjective evolutionary algorithms do not exploit this concept of decomposition. These algorithms treat a MOP as a whole. However in a scalar objective optimization problem, all the solutions can be compared based on their objective function values and a scalar objective evolutionary algorithm (EA) often tries to find one single optimal solution. In MOPs 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. Therefore, conventional selection operators, which were originally designed for scalar optimization, cannot 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 readily extended for dealing with MOPs. The popular fitness assignment strategies include alternating objectives-based fitness assignment such as the vector evaluation genetic algorithm (VEGA) [17], and domination-based fitness assignment such as Pareto archived evolutionary strategy (PAES) [18], strength Pareto evolutionary algorithm II (SPEA-II) [15], and nondominated sorting genetic algorithm II (NSGA-II) [19].

4.1. General MOEA/D Framework

MOEA/D explicitly decomposes the MOP into N scalar optimization subproblems. It solves these subproblems simultaneously by evolving a population of solutions. At each generation, the population is composed of the best solution found so far (i.e. since the start of the run of the algorithm) for each subproblem. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors.The optimal solutions to two neighboring subproblems should be very similar. Each subproblem is optimized in MOEA/D by using information only from its neighboring subproblems. Any decomposition approaches can serve this purpose. In the following description, we suppose that the Tchebycheff approach is employed.

Let $\lambda^1, ..., \lambda^N$ be a be a set of even spread weight vectors and z^* be the reference point. As mentioned above the problem of approximation of the PF of following equation

$$F(x) = (f_1(x), ..., f_m(x))^T \tag{4.1}$$

can be decomposed into N scalar optimization subproblems by using the Tchebycheff approach and the objective function of the j^{th} subproblem is :

$$g^{te}(x\lambda^j, z^*) = max_{1 \leq i \leq m}(\lambda_i | f_i(x) - z_i^*|) \tag{4.2}$$

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

Note that g^{te} is continuous of λ , and the optimized solution of $g^{te}(x|\lambda^i, z^*)$ should be close to that of $g^{te}(x|\lambda^j, z^*)$ if λ^i and λ^j are close to each other. In other words, any weight vector can help out optimizing $g^{te}(x|\lambda^i, z^*)$ as long as it is close to λ^i. This is the major motivation behind the neighborhood concept.

In MOEA/D, a neighborhood of weight vector λ^i is defined as a set of its several closest weight vectors in $\lambda_1, ..., \lambda_N$. The neighborhood of the ith subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. The best solution of any subproblem composes of the population. Only the current solutions to its neighboring subproblems are used for optimizing a subproblem in MOEA/D.

At each run t of MOEAD/D with Tchebycheff method, it comes up with:

(i) A population N points $x^1, ..., x^N \in \Omega$, where x^i is the current solution to the ith subproblem.

(ii) $z = (z_1, ..., z_m)^T$ where z_i is the best value found so far for objective f_i.

(iii) $FV^1, ..., FV^N$, since FV^i is the fitness value for x^i, i.e., $FV^i = F(x^i)$ for all $i = 1, ..., N$;

(iv) Storing the non-dominated individuals to an external population which is called as EP.

Briefly the algorithm works regarding to following order:

Inputs:

(i) MOP is defined as in Equation 4.1

(ii) A stopping criterion;

(iii) N: the number of the subproblems considered in MOEA/D;

(iv) a uniform spread of N weight vectors $\lambda^1, ..., \lambda^N$;

(v) T: the number of the weight vectors in the neighborhood of each weight vector.

Output: External Population (EP)

The algorithm can decomposed into 3 main steps which are :

(i) Initialization: In this step, first, 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. Second an initial population, which is the set of solutions, is randomly generated and the objective functions are evaluated for these individuals. As third, $z = (z_1, ..., z_m)^T$ is initialized by a problem specific method. The minimum and maximum values of $z = (z_1, ..., z_m)^T$ for each objective function is set to ∞ and $-\infty$.

(ii) Update: The algorithm starts to iterate N times in a loop structure to update. Initially randomly selected two indexes of $B(i)$ are used to generate a new solution by the help of genetic operators. Then an improvement mechanism runs and outputs a new solution which is used to calculate the objective function in order to update the z values. Later on, the following inequality is checked whether it is satisfied or not. If it is satisfied, the solution set and the fitness values are updated.

$$g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$$

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 dominate $F(y')$.

(iii) Stopping Criteria: Finally, if the termination condition is satisfied, the algorithm stops and outputs the EP.

During initialization step, $B(i)$ contains the indexes of the T closest vectors of λ^i. Euclidean distance measurement methods is employed in order to compute the closeness between any two weigth vectors. 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 [1].

4.2. Pros and Cons of MOEA/D

- MOEA/D provides a simple yet efficient way of introducing decomposition approaches into multiobjective evolutionary computation.
- Since MOEA/D optimizes scalar optimization problems rather than directly solving a MOP as a whole, issues such as fitness assignment and diversity maintenance that cause difficulties for nondecomposition MOEAS could become easier to handle in the framework of MOEA/D.
- MOEA/D has lower computational complexity at each generation than NSGA-II [19] and MOGLS. Overall, MOEA/D outperforms, in terms of solution quality, MOGLS on 0-1 multiobjective knapsack 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. MOEA/D with an advanced decomposition approach performs much better than NSGA-II on 3-objective continuous test instances. MOEA/D using a small population is able to produce a small number of very evenly distributed solutions.
- Objective normalization techniques can be incorporated into MOEA/D for dealing with disparately scaled objectives.
- It is very natural to use scalar optimization methods in MOEA/D since each solution is associated with a scalar optimization problem. In contrast, one of the major shortcomings of nondecomposition MOEAs is that there is no easy way for them to take the advantage of scalar optimization methods.

4.3. Critics on MOEA/D

The algorithm employed in the thesis work is based on MOEA/D. So all of its either strong or weak features directly pass to MOEA/D-DE algorithm. That is why it is quite important to discuss some properties of MOEA/D regarding its diversity, complexity or solution range issue.

4.3.1. Why a Finite Number of Subproblems?

Since the computational resource is always limited, optimizing all the possible aggregation functions would not be very practical. MOEA/D spends about the same amount of effort on each of the N aggregation functions. Thus, the weight vector used in MOEA/D is N preselected vector.

4.3.2. How Diversity is Maintained in MOEA/D

Without decomposition, algorithms like NSGA-II 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 these algorithms. In MOEA/D, a MOP is decomposed into a number of scalar optimization subproblems. Different solutions in the current population are associated with different subproblems. The "diversity" among these subproblems will naturally lead to diversity in the population. When the decomposition method and the weight vectors are properly chosen, and thus the optimal solutions to the resultant subproblems are evenly distributed along the PF.

4.3.3. The Role of T in MOEA/D

T is the size of the neighborhood. Only current solutions to the T closest neighbors of a subproblem are used for optimizing it in MOEA/D. In a sense, two solutions have a chance to mate only when they are for two neighboring subproblems. So extra attention should be paid by setting T. If T is too small, the chosen parent solutions may be very similar subproblems, consequently, generated child solution could be very close to their parents. Therefore, the algorithm lacks the ability to explore new areas in the search space. On the other hand, if T is too large, the chosen parents may be poor for the subproblem and so is their child solution. Beside, a too large T will increase the computational overhead.

4.3.4. Complexity Computation of MOEA/D

The major computational costs are in MOEA/D generates N trial solutions. Updating mechanism performs $O(m)$ 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. Therefore, the computational complexity of Step 2 in the above variant of MOEA/D is $O(mNT)$ since it has N passes [1].

5. ENHANCING MOEA/D BY USING SOME NOVEL TECHNIQUES

The thesis works basically employs a multiobjective evolutionary algorithm based on decomposition (MOEA/D) and its extended version by using differential evolution (DE) as the main search engine (MOEA/D-DE) which outperform several widely used multiobjective evolutionary algorithms. MOEA/D 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 thesis work, two mechanisms are investigated to enhance the performance of MOEA/D-DE. Firstly MOEA/D algorithm code has been modified and simplified in order to make it work faster. Secondly, 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. Thirdly, the scaling factor in DE is randomized to enhance the search ability. 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. Beside that, different normalization methods for the objective functions have been implemented and the best one has been chosen. Then, different decomposition techniques have been performed to find the best one. Moreover, DE search algorithm has been included instead of polynomial mutation of the MOEA/D and the scaling factor in DE is randomly picked 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.

Most MOEAs compare solutions based on dominance. However, domination cannot 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)

[19] and strength Pareto evolutionary algorithm 2 (SPEA2) [20] 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) [1], was proposed. It decomposes a MOP into a set of scalar optimization sub-problems with neighborhood relations. The first version of MOEA/D uses simulated binary crossover (SBX) and polynomial mutation as the search engines. Later, a new version using the mutation (DE/best/1/bin [13]) in differential evolution (DE) as the main search engine was proposed and shown to outperform MOEA/D and NSGA-II, especially for complex problems.

There are several possibilities to enhance the performance of the MOEA/D-DE framework. The first one is 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, we rank the parent solutions and first replace those that are closer to the child solution. The second one is to enhance the search ability. We randomize the scaling factor in the DE mutation to achieve this.

