AIDA-CMK: Multi-Algorithm Optimization Kernel applied to Analog IC Sizing

Ricardo Calado Correia Lourenço

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Nuno Cavaco Gomes Horta

Prof. Manuel Barros

Examination Committee

Chairperson: Prof. João Emílio Segurado Pavão Martins

Supervisor: Prof. Nuno Cavaco Gomes Horta

Members of the Committee: Prof. Rui Santos-Tavares

May 2014

Abstract

The rapid evolution and widespread of consumer electronic devices such as cell phones, tablets or Smart TV puts a great innovative pressure on the integrated circuit industry. Most of the functionalities of such devices are implemented using digital circuitry but analog tasks such as converting, receiving and emitting signals, regulating power or communicating to device sensors still play a major role in modern ICs, leading to mixed-signal systems. However, the design automation of analog circuits is far behind that of the digital circuits, either in techniques or in tools, failing to help designers to compete with the demanding time-to-market, lowering the costs and cutting development time. In this context AIDA Framework, an electronic design automation framework fully developed in-house, appears as an Electronic Design Automation tool to aid designers to do their job better and faster. This work focus is AIDA-CMK, by enhancing AIDA-C, which is the circuit optimizer component of AIDA, with a new multi-objective multi-constraint optimization module that construct a base for multiple algorithm implementations. In the proposed solution, three approaches to multi-objective multi-constraint optimization, namely, an evolutionary approach with NSGAII, a swarm intelligence approach with MOPSO and stochastic hill climbing approach with MOSA are implemented. Moreover, the implemented structure allows the easy hybridization between kernels transforming the previous simple NSGAII optimization module into a more evolved and versatile module supporting multiple single and multi-kernel algorithms. The three multi-objective optimization approaches were validated with CEC2009 benchmarks to constrained multi-objective optimization and tested with real analog IC design problems. The achieved results were compared in terms of performance, using statistical results obtained from multiple independent runs showing that NSGA-II outperforms the other two single kernel reference approaches by having a better convergence time, a widespread set of solutions, and, in general, achieving better Pareto fronts. Finally, some hybrid approaches were also experimented, giving a foretaste to a wide range of opportunities to explore in future work.

Keywords: Analog Integrated Circuit Design, AIDA Tool, Electronic Design Automation, Multi-Objective Multi-Constraint Optimization, NSGA-II, MOPSO, MOSA

Resumo

A rápida evolução e ampla difusão de dispositivos eletrónicos, como telemóveis, tablets ou SmartTV, coloca uma grande pressão na indústria de circuitos integrados. As funcionalidades nestes dispositivos são maioritariamente implementadas utilizando circuitos digitais, mas tarefas como a conversão, receção e emissão sinais, regulação de energia ou comunicação com os sensores do dispositivo ainda permanecem analógicos levando ao aparecimento de circuitos integrados mistos. A automação do desenho de circuitos analógicos é muito limitada, quando comparada com a dos circuitos digitais, tanto em técnicas como em ferramentas que não conseguem ajudar os projetistas a dar resposta ao exigente time-to-market de hoje, reduzindo os custos e tempo de projeto. É neste contexto que a plataforma AIDA aparece como uma ferramenta automação do projeto de circuitos integrados analógicos para ajudar os projetistas a fazerem o seu trabalho melhor e mais rápido. O foco desta dissertação é o desenvolvimento do AIDA-CMK, melhorando o optimizador de circuitos existente previamente na Framework, AIDA-C, através da implementação de vários algoritmos de otimização sobre uma nova estrutura que permite a fácil implementação futura de outros algoritmos. A solução apresentada neste trabalho implementa três das principais abordagens para o algoritmo de otimização multi-restrição multi-objetivo, ou seja, NSGAII, Particle Swarm e Simulated Annealing. Mais, a estrutura proposta para o módulo de otimização, também permite uma fácil hibridização entre algoritmos, transformando o módulo anterior, apenas com NSGAII, num modulo mais evoluído e versátil capaz de suportar múltiplos algoritmos e soluções hibridas. Os algoritmos implementados foram testados utilizando funções de teste de referência e circuitos. Os resultados obtidos foram comparados em termos de desempenho, utilizando-se os resultados estatísticos obtidos a partir de vários ensaios, mostrando que o NSGA-II supera as outras duas abordagens de referência por ter um melhor tempo de convergência, um conjunto amplo de soluções, e, em geral, consegue melhores frentes de Pareto. Além disso, algumas soluções hibridas foram abordadas, abrindo o caminho para explorar as oportunidades de implementação de novos algoritmos.

Palavras-chave: Desenho de circuitos integrados analógicos, Ferramenta AIDA, Automação de circuitos elétricos, Otimização multiobjectivo com restrições, NSGA-II, MOPSO, MOSA

Acknowledgments

I would like to acknowledge my supervisor Prof. Nuno Horta and co- supervisor Prof. Manuel Barros for all the support, trust and guidance since the first day, and also, the opportunity to develop my Master Thesis in the area of analog integrated circuit optimization in the Instituto de Telecomunicações. To the Integrated Circuits Group research team, Ricardo Póvoa, António Canelas, Ricardo Martins, Pedro Ventura, and Nuno Lourenço for their support in both AIDA Framework and the constant feedback on electronic matters.

Now, to my friends Pedro Pexirra, Nuno Nobre, Tiago Patrocinio, Joao Domingos, my companions in Instituto Superior Técnico and many others who helped me during my journey in Instituto Superior Técnico

To my family, my parents, brother and wife for motivation and support needed to develop this work.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADA Analog Design Automation

AMS Analog and Mixed-Signal

CAD Computer Aided Design

DSP Digital Signal Processing

EDA Electronic Design Automation

GA Genetic Algorithm

IC Integrated Circuit

MOPSO Multi-Objective Particle Swarm Optimization

MOSA Multi-Objective Simulated Annealing

NSGA Non-dominated Sorting Genetic Algorithm II

POF Pareto Optimal Front

PSO Particle Swarm Optimization

SA Simulated Annealing

SoC System-on-a-Chip

SVM Support Vector Machine

VLSI Very Large Scale Integration

FOM Figure-of-Merit

MOO Multi Objective Optimization

1 Introduction

In the last decades, Very Large Scale Integration (VLSI) technologies have been widely improved, allowing the proliferation of consumer electronics and enabling the steady growth of Integrated Circuit (IC) market. The increasing need of faster, optimized and reliable electronic devices urges the cost-effective development of such devices to efficiently meet customers' demand under highly competitive time-to-market pressure.

Today's electronic systems are extremely complex multimillion transistor ICs. This complexity is only possible because the designers are assisted by Computer Aided Design (CAD) tools that support the design process. While most of the functions are implemented using digital or digital signal processing (DSP), some functions remain analog and are integrated together with the digital part leading to mixed-signal systems-on-chip (SoC) designs.

Analog ICs designs are difficult to design and reuse, consequently designers have been replacing functions of analog circuits for digital computing whenever possible. However, there are some functionalities that will remain analog, such as:

- **Sensing the systems inputs:** the signals of a sensor, microphone or antenna has to be detected or received, amplified and filtered, to enable digitalization with good signal-to-noise and distortion ratio. Typical applications of these circuits are in sensor interfaces, telecommunication receivers or sound recording;
- **Converting analog signals to digital signals:** mixed-signal circuits such as sample-and-hold, analog-to-digital converters, phase-locked loops and frequency synthesizers provide the interface between the input / output of a system and digital processing parts of a SoC;
- **Converting the digital output back to analog:** the signal from digital processing must be converted and strengthened to analog so the signal can be conducted to the output with low distortion;
- **Provide and regulate power:** Voltage/current reference circuits and crystal oscillators offer stable and absolute references for the sample-and-hold, analog-to-digital converters, phase-locked loops and frequency synthesizers;
- **The implementation at transistor level of the digital gates**: The last kind of analog circuits are extremely high performance digital circuits. As exemplified by the microprocessors custom sized as analog circuits, for achieving the highest speed and the lowest power consumption.

Telecommunications, medical and multimedia applications make extensive use of electronic devices where blocks of analog mixed signal (AMS), digital processors and memory blocks are integrated together [1] [2]. The increase in the performance of ICs is mostly supported by an exponential increase in the density of transistors present in ICs while inversely reducing the transistors' cost, as described by Moore's law [3] . Moore's law states that every two years the density of transistors on ICs doubles at the same cost. Moore's law is a law of economics not physics, and Moore itself already preconized its end, as such an exponential law "can't continue forever", but it is still valid today.

Despite the advantages that the new fabrication technologies and the increase in device density has to systems performance, such reduction of device sizes and increase in density, reduced the analog performance of the devices while increasing variability and imposing extra constraints to the circuit designer, further decreasing their productivity.

As stated before the analog parts in a SoC occupy only approximately 20% of the global circuit area (as shown in Figure 1-1) but the design effort is considerably higher in comparison to the design effort of the digital section. In the digital IC design several mature Electronic Design Automation (EDA) tools and design methodologies are available that help the designers keeping up with the new capabilities offered by the technology, and, making the circuit reuse usual, leading to an increased design productivity. By contrast and despite the algorithms and techniques introduced in the last 25 years, in analog design there are no mature and well-defined strategies to address a problem, leading to custom solutions that are difficult to reuse.

Given the giant growth of AMS systems, pressed by the need of electronic products, which are affordable and reliable, and developed under very strict time-to-market constraints, the development and improvement of computer aided design (CAD) tools that increase analog designers' productivity and the quality of the resulting designs is an urgent need.

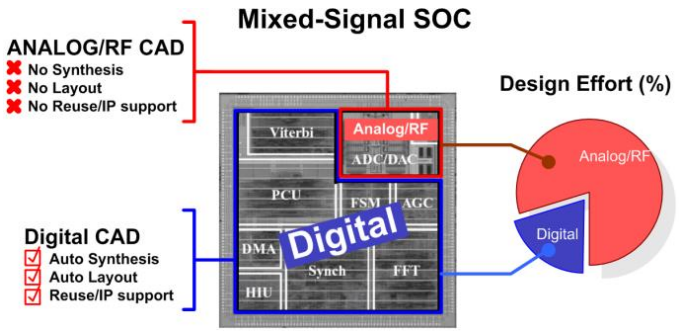


Figure 1-1 - Digital *versus* analog design reality *[4].*

One of the problems with analog design automation is that the exact design flow varies among designers, projects and companies. Nevertheless, a design flow well known and generally accepted for analog-mixed signal ICs is described in Figure 1-2. This design flow was introduced by Gielen and Rutenbar in [2], which consists of a series of top-down design selection and specifications step by step, from system level to the device level, and bottom up layout generation and verification.

At the circuit level, a design task is performed for each analog block. This process is executed iteratively in order to determine the physical dimensions of each device. In this phase are considered two major tasks: the selection of circuit topology and the circuit sizing where the design parameters of the electronic devices are defined. The specifications are then, verified through simulation of the circuit by, e.g., HSPICE® [5] or Spectre® [6]. After the analog blocks have been sized, the project enters into the next phase where the analog blocks are mapped into a physical representation of the circuit, the layout. The layout is a set of geometric shapes that obey to the rules defined by the manufacturing process. Then, the layout passes the design rule check and the layout-versus-schematic verification stages, and

finally, it is extracted and simulated to verify the impact of layout parasitic effects on the overall performance of the circuit.

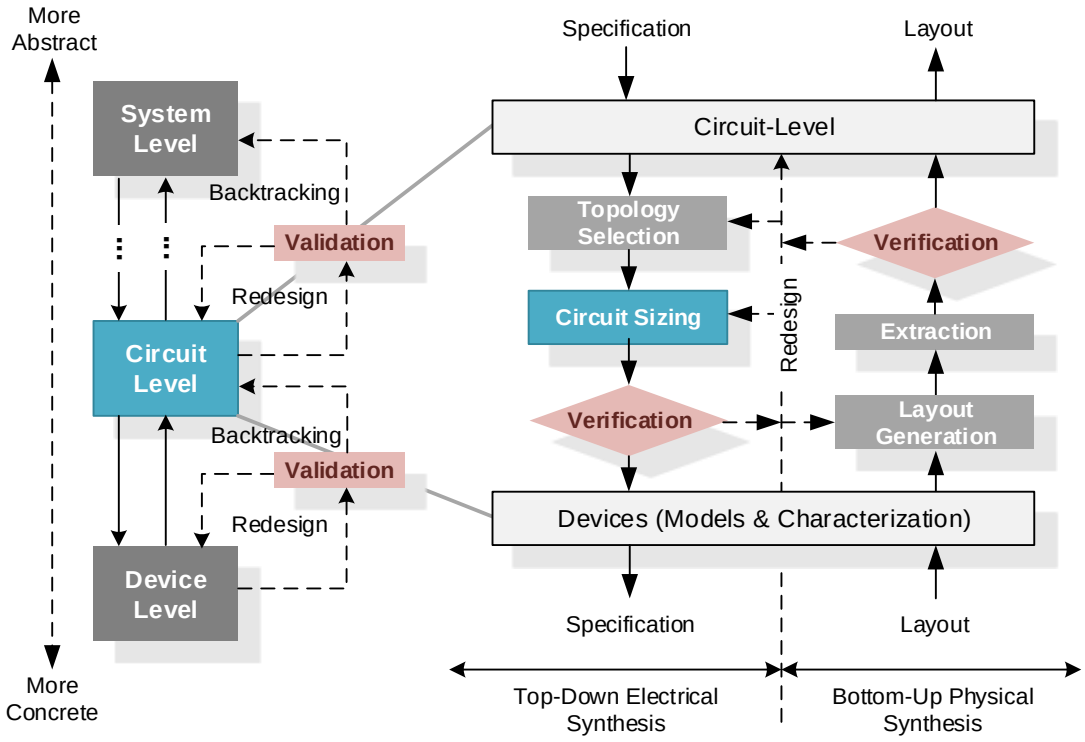


Figure 1-2 - From system level to device level tasks of analog IC design [2].

Due to the lack of automation, designers keep exploring the solution space manually. This method causes long design times, and allied to the non-reusable nature of analog IC, makes analog IC design a cumbersome task. This difference in the level of automation between analog and digital design is because analog in general is less systematic, more heuristic and knowledge intensive than the digital counterpart, and becomes critic when digital and analog circuits are integrated together. As the analog automation tools do not progress at the same pace of technology, knowledge and experience of the designer is always crucial for making decisions at all stages of the analog design flow.