5.1. Enhanced Parts of MOEA/D

5.1.1. Boosted MOEA/D

As initial work, the MOEA/D code has been boosted by modifying its code structure. First of all, it has been realized that the code was composed of lots of structs that slows down it seriously. Therefore, all structs have been eliminated and all parameters were taken out of the structs. Another improvement was that overall algorithm has been transformed into a script file instead of keeping it work with functions. After all, these improvements enhanced the performance of the code. The result is summarized in the Table 5.1.

Table 5.1. Speed Test of enhanced and original MOEA/D code.

Codes	Time(sec)
Nonenhanced	0.41
Enhanced	0.26

5.1.2. Novel Method to Generate Weight Vector

Weight initialization has the foremost importance in terms of solution range. By assigning fairly spread weights to each objective makes the solution space span towards both extreme points. Otherwise the solution might get stuck around a limited solution space. Search in different directions according to different weight vectors can naturally help the diversity. For this purpose novel weigth matrix initialization method has been proposed. Weight vectors are used as a method of decomposition of different subproblems into a single subproblem. Let $\lambda = (\lambda_1, ..., \lambda_m)^T$ be a weight vector providing $\lambda_i \geq 0$ and m keeps the index of the objective functions. In order to initialize the weight matrix properly, equation 5.1 must be satisfied.

$$\sum_{i=1}^{m} \lambda_i = 1 \tag{5.1}$$

By assigning these weights makes the objective functions have either lower or higher possibility to be optimized. In [1] weight initialization is just done for 2 objective optimization problems. The rule behind the initialization method for N objectives is :

$$\lambda^a = [1 - (a-1)/(N-1), (a-1)/(N-1)] \tag{5.2}$$

where $1 \leq a \leq N$. By applying the above formula, the extreme points get the the following values.

$$\lambda^1 = [1, 0]$$
$$\lambda^N = [0, 1]$$

Since this method provides the best distribution quality, it is employed in the proposed work for 2 objectives case. When it comes to the matter of multi-objective case, orthogonal genetic algorithm is implemented [3].

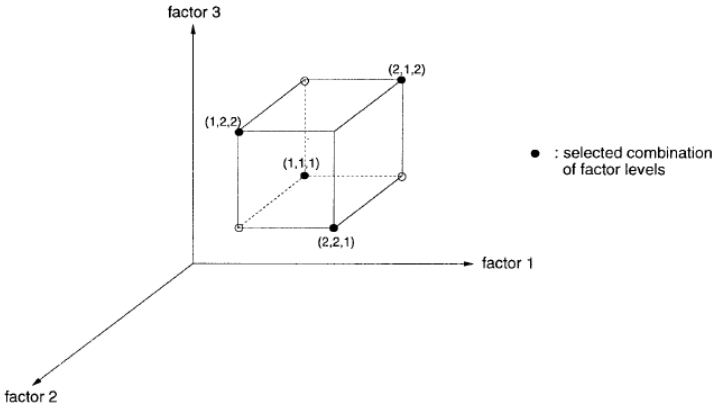

Figure 5.1. Orthogonality of the orthogonal array $L_4(2^3)$ [3].

Table 5.2. Orthogonal Array $L_4(2^3)$ for 3 factors at 2 levels; there are 4 combinations of factor levels.

Combination	Factor1	Factor2	Factor3
1^{st}	1	1	1
2^{nd}	1	2	2
3^{rd}	2	1	2
4^{th}	2	2	1

Figure 5.1 illustrates the meaning of orthogonality for $L_4(2^3)$. The orthogonal array tries to generate fairly distributed vectors in the whole space. In the table 5.2, each row represents a combination of levels. The orthogonality of an array means that for the factor in any column, every level occurs the same number of times and for the two factors in any two columns, every combination of two levels occurs the same number of times and the selected combinations are uniformly distributed over the whole space of all the possible combinations.

The level number represents the degree of importance on each objectives. The bigger number corresponds that it has more chance to be optimized. The 2^{nd} row of the Table 5.2 has $1, 2, 2$. Referring to equation 5.1, the sum must be equal to 1. So

it must be normalized as $0.2, 0.4, 0.4$. This indicates us that 2^{nd} and 3^{rd} objectives have relatively higher chance to be optimized. 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 5.1 the combinations are decreased to 4, which means with a level of 2 and 4 objective functions, a solution is offered with 4 weight vectors, making the population size 4. Increasing the level number to 10 will increase the population size to 100 and it will create a larger search space and better distribution.

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. Let the number of levels be 9. Under these circumstances, an objective might have the vector which is [1 1 1 2 2 2 3 3 3]. This objective will get its highest weight value at 7^{th} and 8^{th} combination which is $3/9 = 0.33$. This is not enough for the optimization range of the objective in question since the average 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, in problems with more than two objectives, their weights are initialized manually and saved in the look-up tables. This manual method is created 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 orthogonal array algorithm is employed. For the remaining cases, the manually initialized weight matrix which supports up to seven objectives are exploited. If it is more than seven objectives, the rest are created by the orthogonal array method. Beside, the look-up table is for the population sizes up to 150. For the cases more than 150, the weight matrix is filled with an orthogonal array initialization. As a result the final version of the weight initialization might be a combination of a look-up table and orthogonal array method. In Table 5.3, four objectives optimization case is illustrated. Since the population size is less than 150, the values are merely from look-up table. If it is carefully examined, some significant rules can be seen easily. First of all, the first four rows indicate that each objective will be optimized as a single optimization

problems once. Since it has the highest weight value in the vector. Second, the fifth row tells that all objectives are weighted equally once. The rest of the rows are written by changing the significance level of the objectives by making sure that almost every possible value is assigned.

As an alternative way, the weight initialization is created by using Latin Hypercube Sampling (LHS) method which takes place in Matlab function library. LHSDESIGN generates a latin hypercube sample X containing N values on each of P variables. For each column, the N values are randomly distributed with one from each interval (0,1/N), (1/N,2/N), ..., (1-1/N,1), and they are randomly permuted.

$$X = \text{lhsdesign}(N, P)$$

Table 5.4 shows the distribution quality of the both methods. In other words, how far LHS's or orthogonal's extreme weights spanning corresponds directly to distribution range of the pareto front.

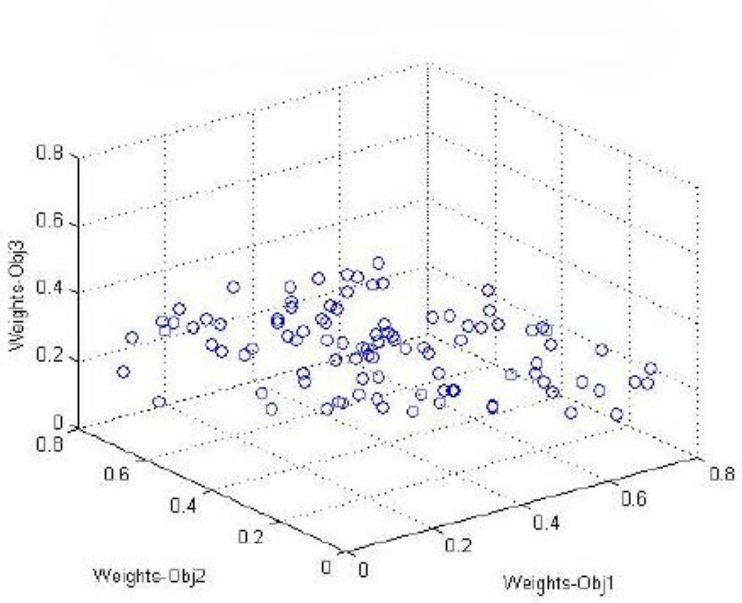

Figure 5.2. Latin Hypercube Sampling Weights.

Latin hypercube designs are useful when you need a sample that is random but that is guaranteed to be relatively uniformly distributed over each dimension.

For the sake of comparison, 10 tests with 500 iteration and 300 population size are done. Benchmark problems UF8, UF9 and UF10 are run twice for both methods. Then the three objective values are compared to find out how well one method is better for finding minimum value. Table 5.5 reveals us how many times each method performs better. For instance, for UF8 problem, LHS methods finds minimum 6 times out of 30 and the remaining 24 values are minimized by orthogonal array method. It is clearly seen that orthogonal method's solutions are better.

5.2. Comparison of Different Decomposition Methods

In chapter 2, details of both decomposition methods are presented. This section is dedicated to their performance comparison in terms of range, distribution and dominance. If we look at the weighted sum approach closer, it is seen that the optimization function g^{ws} will focus on optimizing the f_i with highest value.

$$\begin{gathered}\text{minimize } g^{ws}(\mathrm{x}\lambda) = \sum_{i=1}^{k} \lambda_i \cdot f_i \\ \text{subject to } x \in S, \\ .\end{gathered}$$

There is no minimum/maximum reference for the optimization algorithm to converge so lots of efforts have not been shown for this approach. In Tchebycheff Approach, 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 [1].

$$\text{minimize } g^{te}(x|\lambda,z^*) = \max(\lambda_i f_i(x)\text{- } z^*)$$

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. Before that, for PBI approach, proper θ value was determined. To do so, schotts metric has been used to compare the quality of the pareto front obtained by different θ penalty parameters. The θ values have been searched in a space 0-10. For the gain-gbw trade-off the best θ has been found to be 3,2.