In a traditional analog design, the designer defines a methodology; interacts manually with the proper tools in order to achieve the project objectives, whether they are the best sizing of circuit parameters to meet the desired performance specifications, either to optimize the parameters for specific application (DC Gain, power, area, etc.), aiming to obtain at the end a robust design. However, the search space of the objective function, which relates the design variables to the performance specifications of the circuit, is characterized by a complex multidimensional and irregular space, making the manual search for the ideal solution difficult to achieve. Along with the time constraints that designers face, this search of the design space becomes very limited.

Despite its fundamental aid to designers, those tools have limited automation options, and the ones available are usually not used by the majority of the designers. The time required to manually implement an analog project is usually of weeks or months, which is in opposition to the market pressure to accelerate the release of new and high performance ICs. To address all the difficulties to solve these

problems, electronic design automation (EDA) CAD is a solution increasingly strong and solid. The main focus of this dissertation is at automatic circuit-level sizing and optimization.

1.1 Motivation

In summary, despite the trend to implement almost everything using digital circuitry, is notorious the importance of analog circuits as some parts cannot be ported to the digital world. The differences of effort and time consumption spent between digital and analog circuitry design are easy to spot, putting an enormous pressure into the industry and academia to improve tools and methodologies to aid analog circuit designers to improve their productivity and reduce production costs. Moreover, because of the extremely large number of variations of analog circuits and fabrications processes, it is extremely difficult to make fair and accurate comparison between approaches. This is the scenario where AIDA Framework [7] appears as an EDA tool, more details about the AIDA project can be found in *'www.aidasoft.com'*, where AIDA-C is the tool developed for analog integrated circuit sizing and optimization.

1.2 Objectives

Lead by those thriving ideas, this work focus is AIDA-CMK that targets the sizing of the devices in analog circuits using state-of-the-art and innovative multi-objective optimization (MOO) techniques, by enhancing AIDA-C with a new abstraction layer that permits the easy inclusion of new multi-objective multi-constraint optimization techniques aside the NSGA-II that was originally supported. Taking advantage of the new abstraction layer, the MOO kernels, NSGA-II, MOPSO, MOSA and the two multiple kernel hybridization are included in AIDA-CMK.

This work was supported in part by the Instituto de Telecomunicações (Research project OPERA - PEst-OE/EEI/LA0008/2013) and by the Fundação para a Ciência e Tecnologia (Research project DISRUPTIVE EXCL/EEI-ELC/0261/2012.

The specific goals of this work are:

- Implementation of a modular framework for optimizations in the scope of analog circuits optimization;
- Implement some classical multi objective optimization (MOO) methods to verify the approach;
- Test the performance of different optimizations approaches:
    - Standard benchmarks;
    - Analog Integrated Circuits;
- Evaluate hybrid optimization methods.

1.3 Achievements

The development of this work lead to the following achievements (1 conference paper; 1 book chapter; 1 EDA tool):

- R. Póvoa, **R. Lourenço**, N. Lourenço, A. Canelas, R. Martins, N. Horta, *"LC-VCO Automatic Synthesis Using Multi-Objective Evolutionary Techniques",* Proceedings of the International Symposium on Circuits and Systems (ISCAS 2014), Melbourne, Australia, June 2014. (selected as one of the **10 finalists** for the **ISCAS 2014 Student Best Paper Award** contest)
- R. Póvoa, **R. Lourenço**, N. Lourenço, A. Canelas, R. Martins, N. Horta, "Synthesis of LC-Oscillators using Rival Multi-Objective/Multi-Constraint Optimization Kernels", chapter in "Performance Optimization Techniques in Analog, Mixed-Signal, and Radio-Frequency Circuit Design", IGI Global chapter (accepted).
- AIDA-CMK (AIDA-C Multi-Kernel).

1.4 Document structure

This rest of this document is organized as follows:

In chapter 2 an overview of the state of the art in analog circuit optimization is presented, focusing on the optimization approaches that are used. This study is then used to select the optimization methods to be considered in the framework to be developed.

In chapter 3, AIDA-CMK and the circuit optimization is presented, describing the proposed architecture for the multi objective circuit optimization framework. Also, the NSGA-II, MOSA and MOPSO algorithms implemented are described.

In Chapter 4 the details about the implementation are presented, showing the application layers and their implementation. The structure of the classes implemented is described in detail showing their relations and the flexibility of the implemented framework.

In chapter 5 the algorithms are experimented with constrained problems from CEC 2009.´In chapter 6 the application of multiple optimization kernels to optimize real analog IC designs. Finally, in Chapter 7 the conclusions are addressed and some directions for future developments are suggested.

2 Previous work on automated analog circuit sizing

In the last 25 years, the scientific community proposed many techniques for the automation of the analog circuit sizing. In this chapter those approaches are briefly surveyed focusing on the optimization techniques that are used.

The analog circuit sizing automation techniques are classified into two main groups, the knowledge-based approaches and the optimization based approaches [4]. This classification is based on the fundamental techniques used to address the problem, as illustrated in Figure 2-1.

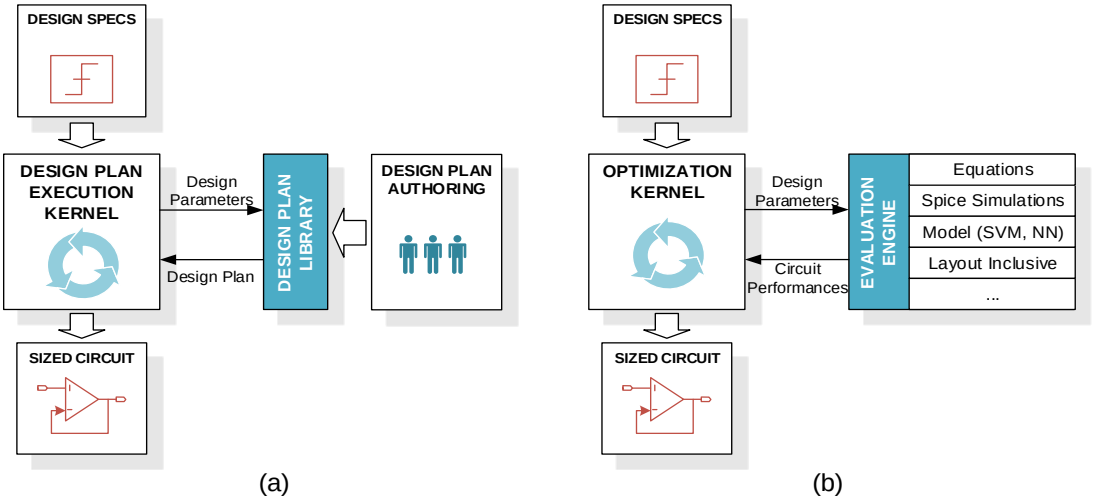


Figure 2-1 - Automatic circuit sizing approaches
(a) knowledge-based (b) optimization-based

Early automation systems [8, 9, 10, 11] did not use optimization and tried to systematize the design by using a design plan derived from expert knowledge. In these methods, a plan is built with design equations and a design strategy that produce the component sizes that meet the performance requirements. These knowledge-based approaches were applied with moderate success. The main advantage of this approach is the short execution time. However, deriving the design plan is hard and time-consuming, and the design plan requires constant maintenance in order to keep it up to date with technological evolution, also, the results are not optimal, suitable only as a first-cut-design.

The next generations of sizing tools apply optimization techniques to analog IC sizing, they can be further classified into two main subclasses: equation-based or simulation-based, from the method used to evaluate the circuit's performance.

The equation-based methods use analytic design equations to relate the circuit's performance figures to the design variables. Different optimization techniques are used, both deterministic and stochastic. Taking advantage of knowing the equations and their properties allows the use of the classical optimization methods. In OPASYN [12] the optimization is performed using steepest descent, similarly in STAIC [13] it is used a successive solution refinements technique.

Maulik et. al. [14, 15] define the sizing problem as a constrained nonlinear optimization problem using spice models and DC operating point constraints, solving it using sequential quadratic programming. Matsukawa et. al. [16] design ΔΣ and pipeline analog to digital converters solving via convex optimization the equations that relate the performance of the converter to the size of the components.

In GPCAD [17] a posynomial circuit model is optimized using Geometrical Programming (GP), the execution time is in the order of few seconds, but the general application of posynomial models is difficult and the time to derive the model for new circuits is still high. Kuo-Hsuan et. al. [18] revisited the posynomial modeling recently, surpassing the accuracy issue by introducing an additional generation step, where local optimization using simulated annealing (SA) and a circuit simulator is performed. The same strategy is applied in FASY [19, 20] were analytical expressions are solved to generate an initial solution and a simulation-based optimization is performed to fine tune the solution.

Other equation based approaches do not limit the problem formulation in order to use a specific optimization technique at all, relying on heuristic optimization instead. OPTIMAN [21] uses SA applied to analytical models, and, in ASTRX/OBLX [22], a SA optimization is also performed using and cost function defined by equations for dc operation point, and small signal Asymptotic Waveform Evaluation (AWE)-based simulation, this evaluation technique is also used in DARWIN [23], which uses Genetic Algorithms (GA) instead. Doboli et. al. [24] also applies genetic programming techniques to simultaneously derive the sub-blocks specifications, sub-block topology selection and transistor sizing.

The equation-based methods' strong point is the short evaluation time, making them, like the knowledge-based approaches, extremely suited to derive first-cut designs. The main drawback is that, despite the advances in symbolic analysis, not all design characteristics can be easily captured by analytic equations, making the generalization of the method to different circuits very difficult. In addition, the approximations introduced in the equations yield low accuracy designs especially for complex circuits, requiring additional work to ensure that the circuit really meets the specifications.

2.1 Simulation-based automatic circuit sizing

With the availability of computing resources, simulation based optimization gained ground, being the most common approach found in recent approaches. In simulation-based sizing, as in the case of AIDA-C, a circuit simulator, e.g., SPICE [25], is used to evaluate the circuit.

Early approaches to simulation based automatic sizing used local optimization around a "good" solution, where SA [26] is the most common optimization technique used. SA mimics the annealing of material under slow cooling to minimize the internal energy, as the name suggests. In DELIGTH.SPICE [27] the optimization algorithm (phase I-II-III method of feasible directions) is used to perform local design optimization around a user provided starting point. Kuo-Hsuan et. al. [18] and FASY [20, 19] use equation-based techniques to derive an approximate solution, and then use simulation within a SA kernel to optimize the design. Likewise, Cheng et al. [28] also uses SA but considers the transistor bias conditions to constrain the problem, and, instead of solving the circuit by finding transistor sizes, the problem is solved by finding the bias of the transistors. FRIDGE [29] aims for general applicability

approach by using an annealing-like optimization without any restriction to the starting point. Castro-Lopez et al. [30] use SA followed by a deterministic method for fine-tuning to perform the optimization.

Other widely used class of optimization methods is the Genetic Algorithms (GA). The principle of the GA is to mimic natural evolution where new solution vectors are obtained from the current population by the application of the genetic operators of mutation and crossover. Barros et. al. [4, 31, 32] presents a circuit sizing optimization supported by a genetic algorithm where the evaluations of the populations was made using a both the circuit simulator and automatically trained support vector machine. Alpaydin et. al. [33] use hybridization of evolutionary and annealing optimization strategy where the circuits' performance figures are computed using a blend of equations and simulations. Given the affinity evolutionary algorithms have with parallel implementations, in Santos-Tavares [35], MAELSTROM [34] and ANACONDA [36] the time to simulate the population reduced by a parallel mechanism that shares the evaluation load among multiple computers. Because the traditional use of local search methods in many implementations, the MAELSTROM's authors option was to use a hybridization, i.e., the parallel re-combinative simulated annealing (PRSA). In ANACONDA the approach is similar but instead of the PRSA it is applied a variation of pattern search algorithms, named by the authors as stochastic pattern search.

A different approach to circuit sizing optimization that also employs evolutionary methods is to simultaneously generate the circuit topologies (the arrangement of the devices) and the device sizes. Koza [37] Sripramong [38], and more recently Hongying [39] proposed a design methodology able to create new topologies by exploring the immense possibilities starting from low abstraction level. Small elementary blocks are connected bottom-up to each other to form a new topology. Various fundamental entities can be applied, such as, single transistors, elementary building blocks or node connections. However, these approaches are met with great skepticism as designers are suspicious of the generated structures, because they often differ too much from the well-known analog circuit structures.

Swarm intelligence algorithms [40] can also be found in the literature applied to analog circuit sizing. The fundament of swarm intelligence algorithms is to use many simple agents that lead an intelligent global behavior, like the one observed in many insect hives. From these methods, the most common are the ant colony optimization (ACO), which was successfully applied in [41, 42], and particle swarm optimization (PSO) that can be found in [43, 44, 45].

Circuit sizing is in its essence a multi-objective multi constraint problem, and the designer often explores the tradeoff among contradictory performance measures, for example, minimizing power consumption while maximizing bandwidth, or maximizing gain and minimizing area of an amplifier, as such, the usage of multi-objective optimization techniques is becoming more common. When considering multiple objectives the output is not one solution, but a set of optimal design tradeoff solutions, usually referred as Pareto Optimal front (POF). Given the multiple elements already present in both evolutionary and swarm intelligence algorithms, these are the natural candidates to implement such approach. In GENOM-POF [46, 47] and MOJITO [48] the evolutionary multi-objective methods are applied, respectively, to circuit sizing and both sizing and topology exploration, whereas in [45] the particle swarm

optimization is explored in both single and multi-objective approaches. A different approach is taken by Pradhan and Vemuri in [49], where the multi-objective simulated annealing (MOSA) is used.

Instead of executing the sizing on-the-fly, in some approaches the non-dominated solutions are generated using the previously referred multi-objective optimization methods or variations of them for the most relevant tradeoff (prior to the design task) and then, the suitable solution is selected from the already sized solutions [50, 51, 52, 53].

From the study of analog circuit sizing and optimization approaches proposed by the scientific community recently, it is clear that there is not a specific trend to single algorithm, but many were experimented with. In the next section the summary of the surveyed approaches is presented, and finally the objectives for this work are refined, namely the selection of the optimization kernels to be initially included in the platform.