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.

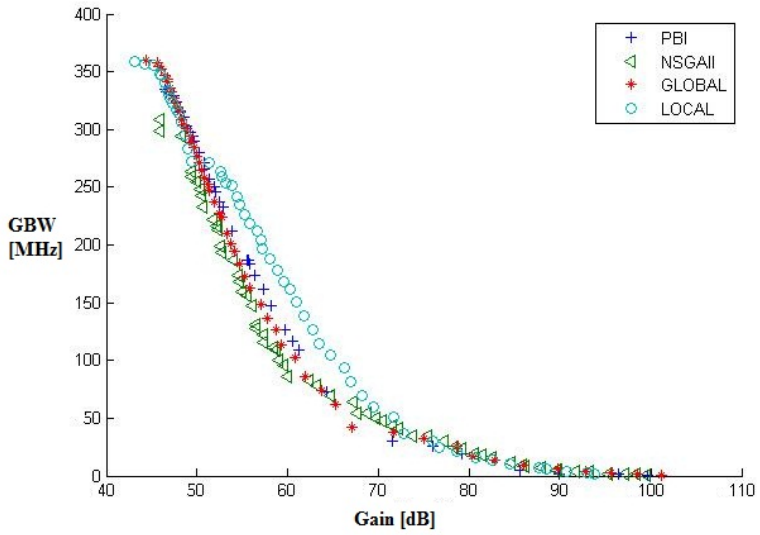

Figure 5.3. Comparisons of PBI with different TE methods.

Figure 5.3 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.

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 while 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.

5.3. Genetic Algorithms and Enhancing Searching Ability

Genetic algorithms are suitable for a wide variety of optimization problems. They present artificial implementations of the natural evolution of species. The optimization problem is described as a set of candidate solutions and an objective function that has to be maximized/minimized. The fitness function assigns a value to each candidate solution. The candidate with the highest fitness is the global optimum of the optimization problem. In order to find such a solution, a genetic algorithm produces a sequence of populations of candidate solutions. The generation of each successive population is a random process, guided by the fitness of the members of the previous population.

5.3.1. Genetic Operators

- Generate an initial population of candidate solutions, represented by chromosomes. This population is often generated at random, but it is also possible to include candidate solutions that are likely to have a high fitness.
- Evaluate each member of the initial population by means of the fitness function.
- Select one or more sets of parents (two population members) based on the fitness of the members of the population. The fitter members will have more chance to be selected than lesser fit members. For this selection, the fitnesses are scaled linearly, in a way that the lowest fitness equals to zero. The chance of a member to be selected as parent is than proportional to the scaled fitness of that member. The genetic operator cross-over is applied on the sets of parents in order to generate new candidate solutions. The ratio between the number of new candidates solution and the population size is called the cross-over rate. In order to keep

the population size constant when we insert the newly generated solutions in the population, an equal number of existing members is removed from the population. The filter members have more chance to be selected for the next generation than the lesser fit members. The newly generated candidate solutions fill in the empty places in the population.

5.3.2. Gaussian Mutation

The main reproduction operator in evolutionary strategies is Gaussian mutation, in which a random value from a Gaussian distribution is added to each element of an individual's vector to create a new offspring. In thesis work, two individuals are randomly selected from neighborhood set T and these two values are subjected to recombination. Then the created individual is sent to the Gaussian mutation operator. Mechanism works as seen in 5.4 and new individual is created. Mutation shows its main advantageous on diversity which is quite important parameter.

$$f(x) = \frac{\sum_{i-1}^{N} x_i^2}{N} \tag{5.3}$$

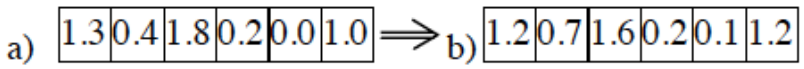

Figure 5.4. Gaussian Mutation of Parent a to form Offspring b.

5.3.3. Crossover Operator

In genetic algorithms, the most popular searching operator is crossover operator. The background work, MOEA/D, uses simulated binary crossover (SBX) and polynomial mutation as the search engines. Later, in the thesis work, a new version using the mutation (DE/best/1/bin) in differential evolution (DE) as the main search engine was proposed and shown to outperform MOEA/D. The search power of a crossover opera-

tor is defined here as a measure of how flexible the operator is to create an arbitrary point in the search space. The action of crossover is to create two new children strings from two parent strings. For instance, in a single-point crossover, a random cross site along the length of the string is chosen and the bits on one side of the cross site are swapped between two parent strings. Further, it is observed that the children points may lie inside or outside the region bounded by the parent strings depending on the strings and the location of the cross site. In order to define the spread of the children points with respect to that of the parents strings, we define a spread facto β as the ratio of the spread of children points to that of the parents points as seen in Equation 5.4.

$$\beta = |\frac{c_1 - c_2}{p_1 - p_2}| \tag{5.4}$$

Another self-adaptive crossover is the Simulated Binary Crossover (SBX). The SBX crossover puts the stress on generating offspring near the parents. So, the crossover guarantees that the extent of the children is proportional to the extent of the parents, and also favors that near parent individuals are more likely to be chosen as children than individuals distant from the parents [21].

5.3.4. DE Mechanism and Its Contribution to Searching

DE is a simple, but powerful algorithm that simulates natural evolution combined with a mechanism to generate multiple search directions based on the distribution of solutions (vectors) in the current population. Each vector i, $i = 1, ..., NP$ in the population at generation g, $x_{i,g} = [x_{1,i,g}, ..., x_{n,i,g}]^T$, called at the moment of reproduction as the target vector, will be able to generate one offspring, called trial vector $u_{i,g}$. This trial vector is generated as follows: First of all, a search direction is defined by calculating a difference vector between a pair of vectors x^{r1} and x^{r2}, both of them chosen at random from the population. This difference vector is also scaled by using a user-defined parameter called scale factor $F > 0$. This scaled difference vector is then added to a third vector $x_{r_0,g}$, called base vector. As a result, a new vector is obtained, known as the mutant vector. After that, this mutant vector is recombined, based on

a user-defined parameter, called crossover probability $0 \leq CR \leq 1$, with the target vector (also called parent vector) by using discrete recombination, usually uniform, i.e. binomial crossover, to generate a trial (child) vector. The CR value determines how similar the trial vector will be with respect to the mutant vector [4].

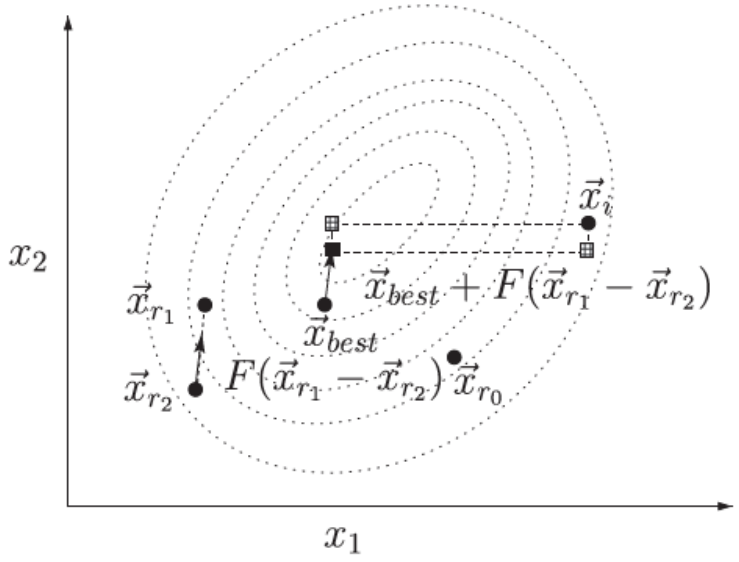

Figure 5.5. DE/best/1/bin graphical example [4].

In Figure 5.5, $\vec{x_i}$ is the target vector, $\vec{x_{best}}$ is the base vector (the best vector so far in the population), $\vec{x_{r_1}}$ and $\vec{x_{r_2}}$ (also chosen at random) are used to generate the difference vector as to define a second search direction. The black square represents the mutant vector, which can be the location of the trial vector generated after performing recombination. The two filled squares represent the other two possible locations for the trial vector after recombination. [4].

MOEA/D-DE uses the $DE/best/1/bin$ mutation, which is as follows:

$$y^i = x^i(t) + F(\vec{x_{r_1}}(t) - \vec{x_{r_2}}(t)) \tag{5.5}$$

The general convention used above is DE/x/y/z. DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z stands for the type of crossover being used (exp: exponential; bin: binomial). the vector to be perturbed is the best per-

forming vector of the current generation. In order to increase the potential diversity, crossover is introduced. In the thesis work this version of DE is exploited instead of using the mutation method of the MOEAD/D.

In equation 5.5, r_1 and r_2 (r_1 and $r_2 \in P$) are randomly chosen and mutually different, and also different from the current index i. $F \in (0, 1]$ is a constant called the scaling factor, which controls the amplification of the differential variation $\vec{x_{r_1}}(t) - \vec{x_{r_2}}(t)$. Most of the population-based search algorithms try to balance between two contradictory aspects of their performance: exploration and exploitation. The first one means the ability of the algorithm to "explore" or search every region of the feasible search space while the second denotes the ability to converge to the near-optimal solutions as quickly as possible.

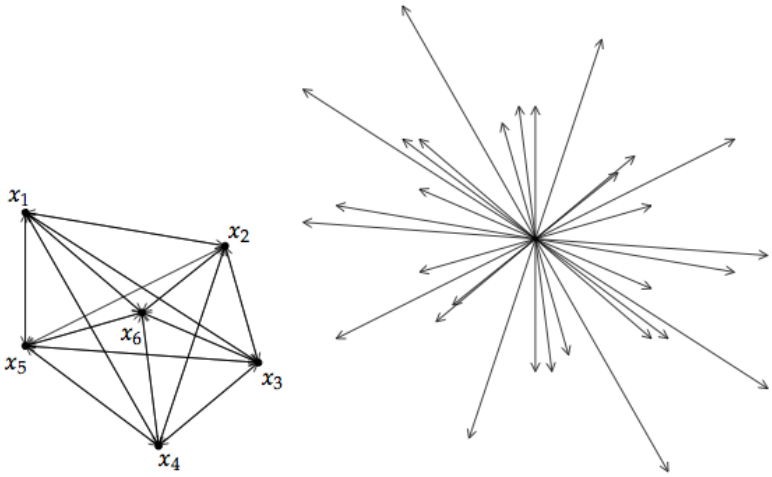

Figure 5.6. Difference vectors and their distribution for a population of six points.

Figure 5.6 shows the distribution of difference vectors that can be constructed from a population of six points. The vectors have all been re-arranged around one point for a clearer view of the points that can potentially be reached when applying a difference vector to one point. An important fact that can be discovered by examining Figure 5.6b is that for every point in the population, there is only a limited number of points from the search space that can be reached by applying such a mutation scheme. The DE variant known as $DE/best/1/bin$ uses the best vector of the population to generate donor vectors. By "best" we mean the vector that corresponds to the best fitness (e.g., the lowest objective function value for a minimization problem) in the entire population at a particular generation. The scheme promotes exploitation since

all the vectors/genomes are attracted towards the same best position (pointed to by the best vector) on the fitness landscape through iterations, thereby converging faster to that point. In addition, DE employs a greedy selection strategy (the better between the target and the trial vectors is selected) and uses a fixed scale factor F. Thus if the difference vector $x^{r1}(t) - x^{r2}(t)$, used for perturbation is small (this is usually the case when the vectors come very close to each other and the population converges to a small domain), the vectors may not be able to explore any better region of the search space, thereby finding it difficult to escape large plateaus or suboptimal peaks/valleys. In this scheme the weight factor of each vector is made to vary as a uniformly distributed random number from 0.5 to 1 is shown to have higher successful rate to reach the global optimization point in single objective problems. In thesis work, Gaussian distributed random scaling factor with mean value μ and variance σ : $F_{i,k} = norm(\mu, \sigma)$, $i = 1, ..., N$ and $k = 1, ..., n$. For each variable in the search space, the scaling factor $F_{i,k}$ of each differential variation $x^{r1}(t) - x^{r2}(t)$ is different. F is continuously generated randomly in each iteration.

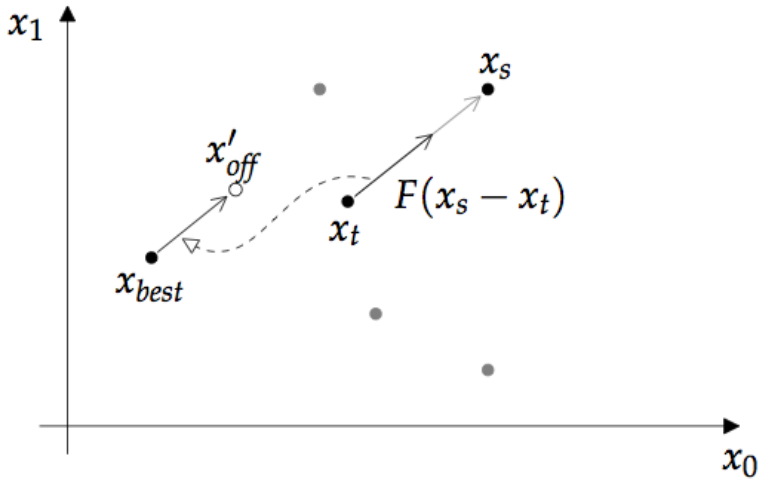

Figure 5.7. The scheme uses the best performance as the base point.

Figure 5.7 illustrates where offsprings are generated in the search space. The scheme using the individual presenting the best performance as the base point, such as DE/best/1/bin. These scheme tend to generate the provisional offsprings around the best individuals, which is also one characteristic of Swarm Intelligence algorithms.

$$
\begin{aligned}
F &= (f_1(x,y), f_2(x,y), f_3(x,y), f_4(x,y)) \\
f_1(x,y) &= \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3 \\
f_2(x,y) &= \frac{(x+y-3)^2}{175} + \frac{(2\cdot y - x)^2}{17} - 13 \\
f_3(x,y) &= \frac{(3\cdot x - 2\cdot y + 4)^2}{8} + \frac{(x-y+1)^2}{27} + 15 \\
f_4(x,y) &= \frac{(y-x-3)^2}{16} + \frac{(x-y+2)^2}{8} + 5
\end{aligned}
\tag{5.6}
$$

Table 5.3. Illustration of the weight matrix for the 4 objective case.

Objective1	Objective2	Objective3	Objective4
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

Table 5.4. Max value comparison of LHS and Orthogonal Weight Initilization.

Obj No	Obj1	Obj2	Obj3	Obj4	Obj5
LHS 3obj	0.7579	0.7983	0.7663	-	-
LHS 4obj	0.6242	0.6775	0.7228	0.7179	-
LHS 5obj	0.4267	0.4649	0.5774	0.4749	0.4875
Orthogonal 4obj	0.6667	0.5833	0.5625	0.6429	-
Orthogonal 5obj	0.4615	0.4545	0.5263	0.5625	0.4545

Table 5.5. Orthogonal vs LHS methods.

Problem	LHS	Orthogonal Array
UF8	6	24
UF9	17	13
UF10	15	15

Table 5.6. IGD comparison of different methods.

Mut	Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Average
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**	**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
Poly Mut.	0.0028	0.0017	0.0021	0.0034	0.0045	0.0022	0.0038	0.0031	0.0025	0.0024	0.0029
NSGAII	0.0053	0.0122	0.0089	0.0035	0.0089	0.0056	0.0081	0.0068	0.0070	0.0095	0.0076

6. IMPLEMENTATION OF ALGORITHMS ON BENCHMARK AND ANALOG CIRCUIT PROBLEMS

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 empirical 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) [22] and a real world problem, sizing of folded-cascode amplifier (4 objectives) are performed and 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. 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 [22], 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 0.1N, , 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 6.1. The IGD statistics based in 20 runs.

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, 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 6.1 shows the mean values of IGD results for each problem in 20 runs. The runs with smallest IGD values are drawn in Figure 6.2 and 6.3.

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/DDE 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

Table 6.2. Ranking of the IGD Values.

Tests	FRD	FD	RD	OD
UF1	Rank4	Rank3	Rank1	Rank2
UF2	Rank2	Rank1	Rank3	Rank4
UF3	Rank4	Rank3	Rank1	Rank2
UF4	Rank1	Rank2	Rank4	Rank3
UF5	Rank1	Rank2	Rank3	Rank4
UF6	Rank1	Rank4	Rank2	Rank3
UF7	Rank3	Rank4	Rank2	Rank1
UF8	Rank4	Rank3	Rank2	Rank1
UF9	Rank3	Rank2	Rank1	Rank4
UF10	Rank1	Rank3	Rank2	Rank4
Analog	Rank1	Rank2	Rank3	Rank4

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.

6.1. Folded Cascode Amplifier Design

The multiobjective analog sizing is optimization of a folded-cascode amplifier in Figure 6.1, 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 have a range of 0.24μm to 100μm , 5 of which have a range of 0.18μm to 10μm and 1 of which has a range of 1μA to 2.5μA.

For folded cascode sizing problem, the number of design variables is 11. The number of sub-problems is (population size) 148 (though 4 objectives, considering the computational effort, N is reduced to 148). T is set to $0.1N$, n_r is set to $0.01N$, 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 varience of 0.15. In GA operators, η and p_m are the same as MOEA/D-DE. The

Table 6.3. Ranking of the IGD Values.