2.2 Optimization techniques applied to analog circuit sizing

The analog sizing tools approaches surveyed are summarized in Table 2-1. In the equation-based systems the usage of classical optimization methods is possible, however the accuracy of the models and the derivation of such equations strong limits applicability. This limitation of the equation based systems is overcome at the expense of evaluation time by using accurate circuit simulation to evaluate the performance figures being optimized.

By using the circuit simulator methods that take advantage of some properties of the models, quadric or geometric programming cannot be used, leading, as seen, to the usage of stochastic heuristic optimization techniques. From the approaches that were surveyed the most common stochastic algorithms were based on simulation annealing and genetic/evolutionary approaches, with some of the latest implementations considering particle swarm optimization PSO and ant colony methods.

Table 2-1 - Summary of analog IC design automation tools for sizing and optimization

Tool\Author		Design Plan/ Optimization Method	Evaluation	Time Setup/Exec.
IDAC [10]	1987	Design plan plus SA post-optimization	Equations	months/few sec.
DELIGTH.SPICE [27]	1988	Feasible directions Optimization	Simulator	moderate/18h
OPASYN [12]	1990	Steepest descent	Equations	2 weeks/5 min.
OPTIMAN [21]	1990	SA	Equations	⦸/1 min.
STAIC [13]	1992	2 step optimization	Equations	long/2 min.
Maulik et al. [14, 15]	1993	B&B, and Seq. Quadratic Progr.	Equations and BSIM models	6 months/1 min
FRIDGE [29]	1994	SA	Simulator	1 hour/45 min
DARWIN [23]	1995	GA	small signal, analytical expressions.	⦸/⦸
ISAID [8, 9]	1995	Qualitative reasoning + post optimization	Equations and Qualitative reasoning	⦸/⦸
FASY [20, 19]	1996	SA + Gradient	Simulator	⦸/6 hours
ASTRX/OBLX [22]	1996	SA	AWE Equations	few days/few seconds
Koza [37]	1997	GA	Simulator	⦸/⦸
GPCAD [17]	1998	Geometric Programming	Posynomial	⦸/fast
MAELSTROM [34]	1999	GA+SA	Simulator	⦸/3,6 hours
ANACONDA [36]	2000	Stochastic pattern search	Simulator	⦸/10 hours
Sripramong [38]	2002	GA	Simulator	⦸/3 days
Alpaydin [33]	2003	Evolutionary strategies + SA	Fuzzy + NN trained with Simulator	⦸/45 mins
Shoou-Jin [54]	2006	GA	Equations	⦸/⦸
Barros [4, 31, 32]	2006	GA	Simulator	⦸/20 min
Castro-Lopez [30]	2008	SA + Powels method	Simulator	⦸/25 min
Santos-Tavares [35]	2008	GA	Simulator	⦸/⦸
MOJITO [48]	2009	NSGA-II	Simulator	⦸/<7 days
Pradhan [49]	2009	Multi-Objective SA	Layout aware MNA models	⦸/16 min
Matsukawa [16]	2009	Convex Optimization	Convex functions	⦸/⦸
Cheng [28]	2009	SA	Equations	⦸/<1 hour
Hongying [39]	2010	GA with VDE	Simulator	⦸/⦸
Fakhfakh [45]	2010	Multi-objective PSO	Equations	⦸/<1 min.
Kuo-Hsuan [18]	2011	Convex optimization Stochastic Fine Tuning	Posynomial Simulator	⦸/1 hour
Kamisetty et al. [43]	2011	PSO		⦸/⦸
Benhala et al. [42]	2012	ACO	Equation	⦸/<1 min.
Roca et al. [52]	2012	NSGA-II	Simulator	⦸/⦸
Gupta & Gosh [41]	2012	ACO	Simulator	⦸/<2 hour
Kumar & Duraiswamy [44]	2012	PSO	Simulator	⦸/⦸
Genom-POF [47, 46]	2012	NSGA-II	Simulator	⦸/<1 hour

2.2.1 Optimization methods selection

This work is in the scope of circuit sizing which considers electric simulation to evaluate the circuits' performance, as illustrated in Figure 2-2. The generality of the approach is increased and the setup time for new circuits is decreased. However, the relation between the performance figures and the design variables becomes unknown, making the usage of classic optimization methods inappropriate, as seen in the surveyed simulation-based systems where almost all consider heuristic optimization methods.

In AIDA-C the circuit sizing and optimization problem, which will be described in detail in chapter 3, is modeled as a multi-objective multi-constraint optimization problem. In this context special relevance is given to multi-objective algorithms. Historically, both SA and GA have been used intensively, in this sense is natural to consider at least an evolutionary and an annealing. Given the recent experiments with swarm intelligence in this domain, and to broad the scope of the implemented framework, this class of methods should also be considered.

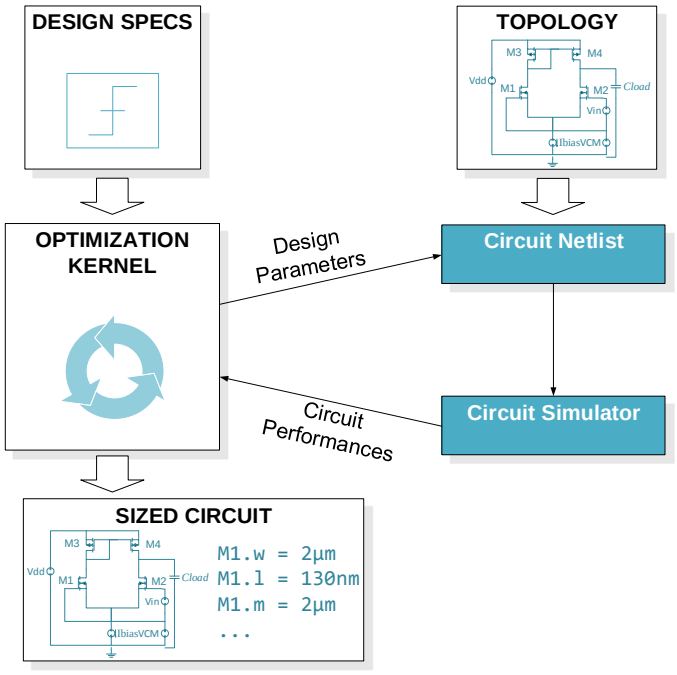


Figure 2-2 - Automatic optimization-based circuit sizing using simulator

From a brief perusal of the multi-objectives implementations, the ideas like non-sorted domination or crowding distance (further described in chapter 3) presented in NSGA-II are reused by several other methods, as that the advantages of the inclusion of NSGAII in the framework are clear. Given the usage of SA, some sort of multi-objective SA should also be considered.

In terms of the swarm intelligence algorithms, both ACO and PSO have been applied to circuit sizing. Because of MOPSO is already found in the literature and the un-natural application to real valued problems of the path finding ideas of the ACO, MOPSO will be considered.

2.3 Conclusions

In this chapter a survey of techniques and approaches to the automation of analog circuit sizing were presented. The different approaches were classified in terms of the techniques used and the most significant aspect observed was the setup and the execution time, as well as the accuracy in the evaluation of the solutions. In this survey were presented several ADA tools and analyzed to better understand the advantages, and, drawbacks that can be improved in the future. It was also possible to identify that a wide range of optimization techniques are considered in this domain and new ones are always being introduced.

In this work in, AIDA-CMK improves AIDA-C by adding a flexible and systematic manner to try and experiment new optimization techniques, so that further improvements to the automation of analog circuits design, namely in the circuit sizing and optimization, can be achieved more efficiently. The trends in optimization methods were also surveyed, showing a predominance of the multi-objective approaches, and the presented study was used to select a set of methods that will be considered initially to demonstrate the proposed solution.

3 AIDA-CMK: AIDA-C with MOO framework

This chapter explains the circuit optimization tool, AIDA-C, and the changes proposed in this work to enhance the tool with multiple algorithms, leading to AIDA-CMK.

3.1 AIDA-C

AIDA-C stems from GENOM-POF [46] and is part of the AIDA [7] design automation framework illustrated in Figure 3-1, that implements the complete analog IC design flow from device sizing to layout generation. AIDA-C performs the analog circuit sizing and optimization part of the flow, addressing robust design requirements by considering extreme process, voltage and temperature (PVT) corner conditions together with the use of the industrial grade circuit simulators, HSPICE® [5] and ELDO® [55], for accurate circuit's performance evaluation. AIDA-C can also use AIDA-L's floorplanner to add geometrical layout measures (e.g. total area, device area, aspect ratio, etc.) to the set of circuit's performance figures to be considered during optimization.

Finally, the layout generator AIDA-L, previously known as LAYGEN-II [56], inputs the device sizes and floorplan template to generate the corresponding layout by placing and routing all the devices, completing the design flow. A final validation step is done using the physical verification tool CALIBRE® [57].

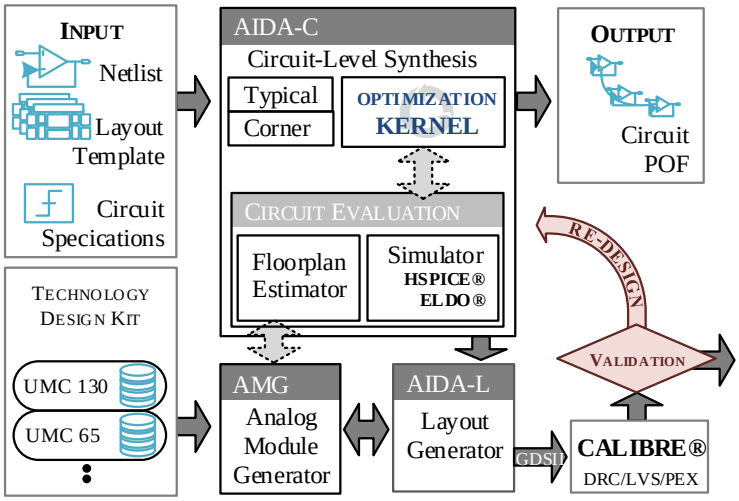


Figure 3-1 - AIDA framework

In AIDA-C, circuit sizing and optimization is implemented as a multi-objective multi-constraint optimization problem, originally supported by the multi-objective evolutionary algorithm NSGA-II, where the multi-objective optimization problem is defined as:

$$\begin{aligned} &find\ x\ that\ minimize\ f_m(x) && m = 1,2,...M \\ &subject\ to\ g_j(x) \geq 0 && j = 1,2,...J \\ &x_i^L \leq x_i \leq x_i^U && i = 1,2,...N \end{aligned} \qquad (3.1)$$

where, x is a vector of N optimization variables, $g_j(x)$ one of the J constraints and $f_m(x)$ one of the M objective functions. Given the multi-objective nature of the sizing method, the output is not one solution

but a set of solutions all compliant with the design specifications. The optimizer's output is a set of Pareto non-dominated solutions or Pareto front. Pareto dominance states that one point in the solution space, A, is not dominated by another point B, if $\exists_m: f_m(A) < f_m(B)$. Figure 3-2 depicts a Pareto front, illustrating the concept, where solutions A and B are non-dominated and both dominate solution point C.

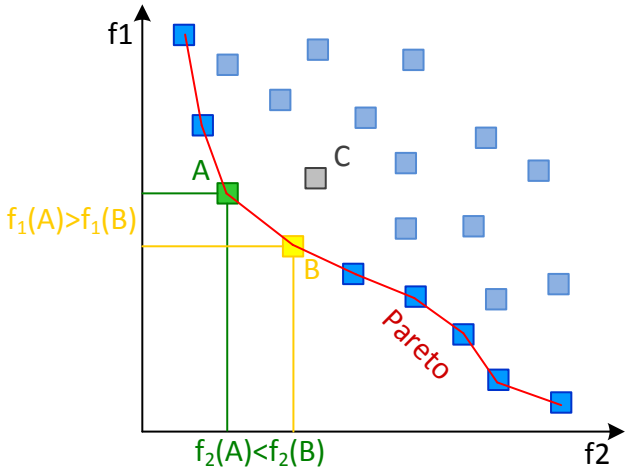


Figure 3-2 - Pareto front illustration

AIDA-C targets the sizing of the devices in analog circuits using state-of-the-art and innovative techniques. The focus of this work is to enhance the optimization kernel with an abstraction layer that permits the easy inclusion of new optimization techniques aside the NSGA-II, which was originally supported. As stated in chapter 2, besides the definition and implementation of the framework, default implementations of MOPSO and MOSA methods are also provided and applied to circuit optimization. The new tool AIDA-CMK, after the implementation of the optimization kernel framework, is shown in Figure 3-3. Additionally, the algorithm implementations share a common interface with AIDA-CMK and between themselves, easing the intermingling of tentative solutions between technics in order to, not only use the different approaches by themselves, but also ease the hybridization. To explore this feature a Hybrid method that combines the previously referred optimization kernels is also implemented and explained in Section 3.6.

The definition in (3.1) is used in to create an abstraction layer between the optimization method and the circuit being optimized, where the evaluation of the circuit performance is done using the circuit simulator. Geometric and performance figures that depend of the physical layout representation of the circuit are considered using AIDA-L to generate the layout and extract the corresponding figures. However, and in the scope of this work, as the algorithms to be implemented do not depend on tool or tools used to measure the circuit performance, layout dependent performance measures are not considered.

The next section describes how the circuit sizing design problem is mapped into the multi objective optimization problem as defined in (3.1), and the following sections will describe with more detail the optimization kernels considered in this first implementation.

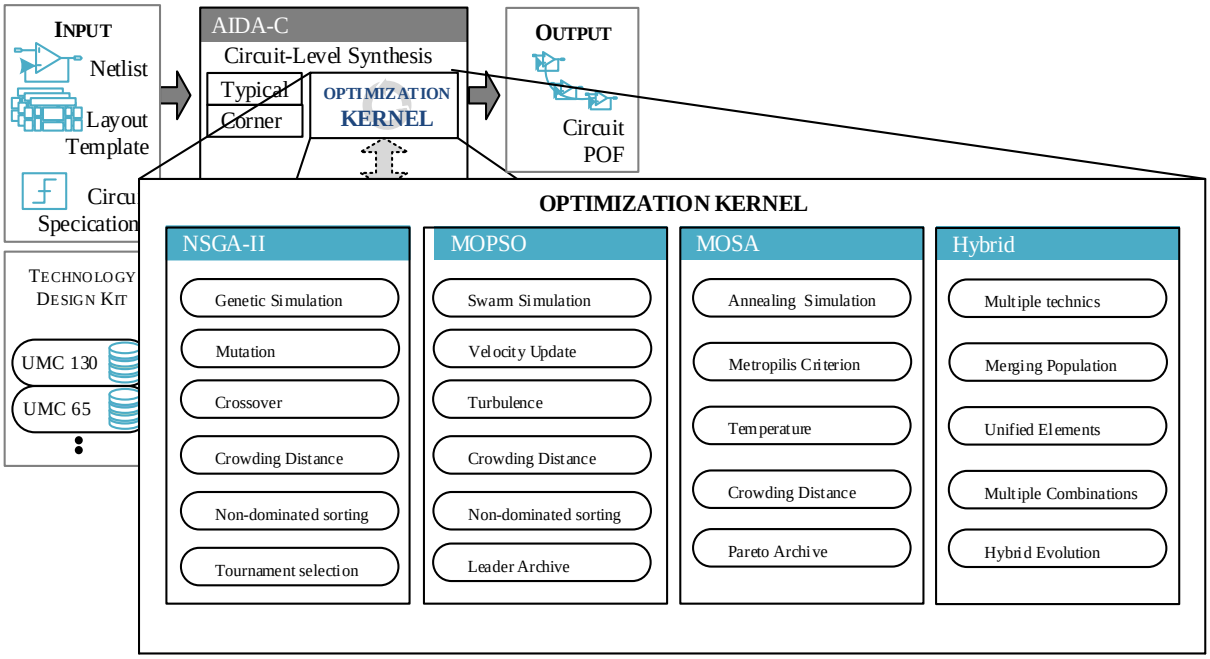


Figure 3-3 - AIDA-CMK optimization algorithms

3.2 Circuit sizing as multi-objective optimization problem

In this section, the procedure used to convert the analog IC designer inputs to the problem formulation shown in (3.1) is described. The multi-objective optimization, as defined in (3.1), to the circuit sizing problem, the inputs from the designer, which are not provided as the tuple {*x*, *f*, *g*}, need to be properly mapped. The simple differential amplifier from Figure 3-4 will be used to illustrate the procedure, where Table 3-1, Table 3-2 and Table 3-3 show the circuit parameters, design objectives and target specifications, respectively.

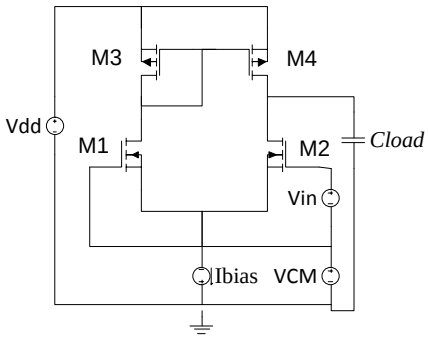


Figure 3-4 - Differential amplifier

Table 3-1 - Parameters ranges for the differential amplifier.

Var.	W1	W2	L1	L2	Ib
Max.	400.0e-6	400.0e-6	15.0e-6	15.0e-6	500.0e-6
Min.	1.0e-6	1.0e-6	0.35e-6	0.35e-6	100.0e-6

Table 3-2 - Objectives for the example in differential amplifier.

Performance	Target	Units	Description
A0	Maximize	dB	Gain DC
Gbw	Maximize	MHz	Unit-gain frequency

Table 3-3 - Design specifications for the example in differential amplifier.

Performance	Target	Units	Description
Gbw	≥ 35	MHz	Unit-gain frequency
Pm	65 ≤ pm ≤ 90	Degree	Phase margin
vov_m1	50 ≤ vov_m1 ≤ 200	mV	Vgs –Vt
vov_m2	50 ≤ vov_m2 ≤ 200	mV	Vgs –Vt
vov_m3	100 ≤ vov_m3 ≤ 300	mV	Vgs –Vt
vov_m4	100 ≤ vov_m4 ≤ 300	mV	Vgs –Vt
delta_m1	≥ 50	mV	Vds – Vdsat
delta_m2	≥ 50	mV	Vds – Vdsat
delta_m3	≥ 50	mV	Vds – Vdsat
delta_m4	≥ 50	mV	Vds – Vdsat

When considering only typical case simulation, the design objectives being minimized are used directly as one of the $f_m(x)$, and the ones being maximized are multiplied by -1. The design constraints are normalized and multiplied by -1, if necessary, according to (3.2).

$$f_m(x)=\begin{cases} p_m & \text{when minimizing } p_m \\ -p_m & \text{when maximazing } p_m \end{cases}$$

$$g_i(x)=\begin{cases} \dfrac{p_i-P_i}{|P_i|} & \text{when the constraint is } p_i \geq P_i \\ p_i & \text{when the constraint is } p_i \geq P_i \text{ and } P_i=0 \\ -p_i & \text{when the constraint is } p_i \leq P_i \text{ and } P_i=0 \\ \dfrac{P_i-p_i}{|P_i|} & \text{when the constraint is } p_i \leq P_i \end{cases} \qquad (3.2)$$

where, p_i is the measured circuit characteristic, and P_i is the correspondent acceptable limit. The circuit parameters are used as ranged design variables and define the search space.

Table 3-4 illustrates the objective and constraint functions for the differential amplifier circuit in Figure 3-4. Modeling the problem in this manner, the circuit is evaluated using only for nominal conditions, therefore it requires less simulations, i.e. is faster. Despite the output does not consider the limitations imposed by the extreme variations of process and environment parameters, it is useful to the circuit designer to perform tradeoff analysis.

Table 3-4 - $f_m(x)$ and $g_j(x)$ for the differential amplifier example.

Constraints	$g_0(x)=\dfrac{gbw}{35\times10^6}-1$	$g_1(x)=\dfrac{pm}{65}-1$	$g_2(x)=1-\dfrac{pm}{90}$
	$g_3(x)=\dfrac{vov\_m1}{50\times10^{-3}}-1$	…	$g_{15}(x)=\dfrac{delta\_m4}{50\times10^{-3}}-1$
Objectives	$f_0(x)=-gain\_dc$	$f_1(x)=-gbw$	-

A critical problem in analog IC design is the process and environment variability, i.e., devices designed to be equal are different after production due to manufacturing mismatches and variation in the supply voltages and temperatures. This phenomenon affects devices in different chips but also devices within the same chip, and must be solved by a robust circuit design. To verify if the design is robust, i.e., the vast majority of the fabricated circuits will work according to the specifications, special techniques are

employed. One of such techniques is PVT corners simulations. PVT Corners is a worst-case approach where the circuit is simulated over multiple combinations of extreme process parameters variations, power supply, temperature, etc. Figure 3-5 illustrates 8 corners cases obtained by considering 3 values for power supply, operating temperature and process parameters.

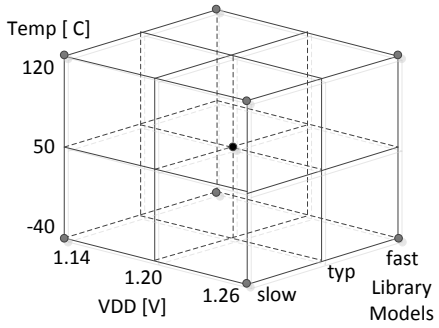


Figure 3-5 - Corner cases

To include this effects in the optimization, the design is evaluated considering all the corners, i.e., for each evaluation, the circuit is simulated once for each corner case, this makes the execution slower when compared to typical, but the output circuits are ensured to be feasible in all tested corner conditions. To handle the multiple corners, the objective and constraint functions are modified using (3.3)

$$\hat{f}_m(x) = \max_{c=1,2,..C}\left(f_m^c(x)\right)$$
$$\hat{g}_j(x) = \sum_{c=1}^{C} c_j^c(x) \text{ with } c_j^c(x) = \begin{cases} 0 \text{ if } g_j^c(x) \geq 0 \\ g_j^c(x) \text{ if } g_j^c(x) < 0 \end{cases} \quad (3.3)$$

where, C is the number of corners, and $f_m^c(x)$ and $g_j^c(x)$ are respectively the objective $f_m(x)$, and the constraint $g_j(x)$, as defined for the typical case evaluated in corner case *c*. In this worst case approach, each objective, which is being minimized, evaluates to the maximum value obtained from the simulation of circuit in all the corner cases, and each constraint is evaluated as the sum of the normalized violation in all the corner cases where it is violated.

In this way, both nominal and worst case optimization is mapped to the tuple (*x*, *f*, *g*). From this point on, the circuit optimization is viewed by the optimization methods as the tuple {x, f, g}. Therefore, given the faster execution and without the loss of generality, testing the algorithms is done considering only the nominal case.

With this interface defined, the definition of a general interface based on the standard definition of the optimization problem would be simple; however, there is a secondary requirement for this interface. In the AIDA framework, the AIDA-L's detailed routing is also an innovative optimization-based approach, where all the wires in the layout are evolved simultaneously, unlike the remaining state-of-the-art approaches. While the study of circuit layout is out of the scope of this thesis, the problem definition within the developed framework should be general enough to accommodate the complex representation of the genome in the detailed Router, so it must be considered when developing the proposed interface.

Another important feature is the possibility to define elements to be used as starting point, as it permits sequential execution of multiple optimization tasks. Two important uses of this feature: execute an optimization where the evaluation is done considering only nominal conditions and use the output of that optimization task as starting point of another that considers the corner cases; and/or execute an initial optimization using algorithms suited for exploration like the Gas, and then, execute an optimizations using algorithms that are more efficient exploiting the local minima like SAs.

The multi-objective optimization kernels that are going to be implemented within the scope of this work, NSGA-II, MOPSO, MOSA and Hybrid/Multi-Kernel algorithm are described in the next sections.

3.3 NSGA-II

The NSGA-II [58] kernel is an evolutionary optimization scheme. The principle of evolutionary computation is to mimic natural evolution. It operates over a population composed by several chromosomes, each one representing a different solution. New solution vectors are obtained from the current population by the application of the genetic operators of mutation and crossover. The crossover operation uses genes from two population elements, called parents, to generate the new elements, called children, combining randomly selected sets of information from each of the parents into the children. The mutation is a random change in individual's genetic information in order to escape from local minima; the mutation operator introduces new information in the chromosome whereas the crossover selected the best pieces of the information present in the population genetic information.

The new individuals' fitness is evaluated and then they are ranked together with the parents. The fittest individuals are selected as the new parents, and the less fit discarded. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is. Selection compares each individual in the population by using a fitness function. Each chromosome has an associated value corresponding to the fitness of the solution it represents. The fitness should correspond to an evaluation of how good the candidate solution is.

The genetic algorithm starts by generating an initial population of chromosomes, the initial parents. This first population must offer a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of the search space can be engendered but generally, the initial population is generated randomly. Then, the genetic algorithm evolves the solutions by applying the genetic operators and then selecting the next parents. The process is repeated until the convergence or ending criterion is reached. The algorithm is stopped when the population converges toward the optimal solution.

To implement genetic algorithms one must first define the genotype and phenotype. The genotype refers to the genetic encoding used to represent one point, which is called phenotype. The genotype define the in the search space and must insure that all points in the feasible space can be represented, not necessarily by only one chromosome, and also that the crossover keeps the parents' common information in the children.

In the particular case of the NSGAII, the algorithm pseudo code is shown in algorithm 1. The NSGA-II uses Pareto dominance concepts to sort the multi-objective solutions.

Algorithm 1 - NSGA II for population size P and number of generations G

```
parents = P new random solutions
generation = 0
while generation < G
  offspring = apply-operators(parents) //Mutation and Crossover
  evaluate(offspring)
  non-dominated-sorting(parents + offspring)
  parents = crowding-distance-selection(parents + offspring)
  generation++
return parents
```

The Pareto dominance is implemented by means of ranking solutions using non-dominated sorting, algorithm 2, and crowding distance criterion, algorithm 3, as a tie breaker for solutions with the same rank. The rank process is iterative making rank 1 elements the ones that are not dominated by any other, these elements are removed from the pool of elements being sorted and the process is repeated for the next rank until there are no more elements in the poll to be sorted, algorithm 2 illustrates this procedure. The elements with lower rank dominate the ones with higher rank.

Algorithm 2 - Non-dominated sorting procedure for a P of size N

```
set F1=∅ and rank=1
for j=1,…,N do:
  Frank = Frank+{P[j]}
    if(k == j) continue
    if P[k] dominates P[j], then remove P[j] from Frank break loop in k
remove Frank from P and update N = N – size(Frank)
if P is not empty, then set rank = rank+1 and go to 2
```

The cubic approach to compute the rank of the elements in the population was introduced to clearly describe the non-dominated sorting, however, in practice, the fast non-dominated sorting algorithm described in [58] is used. It first computes the dominance between all solutions, storing the set of elements that are dominated (*Sd*) and the number of elements that dominate (*d*) for each solution. Using this information, all the solutions that are not dominated (have zero elements dominating it, i.e. *d*=0) are set to the rank 1 and removed from the elements pool decrementing *d* for all the solutions in their *Sd*s. The process is repeated for rank 2, and successively until the elements pool is empty, leading to a quadratic algorithm.

To solve ties between elements of the same rank, the crowding distance criterion is used. The crowding distance is an estimate of the density of elements. Each element with the same rank is assigned a value that related to the distance to the closer elements. Figure 3-6 illustrates the four ranking fronts and the crowding distance of the solution B in a problem with two objectives.

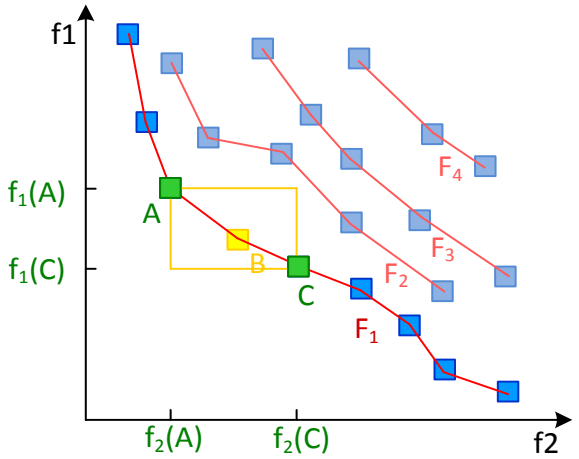


Figure 3-6 - Fronts for multiple ranks, and crowding distance for solution B

The crowding distance of the elements in a front is computed by iterating in the M objective functions, sorting the elements using each objective and for each element accumulating the normalized value of the distance between the elements before and after in the ordered set. The boundary elements (element with smaller and higher value of each objective) are assigned with infinite value of crowding distance. The pseudo code of crowding distance computation is shown in algorithm 3.