Methods	Rank1	Rank2	Rank3	Rank4
FRD	5	1	2	3
FD	1	4	4	2
RD	3	4	3	1
OD	2	2	2	5

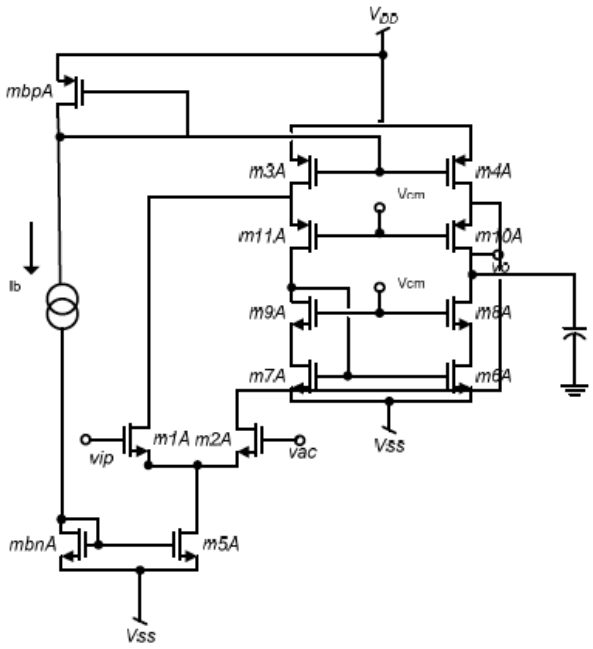

Figure 6.1. Schematic of CMOS Folded Cascode Amplifier.

algorithm stops after 200 iterations.

During the W-L optimization of the folded cascode amplifier there are 11 values to be optimized. W_1, W_3, W_5, W_8, W_{10}, L_1, L_3, L_5, L_8, L_{10} and I_b. The I_b value has been searched in a range 0.5μA to 2.5mA. The W and L values are searched in the range according to Table 6.5.

After the optimization has reached the iteration limit, it terminates. Two optimizations have been run using both orthogonal array based method and look-up table

Table 6.4. IGD Values of Folded Cascode.

Test	FRD	FD	RD	OD
Folded	9.4572	9.5344	9.5199	9.6079

Table 6.5. W and L limits for Optimization.

Process	90nm CMOS	0.18μm CMOS	0.25μm CMOS
W_{min}	120nm	0.24μm	0.36μm
W_{max}	200μm	800μm	800μm
L_{min}	90nm	0.18μm	0.25μm
L_{max}	5μm	10μm	20μm

based (LUT) method. 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. Table 6.6 summarizes the performances of both methods.

Table 6.6. Objective Function Ranges.

Obj. Func	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

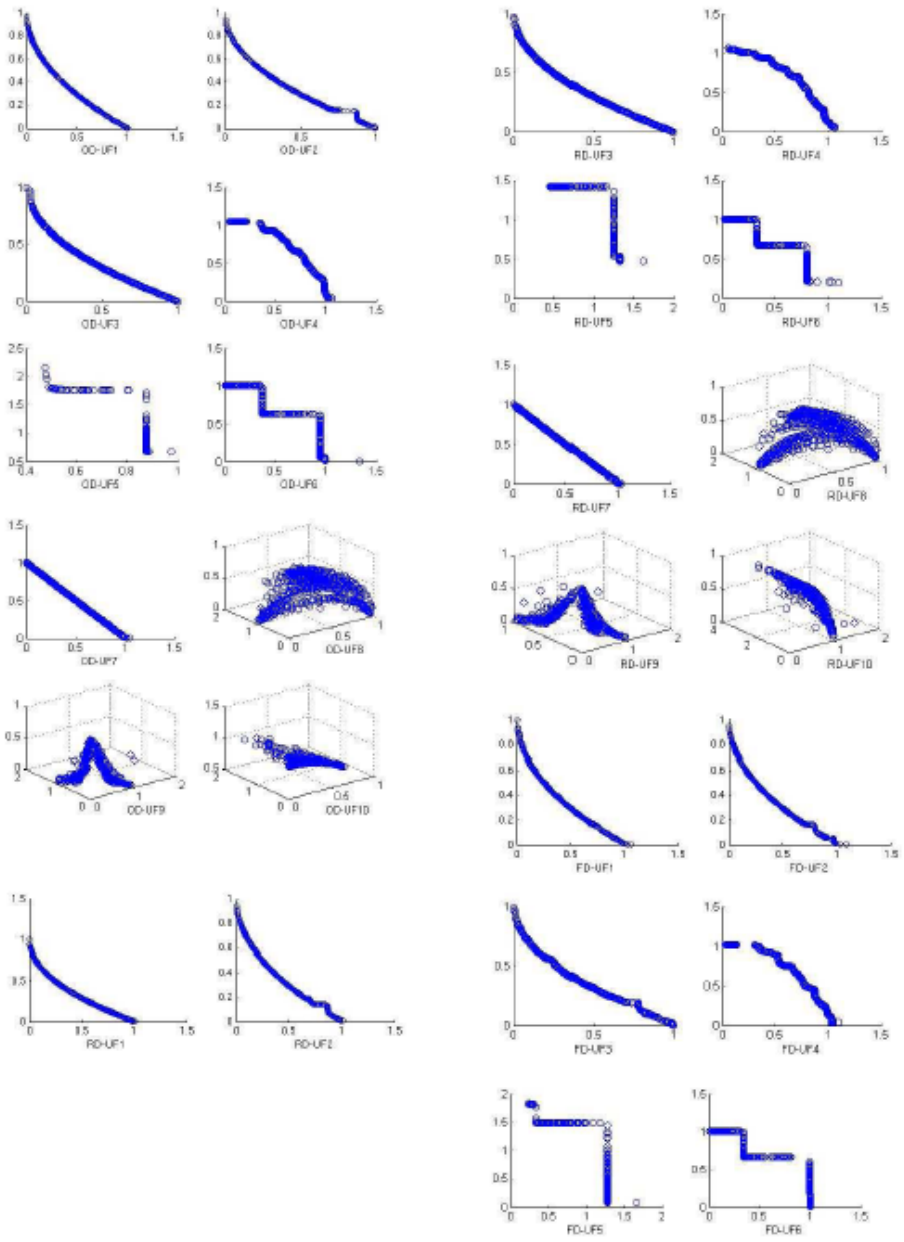

Figure 6.2. PF with the smallest IGD values by different methods.

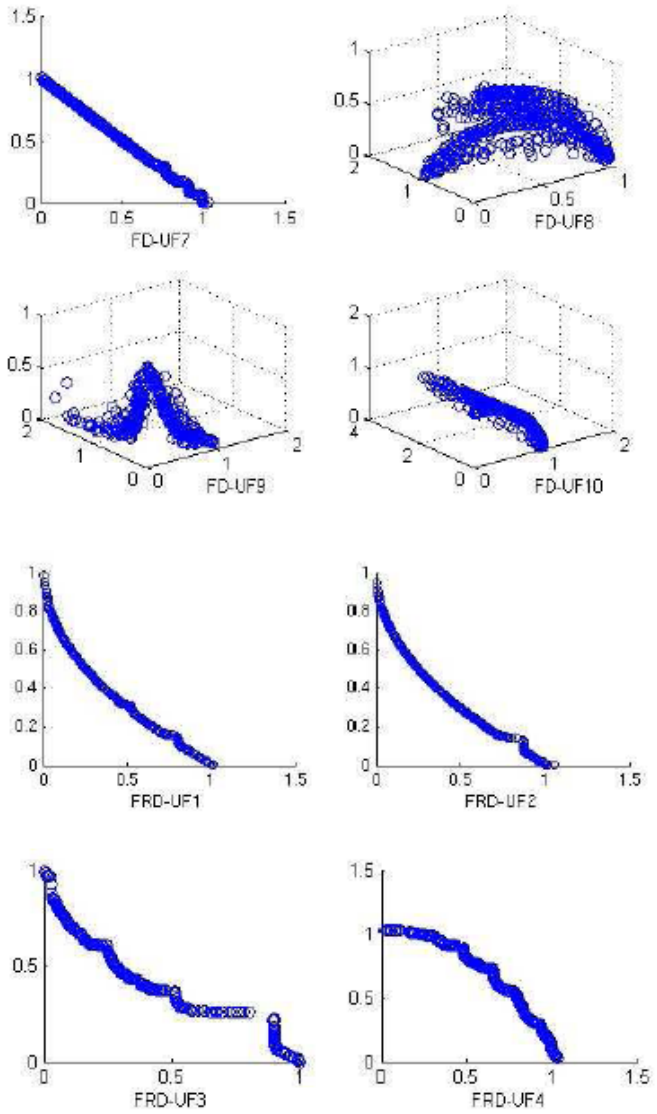

Figure 6.3. PF with the smallest IGD values by different methods.

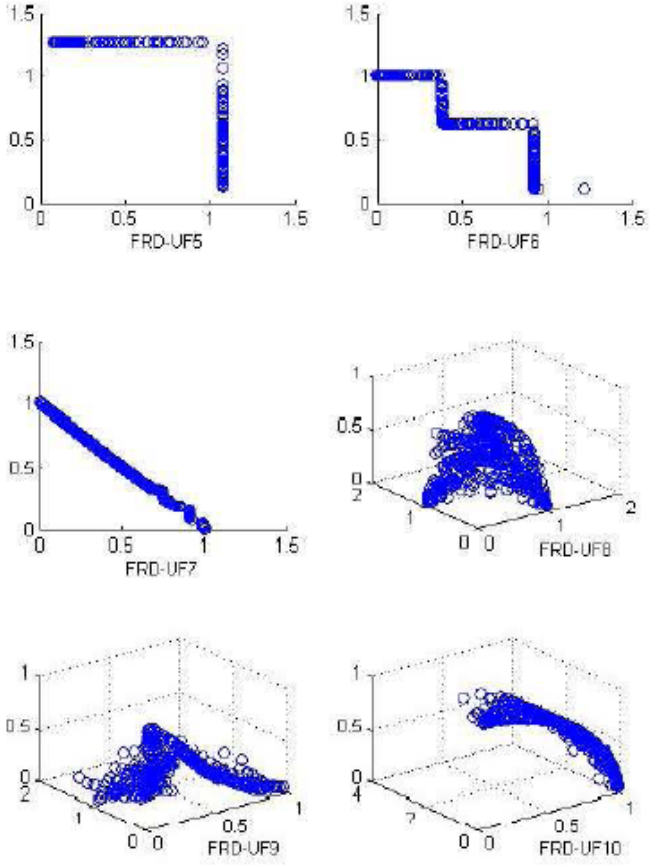

Figure 6.4. PF with the smallest IGD values by different methods.

7. CIRCUIT OPTIMIZATION BY CONSIDERING LAYOUT PARASITICS

7.1. Creating Circuit Design Loop

Another optimization algorithm called SACSES [6] is also quite successful regarding speed of convergence and solution range. Especially, adaptive weight mechanism of the objective functions makes this algorithm unique. Beside, its cost function outputs a single optimal solution instead of pareto optimal set. Hence, it is easier to link it to layout generator [5]. But there is a need for interface for communication between them. TOLAS creates *.gds* file which is recognized by Calibre to extract. After extraction, 3 types of files come out which are *.netlist*, *.pex*, *.pxi*. These have transistors and parasitic RC elements and compatible with HSPICE. But SACSES uses its own simulator SPASE and it requires numerically named nodes contrary to alphanumeric ones. So, a source code is created to combine all three files into one by renaming all nodes. Also, SACSES requires specific file that includes input parameters. This file is also created inside the source code. Finally, the automized feedback loop is created between three program, SACSES, TOLAS and Calibre.

7.2. BTS OPAMP

The circuit schematic of the synthesized BTS OPAMP is given in figure 7.1. Since this is an IC implementation, the first stage biases the second stage in addition to providing some gain. Also, there is no common-mode feedback to adjust the output offset voltage. These limitations make the design of a BTS OPAMP with low output offset voltage, high gain and low output impedance more difficult than it may seem. The number of independent variables is 22.

Table 7.1 shows the performance of hand designed BTS circuit, without parasitic effects. The main goal is to get the similar results after the layout parasitics are in-

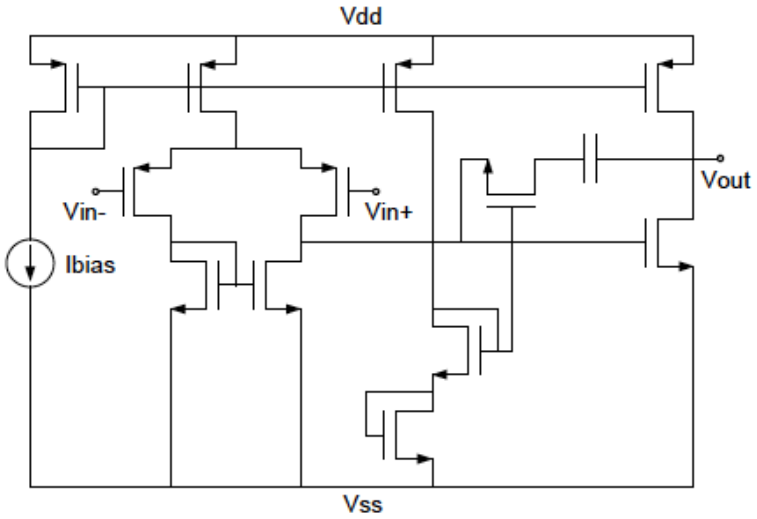

Figure 7.1. Schematic of a basic two stage CMOS operational amplifier.

cluded.

Table 7.1. Specifications of the synthesized BTS OPAMP.

Specifications	HSPICE
A_0(dB)	88
BW(kHz)	5.64
R_{out}(kΩ)	9.95
$V_{os}(mV)$	19.91
PM(deg)	54.8
Power(mW)	13.5

7.3. Time Constant Equilibration Reduction (TICER)

Time Constant Equilibration Reduction (TICER) is a RC reduction method for extraction tools. The time-constant of a node is the total capacitance from the node to other nodes and to ground divided by the sum of conductances from the node to other nodes and ground. Let's consider a RC network for reduction by preserving its behavior within a certain frequency range. To do so, each node of a circuit needs to be classified into one of three categories. This classification depends upon the node's time-constants.

Nodes are referred as quick, slow or normal by defining the frequency range of interest [23]. This classification makes it possible to eliminate quick and slow nodes from the network without significantly altering its behavior at least not in the frequency range of interest. This is so called "Time Constant Equilibration Reduction" abbreviated as TICER. After reduction, the remaining circuit is realizable. That makes TICER pretty desirable. In contrast to alternative methods, TICER focuses on a particular node and its neighbors, temporarily regards these as an multiport, eliminates the internal node, then returns to a flattened view of the circuit before selecting another node for elimination.

Calibre xRC has its own reduction method inside. It can be enabled either from options menu or directly by modifying xRC runset file. For the thesis work, two reduction at 100 and 500 kHz are tried and compared according to their parasitic elements numbers. Table 7.2 reveals the number of parasitics after reduction and without reduction. When the circuit is simulated around the frequency of interest, the parasitic elements decreases.

Table 7.2. Reduction of Parasitic Elements by TICER.

Reduction Type	Number of parasitics
Original	2214
500 kHz	1283
100 kHz	741

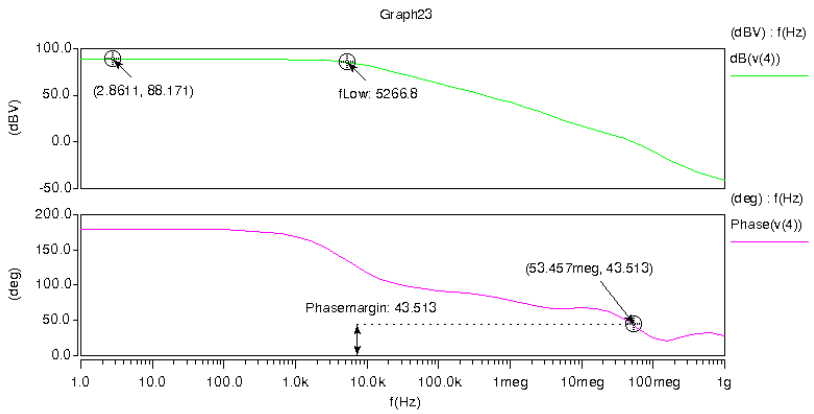

Figure 7.2. TICER with 500kHz.

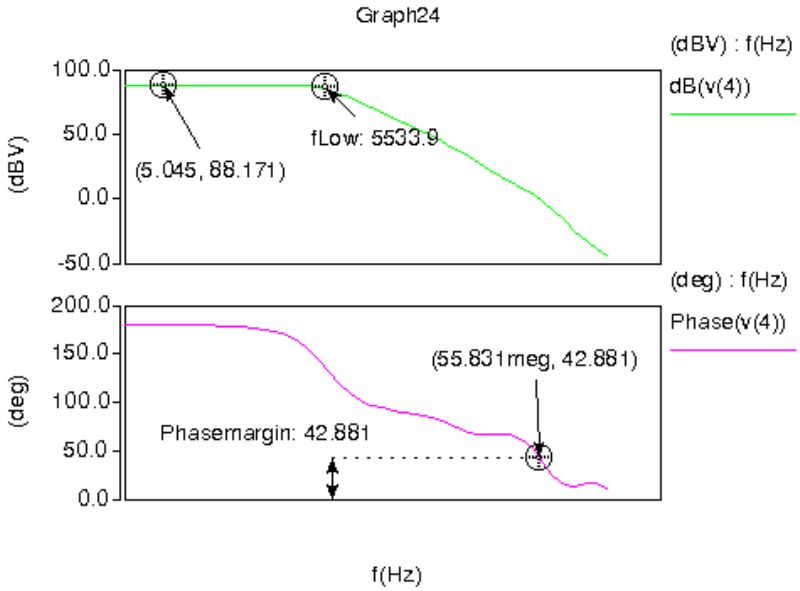

Figure 7.3. TICER with 100mHz.