Algorithm 3 - Crowding distance computation of front F_T of size N in a problem with M objectives

```
1    for i=1,…,N do: F_T[i].cdist=0
2    for m=1,…,M do:
3      S = F_T sorted using the value objective function f_m
4      S[1].cdist = S[N].cdist = ∞
5      for j=2,…,N-1 do:
6        S[1].cdist+=(S[i+1].f_m - S[i-1].f_m)/( f_m^max - f_m^min)
```

3.4 MOSA

The MOSA is an adaptation of the single objective simulated annealing [26] to the multi objective case. Like evolutionary algorithms, SA is inspired by a natural phenomenon. As the algorithm name states, it simulates the cooling and annealing of liquid material. When a liquid material cools and anneals quickly, the material will solidify into a sub-optimal configuration. However if the liquid material cools slowly, the crystals within the material will solidify optimally into a state of minimum energy (i.e. ground state).

The algorithm steps are outlined in algorithm 4. Despite the differences of concept, when comparing SA with stochastic hill climber, one cannot ignore the similarities. Nevertheless, in the SA the cooling schedule introduces the ability to explore broader solutions in the beginning, when the temperature is higher and perform almost like hill climber for very low temperatures. This makes cooling schedule extremely important to the performance of the algorithm.

Algorithm 4 – Single Objective Simulated Annealing

```
1     T = Tmax
2     u = Random Solution
3     fu = evaluate(u)
4     while T < Tmin do:
5       v = neighbor(u)
6       fv = evaluate(v)
7       if fv < fu
8         u = v
9       else if rand(0,1) < exp((fu – fv)/(fu.T))
10        u = v
11      T = update(T)
12    return u
```

In theory, an infinitesimal decrease of T over time leads to the global optimum solution. From o practical point of view, there are some important aspects to consider when devising the cooling schedule. If the initial temperature is to low or temperature drops too fast premature convergence occurs, if the initial temperature is too high or temperature drops too slowly finding the solution take longer. This is the main tradeoff when designing the cooling schedule. One common scheduling procedure is the exponential cooling:

$$T_{K+1} = \frac{T_1}{T_0}^{K} T_K \ . \tag{3.4}$$

Because the nature of simulated annealing is to explore the neighborhood of single tentative solutions, the adaptation to multi-objective requires changes in the structure of the search. The straightforward approach is create a weighted combination of the objectives, and find the set of Pareto optimal solutions with multiple runs of the SA with different weights, but finding the correct weights is complex. Another such adaptation can be found in [59], where the adaptation to the multi-objective case is done using an archive that stores the best solutions and the acceptance of a new solution is based on the dominance with respect to the archive not just the current solution. However by exploring the neighborhood of just one solution at a time the diversity of the solutions found suffers.

The MOSA implementation in this work also follows an archive-based multi-objective simulated annealing technique, but exploring the neighborhood of the entire archive at each iteration, instead of only a single point. In this way the diversity of the solutions explored is increased, another practical advantage is that the simulation of multiples circuits in a batch is much more efficient that simulation a circuit at a time.

The algorithm starts with P elements in the archive instead of only one, to increase the potential for more annealing branches. The acceptance is still made by metropolis criterion and the archive is trimmed by non-dominated sorting and crowding distance when its size exceeds the maximum elements permitted. A Pareto set with all the non-dominated solutions found is also kept, but since the new elements are neighbors of the original solution, the Pareto archive grows very fast in solutions that are

very close to each other, so a crowding distance criterion is also used to trim the Pareto set. The implemented MOSA is shown in algorithm 5, where instead of using *T*min to control the annealed schedule, a maximum number of iterations to control the termination is used.

Algorithm 5 – Multi-Objective Simulated Annealing

```
1    T = Tmax, iteration = 0
2    archive = P Random Solution
3    evaluate(archive)
4    pareto = non-dominated-solutions(archive)
5    while iteration < N do:
6      neighbors = neighbor-foreach-element(archive)
7      evaluate(archive)
8      foreach element,neighbor u,v in archive,neighbors
9        if v dominates u
10         v replace u in archive
11       else if u not-dominates v
12         keep u, add v to archive
13       else if rand(0,1) < exp(|f(u) – f(v)|/(|fu|.T))
14         v replace u in archive
15     pareto = non-dominated-solutions(pareto + archive)
16     crowding-distance-trim(pareto)
17     crowding-distance-trim(archive)
18     T = update(T), iteration++
19   return pareto
```

3.5 MOPSO

The Particle Swarm Optimization algorithms (PSOs) introduced by Kennedy and Eberhart in 1995 [60] are partly inspired by the behavior of large animal swarms such as schooling fish, flocking birds or honey bees. PSO associates each particle as a candidate solution and lets them explore the search space. This technique is focused on the collective behavior of a distributed population of simple agents that interact locally with each other.

Each particle is associated with a stochastic velocity vector which indicates where the particle is moving to. The next move of each particle at a given time is a stochastic combination of the velocity in the previous time instant, of the direction to the particle's best position, and of the direction of the best swarm positions. In the standard model each particle *i* is associated with a current position (x_i) in the search space, a velocity (v_i), the position (p_i) and fitness of the best point encountered by the particle, and the rank (g) to the best particle in the swarm. The interaction between these variables is commonly governed by the following rules:

$$\begin{cases} \vec{x}_{i+1} = \vec{x}_i + \vec{v}_{i+1} \\ \vec{v}_{i+1} = x(w\vec{v}_i + \vec{\varphi}_{i1}(p_i - \vec{x}_i) + \vec{\varphi}_{i2}(p_g - \vec{x}_i)) \\ \vec{\varphi}_{i1} = N(0, \vec{\varphi}_{1\max}) \\ \vec{\varphi}_{i2} = N(0, \vec{\varphi}_{2\max}) \end{cases} \tag{3.5}$$

where, x is the constriction coefficient, w is the inertia weight, and p_g is the position of the best particle in the swarm. The vectors φ_1 and φ_2 are randomly generated for each particle with entries uniformly distributed between 0 and φ_{1max} or φ_{2max} respectively.

The manipulation of some of these parameters develops other variations of the standard algorithm. Controlling the velocity and the direction to the particle's best position allows the implementation of schemas for exploitation and exploration of the search space.

The MOPSO follows the implementation described by [61], using external archive, turbulence, and a fully connected topology and density estimator (crowding distance). In the single objective one of the critical factors in the implementation of a PSO is the selection of the leader, in the multi-objective case the issue persists, even worsening, as there are many options to select the leader. The method selected was to randomly select a solution from the Pareto to increase the pressures for improvement, other possibilities, such as selecting a random solution from the non-dominated set of particles using crowding distance tournament between two solutions to select the leader. The pseudo-code for the MOPSO is shown in algorithm 6.

Algorithm 6 - MOPSO

```
1    step = 0
2    particles = P new random solutions
3    evaluate(particles)
4    pareto = non-dominated-solutions(particles)
5    while step < N do:
6      foreach particle p in particles
7        l = select-leader(pareto)
8        update-particle(p, l)
9        apply-turbolence(p)
10     evaluate(particles)
11     pareto = non-dominated-solutions(pareto + particles)
12     crowding-distance-trim(pareto)
13     step++
14   return pareto
```

3.6 Multi-Kernel algorithm

With the algorithms previously described developed in the platform, new optimization methods that combine their techniques can be explored. By combining and reusing the diverse strategies it is reasonable to assume that is possible to take advantage of their diverse strong points. By a careful implementation of the support framework, these algorithms can be experimented easily.

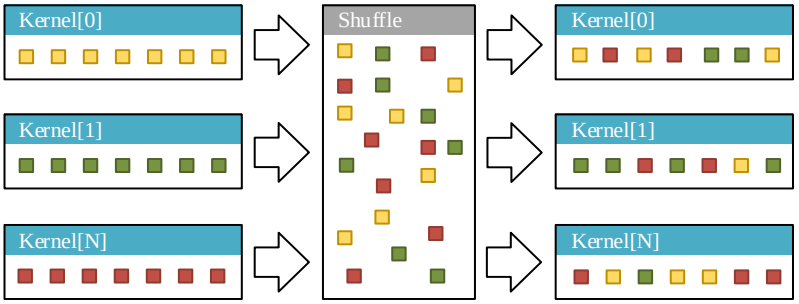
One possible combination is to use multiple algorithms in parallel, as shown in Algorithm 7, sharing the elements to solve the problem more efficiently. The parallel combination uses a pool of elements that is divided between each kernel and evolved using a different approach. Each time is deemed to rearrange the elements between the kernels, if `is-merge-step(step)` is true, the pool of elements is redistributed among the kernels.

Algorithm 7 - Parallel Multi-Kernel

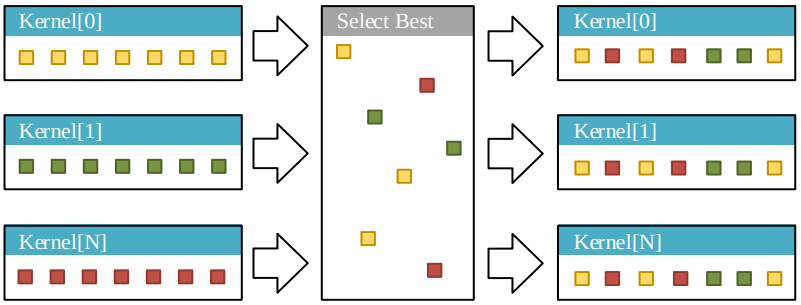
```
1    step = 0
2    archive = P new random solutions
3    evaluate(archive)
4    pareto = non-dominated-solutions(archive)
5    while step < N do:
     // merge elements
9      if is-merge-step(step)
11       foreach kernel k in kernels[]
           redistribute-elements(k, archive)
     // execute optimization step
6      foreach kernel k in kernels[]
7        execute-kernel-step(k)
       archive = collect-elements(kernels[])
15     pareto = non-dominated-solutions(pareto + archive)
16     crowding-distance-trim(pareto)
18     step++
19   return pareto
```

The major decisions in this method are when to redistribute the elements and how the redistribution is done. A simple method to select when to rearrange the elements is to rearrange the samples at uniform periods of time, i.e., at each fixed number of steps. This was the method implemented, but more complex methods can be devised and are easy to add to the platform.

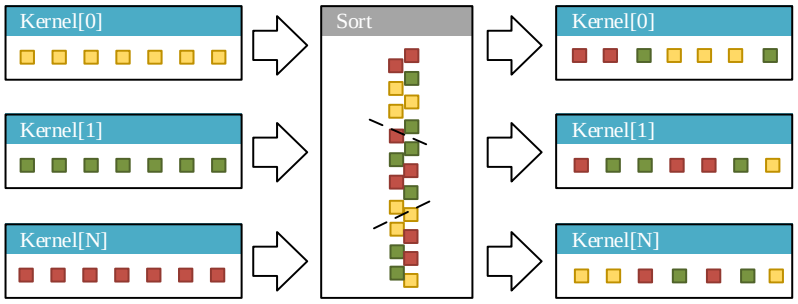
Regarding the redistribution itself, the three methods illustrated in Figure 3-7 were considered. In Figure 3-7 (a) is presented a simple version that shuffles and reassign the elements to the different kernels taking advantage of the different explorations techniques. In Figure 3-7 (b) a more greedy approach is used, following the same principle of using different methods to explore the same space but that selects only the best individuals using rank and crowding distance, and setting the same individuals to all the kernels. Another approach is shown in Figure 3-7 (c), where the elements are sorted using some criteria and split in blocks allowing the algorithms to explore different regions of the search space.



(a) shuffle



(b) Best



(c) Sorted

Figure 3-7 - Different methods to redistribute elements

A different method to combine the kernels is sequentially, where the results from one kernel is passed as input to the next, as shown in Algorithm 8. With this combination it is possible to optimize a problem for example using 400 cycles using NSGAII and then 100 cycles of MOSA, for fine tuning of the solutions. Both multi-kernel approaches use an external Pareto set that is used to store the best solutions achieved during the several iterations, even if they are not the elements being considered currently in the kernels.

By combining these kernels, the framework flexibility is further enhanced and there is the possibly to find combinations of the different methods that may be better that the individual kernels. However, the objective in this work is to provide the infrastructure to such a study and not to conduct that study itself. Nevertheless, some of these approaches were experimented to show that the tool can now take advantage of such advanced combination of methods.

Algorithm 8 - Sequential Multi Kernel

```
1     step = 0
2     archive = P new random solutions
3     evaluate(archive)
4     pareto = non-dominated-solutions(archive)
      k = 0
5     while step < N do:
        execute-kernel-step(kernels[k])
      // change elements
9       if is-switch-step(step)
          copy-elements(kernels[k], kernels[(k + 1)%K])
           k= (k + 1)%K]
        archive = collect-elements(kernels[k])
15      pareto = non-dominated-solutions(pareto + archive)
16      crowding-distance-trim(pareto)
18      step++
19    return pareto
```

3.7 Conclusions

In this chapter AIDA frameworks is introduced, and AIDA-C, the preceding optimization based circuit sizing tool, is described. The architecture for the proposed tool AIDA-CMK is defined, taking into consideration how the circuit sizing is handled and optimization problem, even when considering the extra constraints introduced by PTV corners.

The multi-objective optimization kernels, NSGA-II, MOPSO, MOSA and the two multiple kernel hybridization implemented in AIDA-CMK are described here. Asides the infra-structure of the optimization kernels framework, that will be described in chapter 4, AIDA-CMK supporting these algorithms are the new contributions to the AIDA framework.

4 Multi-objective framework implementation

In this chapter the details about the framework structure are presented, showing the application layers and their implementation. The structure of classes is described in detail showing their relations and the flexibility of the proposed framework.