The Table 7.3 summarizes how TICER affects performance of a circuit. Most of the objective functions get better. These objectives will be fed into SACSES for optimization and all performances will be drawn to the point as targeted. The SACSES' output results will be presented in following chapters.

Table 7.3. TICER Effect on Performance.

Performance Metrics	w/o TICER	500mhz	100mhz
Gain(dB)	87.4	88.17	88.17
PM(deg)	46.8	43.5	42.8
Unity Gain(mHz)	46.8	53.45	55.83
BW(kHz)	5.37	5.26	5.53
Vos(mV)	147	-33.8	-33.8
Power(mW)	13.57	13.46	13.46

7.4. Template Based Layout Generation

After circuit optimization, netlist file is created. Then the netlist taken from SACSES is used by layout level DA tool, TOLAS (Tool Oriented Layout Automation System), which generates layout using the netlist. TOLAS is coded in JAVA and it consists of tools and databases. The block diagram as seen in figure 7.4 shows the template based layout generation. In this kind of implementation, layout database deals with the layout in a hierarchical structure, device database keeps track of the devices and their ports, and process rules are stored in design and electrical rule databases. The "device generator" constructs devices for the particular technology in our case AMS 0.35μm technology. The "Advanced Transistor Generator" builds transistors and handle the routing of fingered transistors. As final step, the GDSII Exporter writes the layout database into a file. The main tool, Template Based Layout Synthesizer, constructs a layout based on an input template using linear programming. The template file is written in Layout Description Script (LDS), which defines constraints between devices as well as routing.

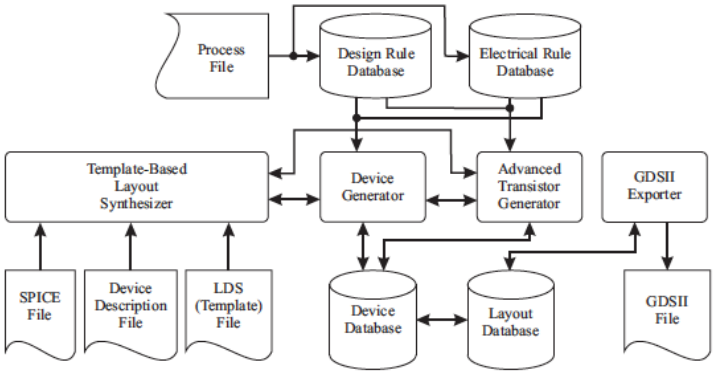

Figure 7.4. Block Diagram of the Synthesizer [5].

As seen in Figure 7.4, the tool needs three files which are spice, device description and LDS files. The "spice file" contains the netlist of the circuit and the "device description file" contains the detailed device information such as: number of fingers for the transistors, information about merged devices, etc. The template is stored in the LDS file and it is composed of statements about placement and routing [5]. The GDSII file is compatible with the commercial extractor Calibre. In the thesis work,

TOLAS' layout is directly fed into Calibre and extracted netlist are taken as required for the automation loop.

7.5. Flattening Extraction Output Files

The layout file created by TOLAS is fed to the extractor to get the parasitic effects. Calibre is exploited for that purpose. It is adjusted to output extraction files in accordance with Hspice. There are three kinds of circuit files that call for each other nestedly. The parasitic resistors and capacitors are in the form of subcircuits. Also, the transistors are fingered after extraction. The file extensions of the extraction files are .pxi, .pex and .netlist. Pex files includes subcircuit definitions as seen in figure 7.5.

```
.subckt PM_37005_VIN1 52 70 81 88 95 102 109 116 123 130 138
c4 52 0 12.1743f
r5 70 130 147.083
r6 70 138 147.254
r7 67 123 147.083
r8 67 70 0.511579
r9 64 116 147.083
r10 64 67 0.170526
r11 61 109 147.083
r12 61 64 0.511579
r13 58 102 147.083
r14 58 61 0.170526
r15 55 95 147.083
r16 55 58 0.511579
r17 52 88 147.083
r18 52 55 0.170526
r19 52 81 147.595
.ends
```

Figure 7.5. A subcircuit in .pex file.

These subcircuits are called in .pxi files in a way seen in figure 7.6

```
x_PM_37005_VIN1 N_VIN1_c_4_p N_VIN1_c_1_p N_VIN1_M1_g N_VIN1_M1@2_g
+ N_VIN1_M1@3_g N_VIN1_M1@4_g N_VIN1_M1@5_g N_VIN1_M1@6_g N_VIN1_M1@7_g
+ N_VIN1_M1@8_g N_VIN1_M1@9_g PM_37005_VIN1
```

Figure 7.6. Subcircuit call in .pxi file.

The .netlist file includes fingered transistors and it calls .pxi file inside. If all

these files are examined, it will seen that names of the nodes include alphanumeric or symbolic characters which are not easy to deal with. Beside the names are too long and by considering the fingered transistors, there exists many of these nodes in the netlist.

On top of that, the in-house circuit simulator SPASE requires flat netlist to run the simulations. For all these reasons, a script code is created to remove all subcircuits, rename the long node names and put three files together into one file. Then DC and AC voltage and current sources, probes, parameters and some measurement commands are included in this flat file. After all, the flat file is ready to be run by SPASE or HSPICE.

The code firstly reads the .pxi file which includes subcircuit calls and coupling capacitors. The subcircuits starts with x are read one by one and copied into structures. Structs keep the names and nodes of the subcircuits which are then substituted with new ones by keeping the old names. Hence, the old names are then used to be searched within netlist file to share the new names with transistor nodes. By the way, some key words are searched in the alphanumeric node names to extract the voltage supply, ground and input nodes. Then fixed numbers are assigned to these fixed nodes like 0 for ground, 1 for supply, 2 for input signal. Another need for the flat netlist is the map showing which numbers corresponds to the original transistor nodes. This is required because SACSES needs to know the meanings of the new node names to optimize the right transistors. After optimization is done, the fingered transistors have to be combined and served as one transistor to TOLAS regarding its working capability. The following lines show the transistor name map. Right side of the arrows is the node names coming out of Calibre and left side is the assigned names by the flattening script code.

$$M3 \rightarrow m0$$

$$M12 \rightarrow m1$$

$$M4 \rightarrow m2$$

$$M12@2 \rightarrow m3$$

$$M4@2 \rightarrow m4$$
$$M12@3 \rightarrow m5$$
$$M3@2 \rightarrow m6$$
$$M12@4 \rightarrow m7$$
$$M12@5 \rightarrow m8$$

$M12@2$ and $M12@3$ corresponds to two different fingers of the $M12$ transistor. But the new names do not contain any information regarding which nodes belong to particular transistors. Hence, this map is quite useful for the optimization side.

7.6. Batch Mode Execution of Calibre

The one of the aims of the thesis is creating an automated optimized circuit design as in figure 7.7 considering parasitic effects. In order to run the system without any interruption, batch mode execution must be used. Batch mode requires some files prepared in advance. These are the circuit netlist whose extension is .src.net, GDSII (Graphic Database System) file and runset file. Runset file is program specific file that contains soma addresses and parameters. For instance the address of the GDSII file and circuit netlist or the address of the design library for the particular technology which is Austria Micro Systems (AMS) 0.35μm CMOS process for our case.

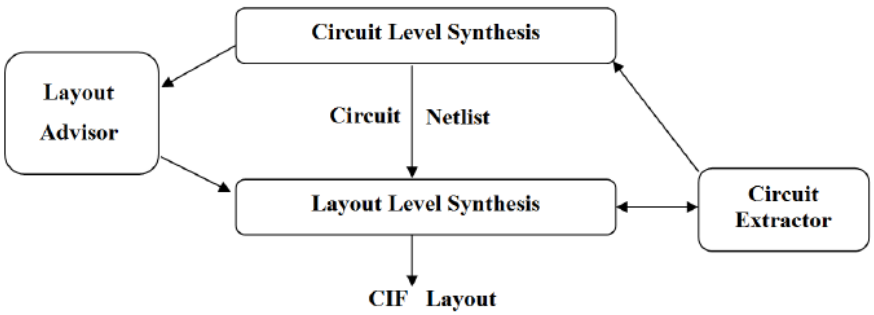

Figure 7.7. Block diagram of automated design loop [6].

The command to run the Calibre in batch mode is

calibre -gui -xrc [*runsetaddress*] -batch

The runset file is a key here because it has the significant information inside. The Figure in 7.8 shows these parameters in runset file.

```
*pexRulesFile: /staff/designkit370updated/calibre/c35b4/cac35b4rules_all.run
*pexRunDir: /mentor/sipahi/tryout
*pexLayoutPaths: 36931.calibre.db
*pexLayoutPrimary: 36931
*pexLayoutGetFromViewer: 1
*pexSourcePath: 36931.src.net
*pexSourcePrimary: 36931
*pexReportFile: 36931.lvs.report
*pexPexNetlistFile: 36931.pex.netlist
*pexPexNetlistFormat: HSPICE
*pexPexNetlistNameSource: LAYOUT
*pexPexReportFile: 36931.pex.report
*pexMaskDBFile: hw1.maskdb
*pexPexEnableTICER: 1
*pexPexTICERThreshold: 1000000
*cmnUseCBforRVE: 0
*cmnRunHier: 2
*cmnSlaveHosts: {use {}} {hostName {}} {cpuCount {}} {a32a64 {}} {rsh {}} {maxMem
{}} {workingDir {}} {layerDir {}} {macLibPath {}} {launchName {}}
```

Figure 7.8. Runset File.