4.1 AIDA-CMK framework structure and design implementation

The approach to the optimization kernel of AIDA-CMK is structured in a multi-layer format considering three different layers, as illustrated in Figure 4-1. The first layer implements the problem evaluation, the second layer implements the problem abstraction and finally, the third layer, the optimization cycle.

In this design the adaptability is reinforced by making the entire Optimization Kernel a self-contained module with well-defined interface to the exterior. One of the targets of the implementation and architecture was the improved capability of extension and maintenance of the framework code. In order to achieve those objectives some design patterns were used such as layer, repository, observer and model view controller.

The interface implementation between optimizer and problem follows a repository design using the optimization element as data representation common to all algorithms. This design has the advantage that every element can be used by any algorithm, but puts some effort to maintain the data representation in order to be usable by all algorithms. Also allows the optimizers to be used in another types of optimization, such as the AIDA-L routing optimization.

The framework is centered around the abstraction layer that implements the general multi-objective problem. This interface can be instantiated as a circuit problem or as a mathematical problem, and each of those problems are evaluated using their own evaluation method, the circuit simulator or a java function, respectively. With this abstraction it is possible to test the framework with known mathematical problems in a fast and simple manner. Having the multi-objective problem in one layer and the evaluation function in another allows the problem modulation to be separated from evaluation method, making possible to switch easily between circuit simulators or other circuit evaluation methods, while sharing the operators when the solution representations are equivalent.

By placing the responsibility of the operators' implementation in the interface between the problem and the optimizer layers, all kernels may use the same elements. This also turns the framework more versatile to new algorithms implementations by centering all the operators in the same interface. This abstraction take advantage of optimization element generalization of the various algorithms operators, allowing the elements to be used by all algorithms and the optimizer abstraction layer makes it simple for AIDA-CMK to switch between algorithms, since all optimizers are handled equally by the optimization kernel.

Other major advantage of this design is the simplicity to implement other algorithms and strategies. The implementations of another algorithm is done in two steps, first the enhancement of the optimization element with the operators of the new algorithm and second the creation of another implementation of

the abstract optimizer. By completing these two steps, the new kernel is ready to be applied to circuit optimization and also usable by the hybrid kernel.

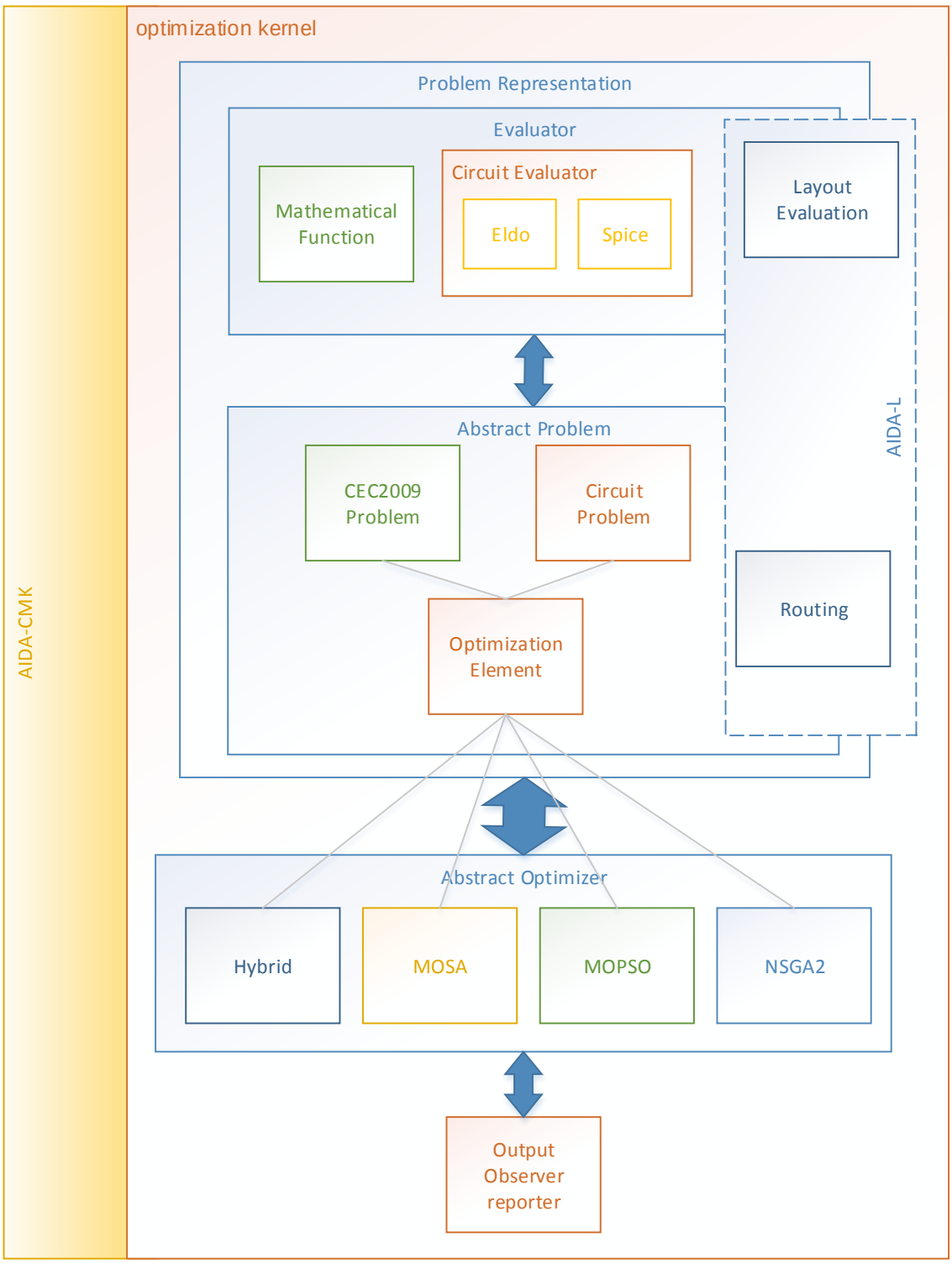


Figure 4-1 - Framework structure

The framework can also handle single objective algorithms using a vector of weights to convert a multi-objective problem in a single objective problem. However, this implementation wasn't explored in the circuit design problems since the circuit is converted only to a multi-objective problem. Also another implementation of the optimization elements is present in the AIDA framework, this implementation is

used in the AIDA-L module of the framework to make the interface with the abstract optimizer and the routing problem. The routing element implements only the NSGA operators so it can only be used by that kernel. However, in this design, implementing the other algorithm operators would make other kernels available to AIDA-L in the optimization of the routing.

The framework also includes a reporting module to easy visualize the algorithms outputs (POF and some statistics) and the optimization elements implements a XML handler for easy write and read the population to and from an XML format file. The output of the optimization kernel is constructed based on the observer pattern in order to create all the statistics of the optimization, passing them to AIDA application using a model view controller design.

To implement the design layers indicated above, the framework is supported by the following key abstract classes and interfaces: IEvaluator, IOptimizationSubject, and AbstractOptimizer, which are described in the following sections.

4.2 Abstract optimization kernel

The AbstractOptimazer class supply to the optimizer kernel, illustrated in Figure 4-2, functions to manipulate a pool of elements, reporting functionalities and multi-threading capabilities.

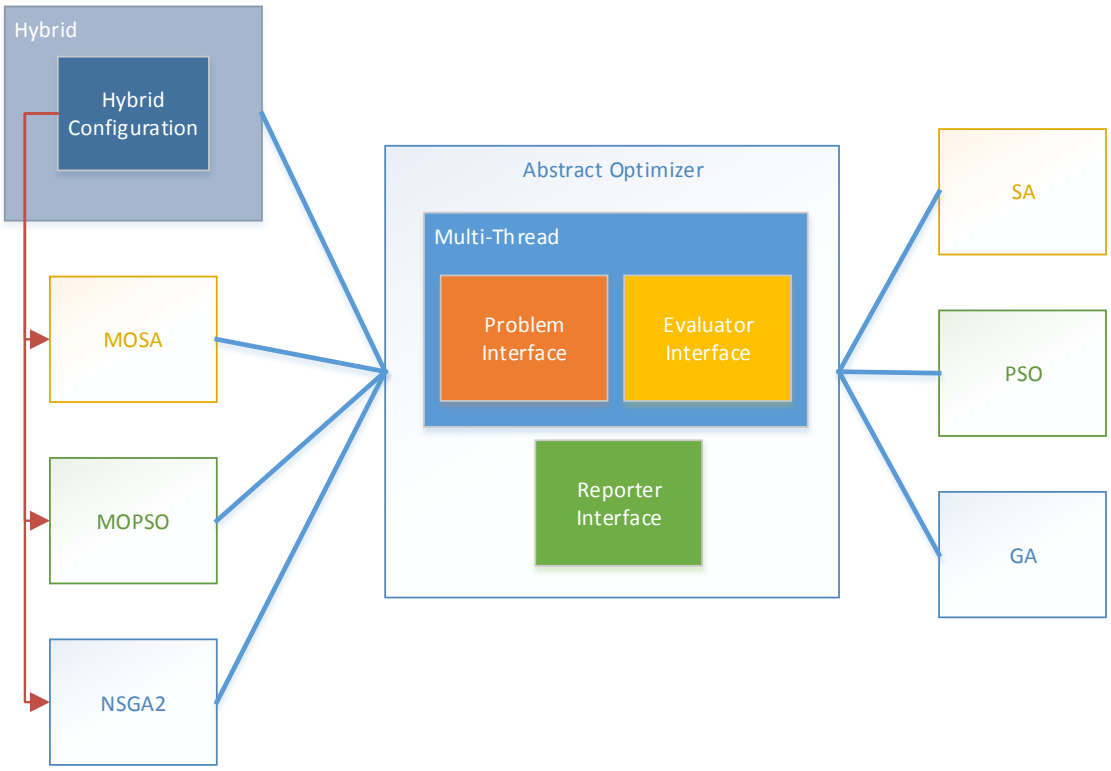


Figure 4-2 - Abstract optimizer kernel

The AbstractOptimizer implements the generic main loop of the algorithms. Leaving the specific implementation to be defined in the overwritten functions:

```
        initializePool()
        evaluateInitialPool()
        updateAfterEval()
        selection()
        while (true){
                evaluatePool()
                updateAfterEval()
                selection()
                if (stopCondition() ){
                        break
                }
}
```

The AbstractOptimizer class is also responsible to maintain the output information and update the algorithm statistics such as number of evaluations and number of elements on the Pareto front. This information is stored in order to make a graphical representation of the Pareto front and it's evolution interactively during the simulation that is plotted using the linux tool gnuplot.

The UML diagram illustrated in Figure 4-3 describes in summary the classes and interfaces used to implement the abstract kernel. Identifying the interface, used by AIDA application, IOptimizer and the three interfaces used by the kernel to communicate with the different components, namely, IOptimizationProblem, IEvaluator and IOptimizationMonitor. These three interfaces assure the integrity of the communication between each layer. Is also presented the abstract methods of the AbstractOptimizer class and all the implemented algorithms as an extension of the class.

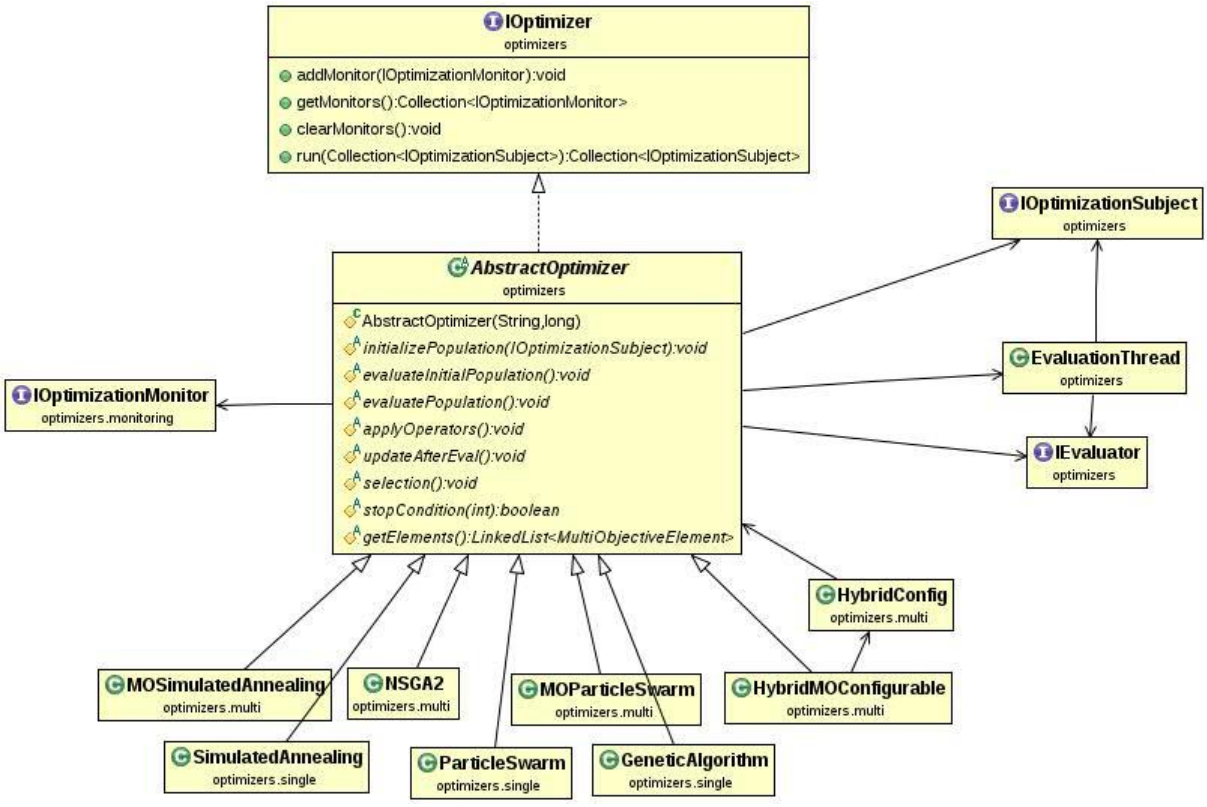


Figure 4-3 - Abstract optimizer UML

With all the output management and multi-threading capabilities inside the abstractOptimizer, the algorithm implementations need only to deal with specific details of the algorithm, easily allowing new developments for the extension of the existing framework. The optimization elements capability to be evaluated based on all the objectives or a combination of them, uses a weighted vector to archive that,

making it possible to implement multi-objective and single objective algorithms. In this work, the focus was on the multi-objective algorithms given the existent mapping of the circuit problems formulated as multi-objective in AIDA-CMK

The architecture of the abstract optimizer was made to take advantage of a pool of elements. That pool can be used at any iteration by any algorithm technique. Making it possible to a multi-kernel algorithm manage the evolution of the simulation. The optimizer can evolve the elements using multiple strategies and paradigms. To do that a list of optimization kernels must be defined in a HybridConfig settings as well as the merge strategy to be used in the algorithm. The HybridConfig also indicates the way the hybrid kernel should use the kernels, sequential or parallel, and the selection method to be used in the parallel execution when the pool is merged and splitted.

4.3 Problem representation

The problem representation, illustrated in Figure 4-4, is another main part of the optimization kernel by abstracting the DoubleValueProblem, which implements the IOptimizationSubject interface, delegating to each problem implementation the responsibility of implementing or communicate with the evaluation function. So it is transparent to the Optimization Kernel of AIDA-CMK the optimization of mathematical problems or circuit problems. This implementation is also responsible to implement the algorithms operators. DoubleValueInputProblem class is extended by the circuitOptimizationProblem and also the mathematical problems. By doing this abstraction is also possible the use of AIDA-CMK optimization kernel in others modules of AIDA Framework namely AIDA-L, as previously referred, which also implements the IOptimizationSubject.

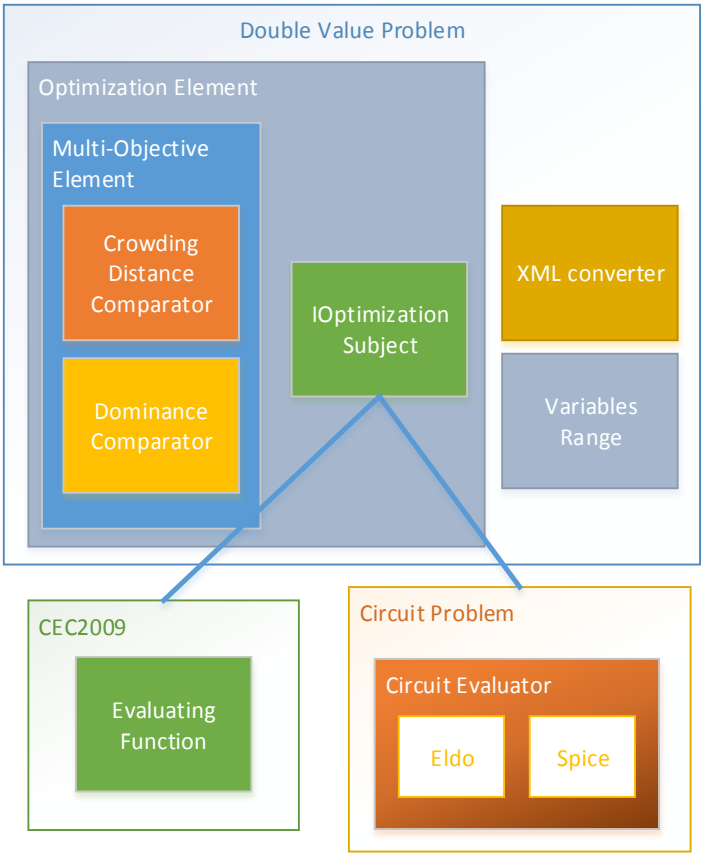


Figure 4-4 - Problem representation

The UML presented in Figure 4-5 shows the class structure of the problem, focusing the importance of the IOptimizationSubject as interface between the optimizer and the problem implementation, making possible to use the same optimizer both in circuit sizing and routing. The UML also shows the abstract DoubleValueInputProblem creates an abstraction layer and a common base for the mathematical and circuit problems. Another important interface and implementation presented are the IEvaluator and CircuitEvaluator. These classes are responsible for the implementation of the connection between the optimizer and the circuit simulator. As referred above the XML interface is also presented. This interface ensure the coherent handling of all the data present both in the optimizationElement and MultiObjectiveElement.

The optimization elements interface are defined by the IOptimizationSubject. This interface is implemented by DoubleValuedInputProblem and Routing. The benchmark problems and circuit problems then extended the DoubleValuedInputProblem. Because the multi-objective algorithms need more information on each element the MultiObjectiveElement was created to accommodate and manipulate the needed information such as crowding distance and dominance. The MultiObjectiveElement has an IOptimizationSubject element inside to represent the problem to be optimized either a mathematical or circuit problem.

The CircuitOptimizationProblem is responsible to map the circuit as a multi-objective problem and the CircuitEvaluator to implement the interface to the circuit simulator. This level of abstraction makes the framework very versatile because it disassociates the implementation of the evaluation function from the algorithm that uses that function, making easy to change the method used to evaluate the circuit by changing the circuit simulator.

Another important feature of the problem representation is the ability to be converted from and to XML, making the elements capable of being saved in a persistent form on the hard disk during the simulation, and also, recover the simulation from a saved checkpoint. This is important in the circuit optimization because the simulation is a time consuming task and failures can occur, so a recuperation mechanism is an important feature to have.

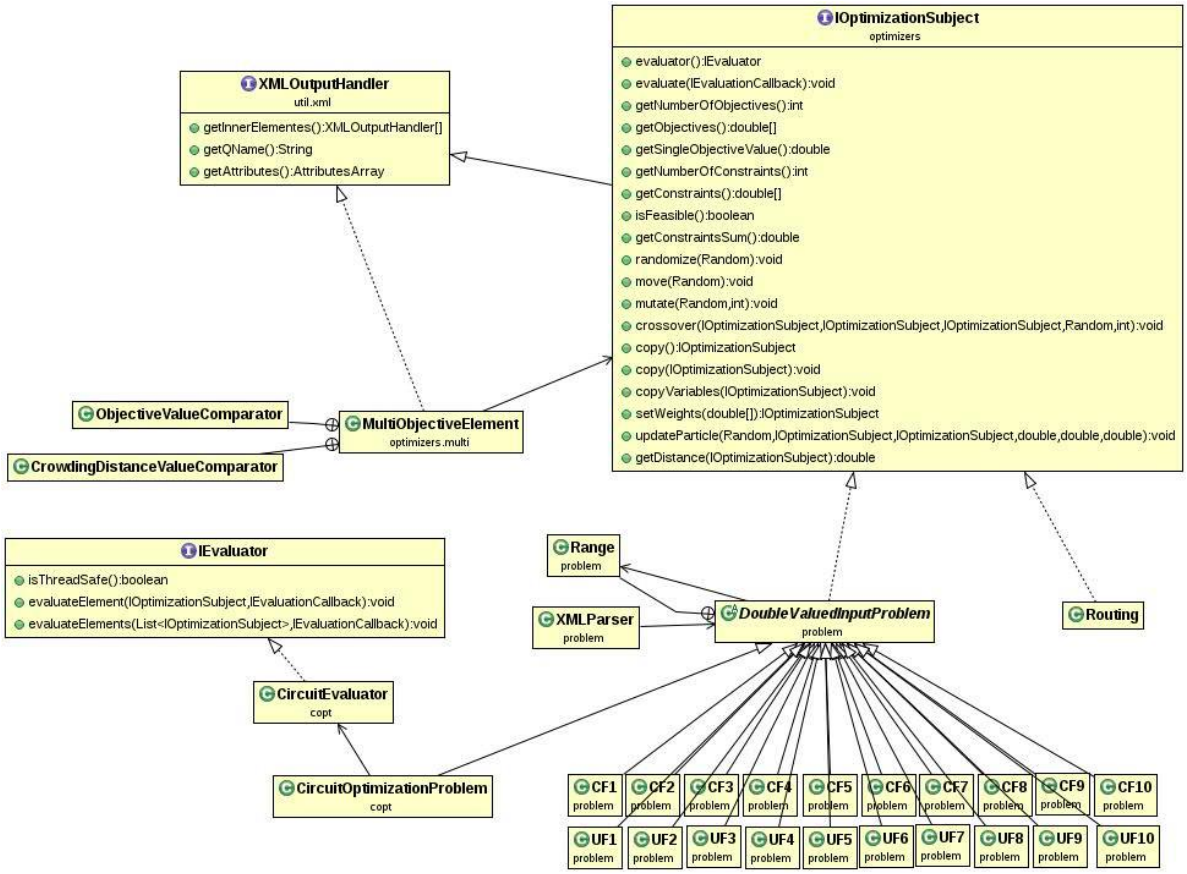


Figure 4-5 - Problem representation UML

4.4 Output and reporting

The framework also have a reporting and graphical module. This module structure is illustrated in Figure 4-6 take advantage of the observer pattern to store and print the simulation data in a graphical form, turning possible to observe the evolution of the simulation using GnuPlot. The output module can also print boxplot graphics given a group of simulations.

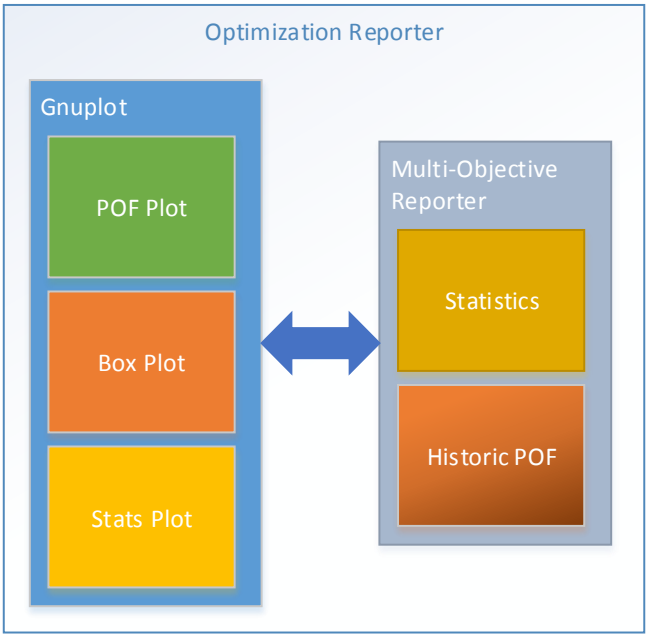


Figure 4-6 - Output and reporting

As this framework is part of AIDA the optimization must have interface with AIDA frontend, as illustrated in Figure 4-8. This output reporting module is very important to maintain the model view controller pattern used by AIDA. The monitor is instantiated by two components as illustrated in Figure 4-7, the ResultsPanel and MultiObjectiveReporter, both used in AIDA application to show the simulation results (A) and graphics (C) respectively illustrated in Figure 4-8.

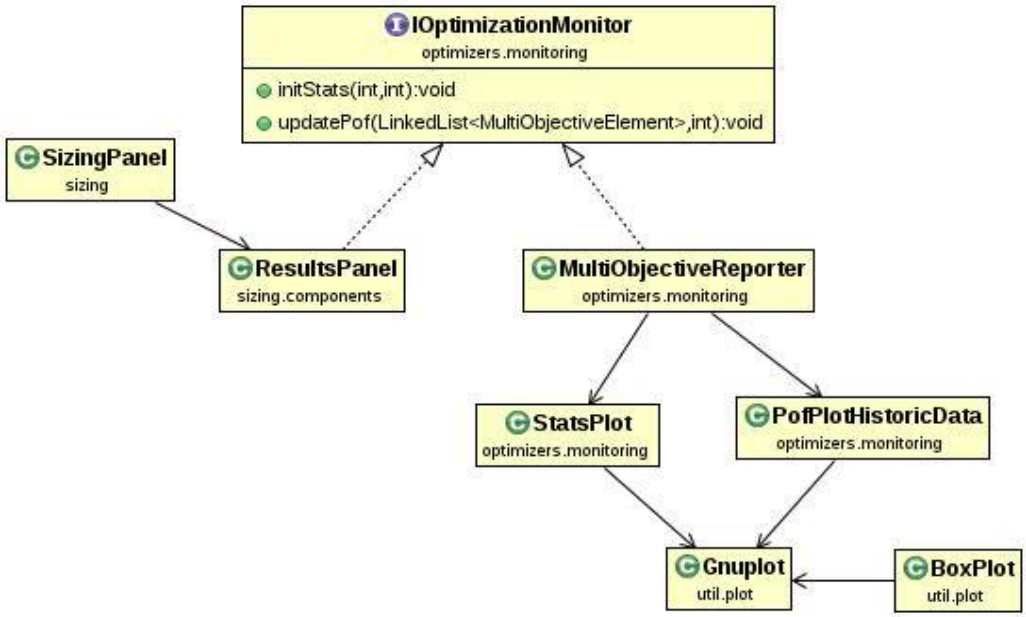


Figure 4-7 - Output and reporting UML

Additionally, several controllers where implemented to configure the kernels. Those controllers are used in the dialog that set the algorithm parameters, shown in Figure 4-8 (B).