The main parameters to modify are rule files path, source path, netlist format, circuit reduction threshold. The value 1 seen in the Figure 7.8 corresponds to bool type. It means enabling the option.

7.7. Layout Aware Circuit Optimization Test Results

After all files are rearranged for SACSES, the optimization is run with PC with 3.1GHz processor and it just uses one core effectively. The optimization takes about 4 hours in average. The following table reveals that after optimization almost all performance metrics have been improved.

SPASE results are different than the HSPICE depending on their capacitance modelling difference. That is why both simulator's results are given in the tables.

Table 7.4. Post Optimization Results with HSPICE.

Performance Metrics	Pre. Opt	Post Opt.
Gain(dB)	88.17	88.7
PM(deg)	42.8	47.1
Unity Gain(mHz)	55.83	56.12
BW(kHz)	5.53	5.07
Vos(mV)	-33.8	7.9
Power(mW)	13.46	12.4

Table 7.5. Pre and Post Optimization Results with SPASE.

Performance Metrics	Pre. Opt SPASE	Post Opt. SPASE
Gain(dB)	88.16	88.7
PM(deg)	54.8	59.3
BW(kHz)	5.7	5.17
V_{os}(mV)	-30.7	4.35
Power(mW)	13.6	12.7
R_{out}	9.43	9.7k
CMRR(dB)	8908	13969

After all, transient behaviour of the circuit is shown in Figure 7.9. All these results indicate that the main goal of the project is accomplished.

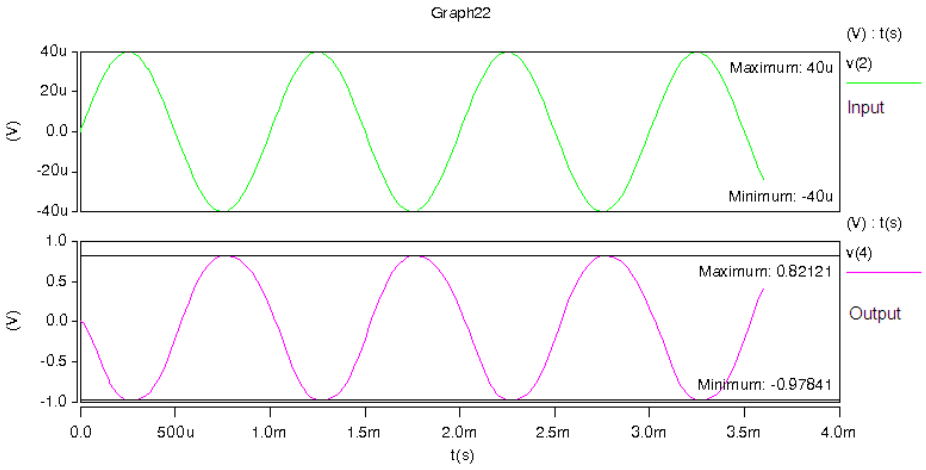

Figure 7.9. Post Optimization Transient Analysis.

8. CONCLUSION FUTURE WORK

A simulation-based analog circuit synthesis methodology and its implementation, MOEA/D-DE, was presented. With the use of simulation-based approach, MOEA/D-DE is topology independent and requires minimal initial effort. The algorithm uses HSPICE as simulator. 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. The method takes advantage of the fast initial convergence of the genetic algorithms. A background work based on the decomposition of the whole problem into different scalar problems has been chosen and several enhancements have been realized to improve that algorithm. First of all, some software enhancements 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. In the second part of the thesis work, the another analog circuit synthesizer (SACSES) and the layout generator (TOLAS) are used to create a feedback loop aimed to minimize the effects of inevitable layout parasitics was proposed and tested by synthesizing a BTS OPAMP circuit. Calibre is used as extractor and the extracted files are rearranged according to the needs of both SACSES and TOLAS. Because each program have their own specific interface both for inputs and outputs. RCC type of extraction is chosen to see the real effects of the parasitics. Also under Calibre options, TICER option is enabled to eliminate unnecessary nodes whose time constants are out of our interest. Two iterations through the loop were necessary in order to obtain a circuit within acceptable performance criteria.

8.1. Future Work

As future work, adaptive iteration number mechanism can be added to MOEA/D-DE algorithm for speeding up the whole process. Also, as in SACSES, the cost function can be added in order to output single point solution and this solution can be integrated with TOLAS again. Another point is better minimization of layout parasitics can help

decreasing the convergence time of the optimization algorithm.

Finally, it can be said that the MOEA/D-DE algorithm is an effective algorithm for optimizing the dimensions of the transistors on analog circuits. The results are then compared with the existing works present in the literature and sometimes it is observed that significant improvements are obtained.

REFERENCES

1. Zhang, Q., and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE*, Vol. 11, December 2007.

2. Robic, T., and B. Filipi, "Demo: Differential evolution for multiobjective optimization," *Springer*, Vol. 10, no. 3, pp. 520–533, 2005.

3. Zhang, Q., and Y.-W. Leung, "An orthogonal genetic algorithm for multimedia multicast routing," *IEEE*, Vol. 3, April 1999.

4. Mezura-Montes, E., M. E. Miranda-Varela, and R. del Carmen Gómez-Ramón, "Differential evolution in constrained numerical optimization: An empirical study," *IEEE Trans. on Computer Aided Design*, Vol. 27, July 2010.

5. Unutulmaz, A., G.Dündar, and F.Fernandez, "A template description script for layouts - lds," tech. report, Bogazici University, 2009.

6. Sonmez, O., *Circuit Level Analog Design Automation.* PhD thesis, Bogazici University, 2010.

7. Debyser, G., F. Leyn, K. Lampaert, J. Vandenbussche, G. Gielen, W. Sansen, P. Veselinovic, and D. Leenaerts, "Amgie- a synthesis environment for cmos analog integrated circuits," *IEEE*, Vol. 20, p. 1, September 2001.

8. Toumazou, C., and C. A. Makris, "Analog ic design automation: Part i-automated circuit generation: New concepts and methods," *IEEE*, Vol. 14, pp. 221–222, February 1995.

9. Xue, F., *Multi-Objective Differential Evolution Theory and Applications.* PhD thesis, Rensselaer Polytechnic Institute, Troy,New York, September 2004.

10. Coello, C. A., G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms*

for Solving Multi-Objective Problems, Springer, 2nd ed., 2007.

11. Fonseca, C. M., and P. J. Fleming, "Genetic algorithms for multiobjective optimization: formulation, discussion, and generalization," 1993.

12. Subbotin, S., and A. Oleynik, "The multi objective evolutionary feature selection," (Lviv-Slavsko, Ukraine), pp. 115–116, TCSET, February 2008.

13. Price, K., and R. Storn, "Differential evolution: A simple evolution strategy for fast optimization," *Dr. Dobb's Journal*, pp. 18–24, 1997.

14. Paquete, L., and T. Stützle, "A two-phase local search for the biobjective traveling salesman problem," pp. 479–493, Proc. Evol. Multi-Criterion Optim, 2003.

15. Miettinen, K., *Nonlinear Multiobjective Optimization*, Norwell, MA: Kluwer: Kluwer's International Series, forth ed., 1999.

16. Tiwari, S., G. Fadel, P. Koch, and K. Deb, "Performance assessment of the hybrid archive-based micro genetic algorithm (amga) on the cec09 test problems," *IEEE*, 2009.

17. Schaffer, J. D., "Multiple objective optimization with vector evaluated genetic algorithms," pp. 93–100, Proc. 1st Int. Conf. Genetic Algorithms, 1985.

18. Knowles, J. D., and D. W. Corne, "The pareto archived evolution strategy: A new baseline algorithm for multiobjective optimisation," (Washington, D.C.), pp. 98–105, in Proc. Congr. Evol. Comput, July 1999.

19. Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan, "Fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE*, Vol. 6, April 2002.

20. Zitzler, E., M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK) Department of Electrical Engineering Swiss Federal Institute

of Technology (ETH) Zurich, ETH Zentrum, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

21. Kalyanmoy Deb, H.-G. B., "Self-adaptive genetic algorithms with simulated binary crossover," Tech. Rep. CI-61/99, Department of Computer Science/XI University of Dortmund, 44221 Dortmund, Germany, March 1999.

22. Zhang, Q., A. Zhou, and S. Zhao, "Multiobjective optimization test instances for the cec 2009 sepcial session and competition," technical report ces-487, The Shool of Computer Science and Electronic Engineering, University of Essex, 2009.

23. Sheehan, B. N., M. Graphics, and Wilsonville, "Ticer realizable reduction of extracted rc circuits," *IEEE*, 1999.