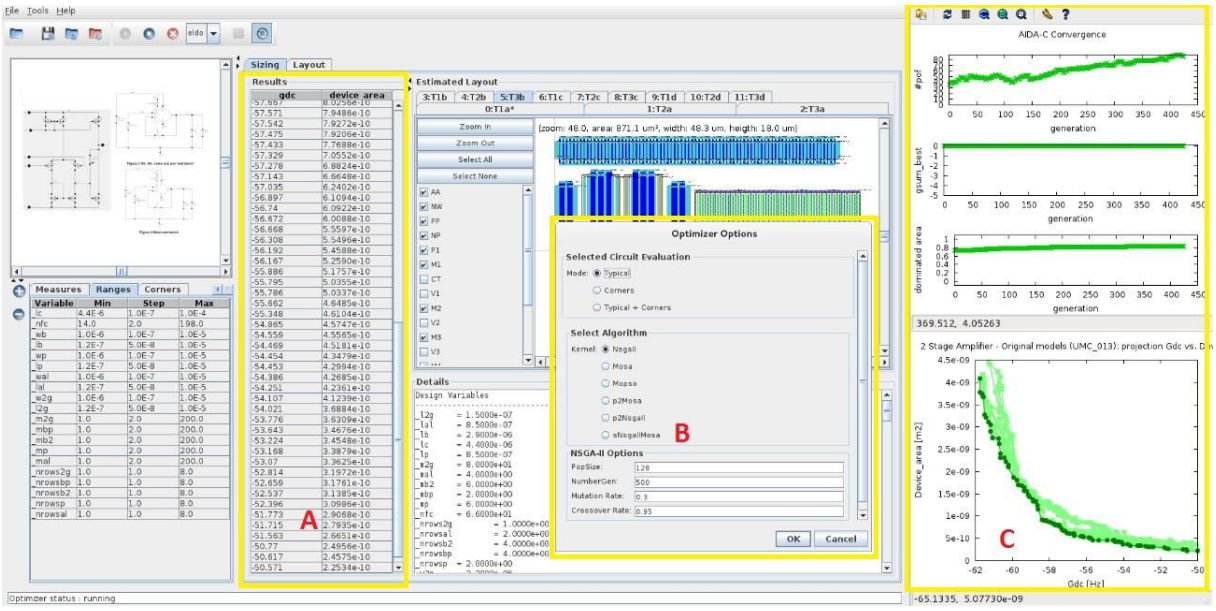


Figure 4-8 - AIDA frontend

4.5 Conclusions

In this chapter the framework design implementation is described. Is illustrated the design and class structure the extensibility capabilities of the framework are showed, indicating the advantages of each design pattern chosen to implement the framework.

5 Kernel validation using CEC2009 benchmarks

To test and empirically tune the algorithms with known functions, some of the CEC2009 competition [62] problems were considered. In the scope of this work and because the circuit is mapped as a constrained multi-objective problem usually with two objectives, the two objective constrained problems, CF1 to CF7, were selected.

5.1 Problem definition

The following paragraphs show problems definitions and the corresponding Pareto fronts are illustrated in Figure 5-1.

- **Constrained Function 1 - CF1**

  The two objective to be minimized are:

$$f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - x_1^{0.5(1+\frac{3(j-2)}{n-2})}\right)^2$$

$$f_2(x) = 1-x_1 + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - x_1^{0.5(1+\frac{3(j-2)}{n-2})}\right)^2$$

  where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$.

  The constraint is:

$$g_1(x) = f_1(x) + f_2(x) - \left|\sin\left(10\pi\left(f_1(x) - f_2(x) + 1\right)\right)\right| - 1 \ge 0$$

  The search space is $[0,1]^n$, where n is the number of variables.

- **Constrained Function 2 - CF2**

  The two objective to be minimized are:

$$f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - \sin\left(6\pi\cdot x_1 + \frac{j\pi}{n}\right)\right)^2$$

$$f_2(x) = 1-\sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - \cos\left(6\pi\cdot x_1 + \frac{j\pi}{n}\right)\right)^2$$

  where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$.

  The constraint is:

$$g_1(x) = \frac{t(x)}{1+e^{4|t(x)|}} \ge 0$$

$$\text{where } t(x) = f_2(x) + \sqrt{f_1(x)} - \sin\left(2\pi\left(\sqrt{f_1(x)} - f_2(x) + 1\right)\right) - 1$$

  The search space is $[0,1]\times[-1,1]^{n-1}$, where n is the number of variables.

- **Constrained Function 3 – CF3**

  The two objective to be minimized are:

$$f_1(x) = x_1 + \frac{2}{|J_1|}\left(4\sum_{j\in J_1} y_j(x)^2 - 2\prod_{j\in J_1}\cos\left(\frac{20y_j(x)\pi}{\sqrt{j}}\right) + 2\right)$$
$$f_2(x) = 1 - x_1 + \frac{2}{|J_2|}\left(4\sum_{j\in J_2} y_j(x)^2 - 2\prod_{j\in J_2}\cos\left(\frac{20y_j(x)\pi}{\sqrt{j}}\right) + 2\right)$$

where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$, $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$ and $y_j(x) = x_j - \sin\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right)$.

The constraint is:

$$g_1(x) = f_2(x) + f_1^2(x) - \sin\left(2\pi\left(f_1^2(x) - f_2(x) + 1\right)\right) - 1 \ge 0$$

The search space is $[0,1]\times[-2,2]^{n-1}$, where n is the number of variables.

- **Constrained Function 4 – CF4**

The two objective to be minimized are:

$$f_1(x) = x_1 + \sum_{j\in J_1} h_j(y_j(x))$$
$$f_2(x) = 1 - x_1 + \sum_{j\in J_2} h_j(y_j(x))$$

where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$, $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$, $y_j(x) = x_j - \sin\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right)$ and

$$h_2(t) = \begin{cases} |t| & \text{if } t < \frac{3}{2}\left(1 - \frac{\sqrt{2}}{2}\right) \\ 0.125 + (t-1)^2 & \text{otherwise} \end{cases}$$
$$h_j(t) = t^2 \text{ for } j = 2,3,\ldots,n.$$

The constraint is:

$$g_1(x) = \frac{t(x)}{1 + e^{4|t(x)|}} \ge 0$$
$$\text{where } t(x) = x_2 - \sin\left(6\pi.x_1 + \frac{2\pi}{n}\right) - 0.5x_1 + 0.25$$

The search space is $[0,1]\times[-2,2]^{n-1}$, where n is the number of variables.

- **Constrained Function 5 – CF5**

The two objective to be minimized are:

$$f_1(x) = x_1 + \sum_{j\in J_1} h_j(y_j(x))$$
$$f_2(x) = 1 - x_1 + \sum_{j\in J_2} h_j(y_j(x))$$

where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$, $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$,

$$y_j(x) = \begin{cases} x_j - 0.8x_1\cos\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_1 \\ x_j - 0.8x_1\sin\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right) & \text{if } j \in J_2 \end{cases}, \text{ and } h_2(t) = \begin{cases} |t| & \text{if } t < \frac{3}{2}\left(1 - \frac{\sqrt{2}}{2}\right) \\ 0.125 + (t-1)^2 & \text{otherwise} \end{cases}$$
$$h_j(t) = t^2 \text{ for } j = 3,4,\ldots,n.$$

The constraint is:

$$g_1(x) = x_2 - 0.8x_1\sin\left(6\pi \cdot x_1 + \frac{2\pi}{n}\right) - 0.5x_1 + 0.25 \ge 0$$

And the search space is $[0,1]\times[-2,2]^{n-1}$, where n is the number of variables.

- **Constrained Function 6 – CF6**

  The two objective to be minimized are:

$$f_1(x) = x_1 + \sum_{j\in J_1} y_j(x)^2$$
$$f_2(x) = (1-x_1)^2 + \sum_{j\in J_2} y_j(x)^2$$

  where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$, $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$, and

$$y_j(x) = \begin{cases} x_j - 0.8x_1 \cos\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right) \text{if } j \in J_1 \\ x_j - 0.8x_1 \sin\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right) \text{if } j \in J_2 \end{cases}.$$

  The constraints are:

$$g_1(x) = x_2 - 0.8x_1 \sin\left(6\pi \cdot x_1 + \frac{2\pi}{n}\right) - \text{sign}\left(0.5(1-x_1) - (1-x_1)^2\right)\sqrt{\left|0.5(1-x_1) - (1-x_1)^2\right|} \ge 0$$
$$g_2(x) = x_4 - 0.8x_1 \sin\left(6\pi \cdot x_1 + \frac{4\pi}{n}\right) - \text{sign}\left(0.25\sqrt{1-x_1} - 0.5(1-x_1)\right)\sqrt{\left|0.25\sqrt{1-x_1} - 0.5(1-x_1)\right|} \ge 0$$

  And the search space is $[0,1]\times[-2,2]^{n-1}$, where n is the number of variables.

- **Constrained Function 7 – CF7**

  The two objective to be minimized are:

$$f_1(x) = x_1 + \sum_{j\in J_1} h_j(y_j(x))$$
$$f_2(x) = 1-x_1 + \sum_{j\in J_2} h_j(y_j(x))$$

  where $J_1 = \{j \mid j \text{ is odd} \cap 2 \le j \le n\}$, $J_2 = \{j \mid j \text{ is even} \cap 2 \le j \le n\}$,

$$y_j(x) = \begin{cases} x_j - \cos\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right) \text{if } j \in J_1 \\ x_j - \sin\left(6\pi \cdot x_1 + \frac{j\pi}{n}\right) \text{if } j \in J_2 \end{cases}, \text{ and } \begin{matrix} h_2(t) = h_4(t) = t^2 \\ h_j(t) = 2t^2 - \cos(4\pi) + 1 \text{ for } j = 3,5,6,\ldots,n. \end{matrix}.$$

  The constraints are:

$$g_1(x) = x_2 - \sin\left(6\pi \cdot x_1 + \frac{2\pi}{n}\right) - \text{sign}\left(0.5(1-x_1) - (1-x_1)^2\right)\sqrt{\left|0.5(1-x_1) - (1-x_1)^2\right|} \ge 0$$
$$g_2(x) = x_4 - \sin\left(6\pi \cdot x_1 + \frac{4\pi}{n}\right) - \text{sign}\left(0.25\sqrt{1-x_1} - 0.5(1-x_1)\right)\sqrt{\left|0.25\sqrt{1-x_1} - 0.5(1-x_1)\right|} \ge 0$$

And the search space is $[0,1]\times[-2,2]^{n-1}$, where n is the number of variables.

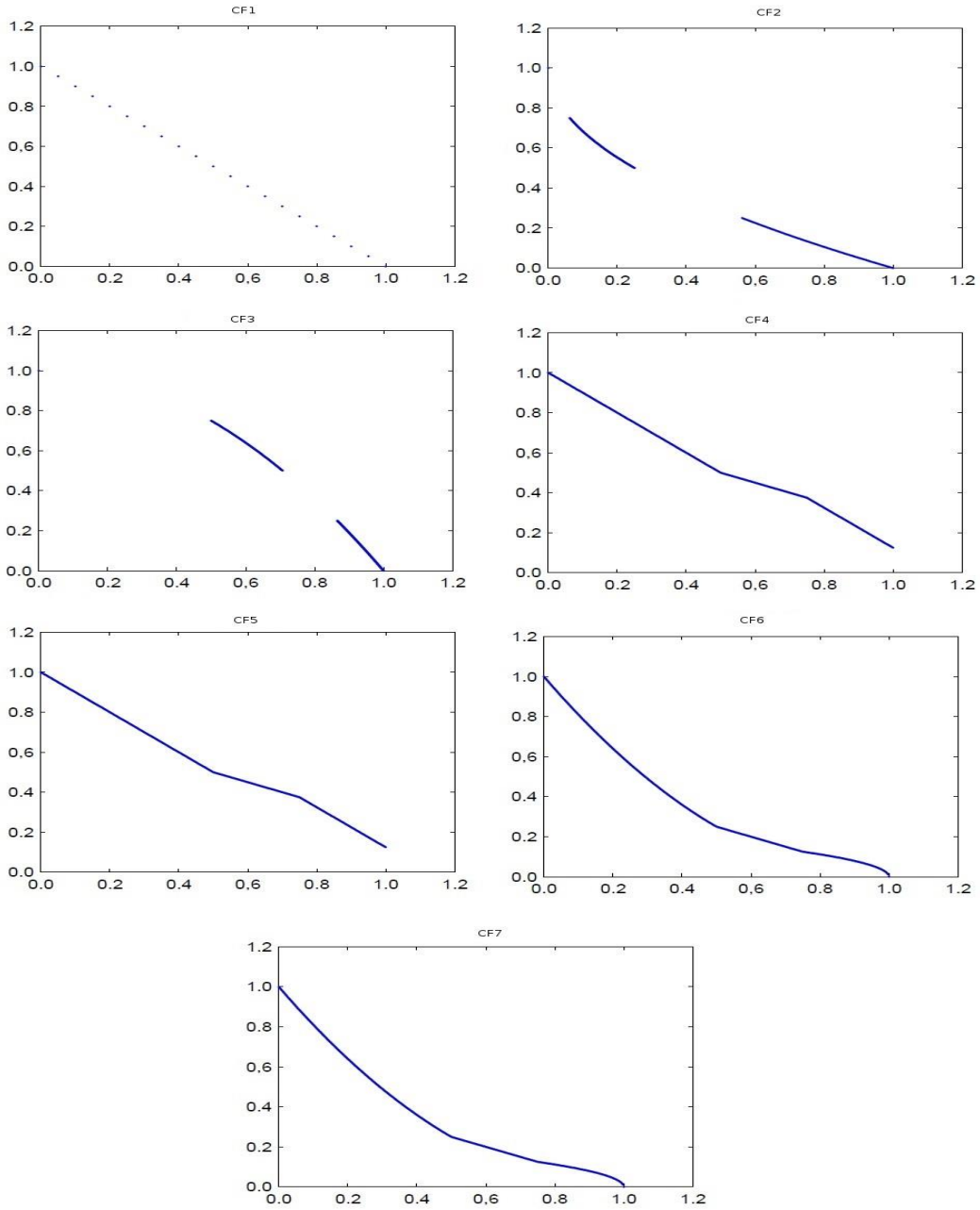


Figure 5-1 - CEC2009 Pareto fronts: CF1 to CF7.

5.2 Evaluation of the single kernel methods

To verify the behavior of the implemented algorithms and tune the algorithm parameters, several executions were conducted for the problems defined previously. For the initial executions the number of evaluations was selected to be around 300000 (as in the CEC2009 competition described in [62]) and n is set to 4 for all the problems.

The parameters of the Single Kernel Methods were set to values that are common in the literature and an empirical tuning was done. A more extensive study of the parameters influence by conducting systematic sweeps or parameter combination sampling to fine tune the parameters could have been done, but the fine tuning of specific algorithms is not the objective of this work.

Using those parameters several simulations were executed for the CEC2009 CF1-CF7 problems with 4 variables. The number of evaluations considered was 294400, leading to a population size of 128 and 2300 generation in the NSGAII, equivalently a swarm size of 128 and 2300 steps for the MOPSO and archive size of 128 and 2300 iterations for the MOSA. The attained results are illustrated in Figure 5-2. From the analysis of the plots is clear that all the algorithms performance in these problems is very similar.

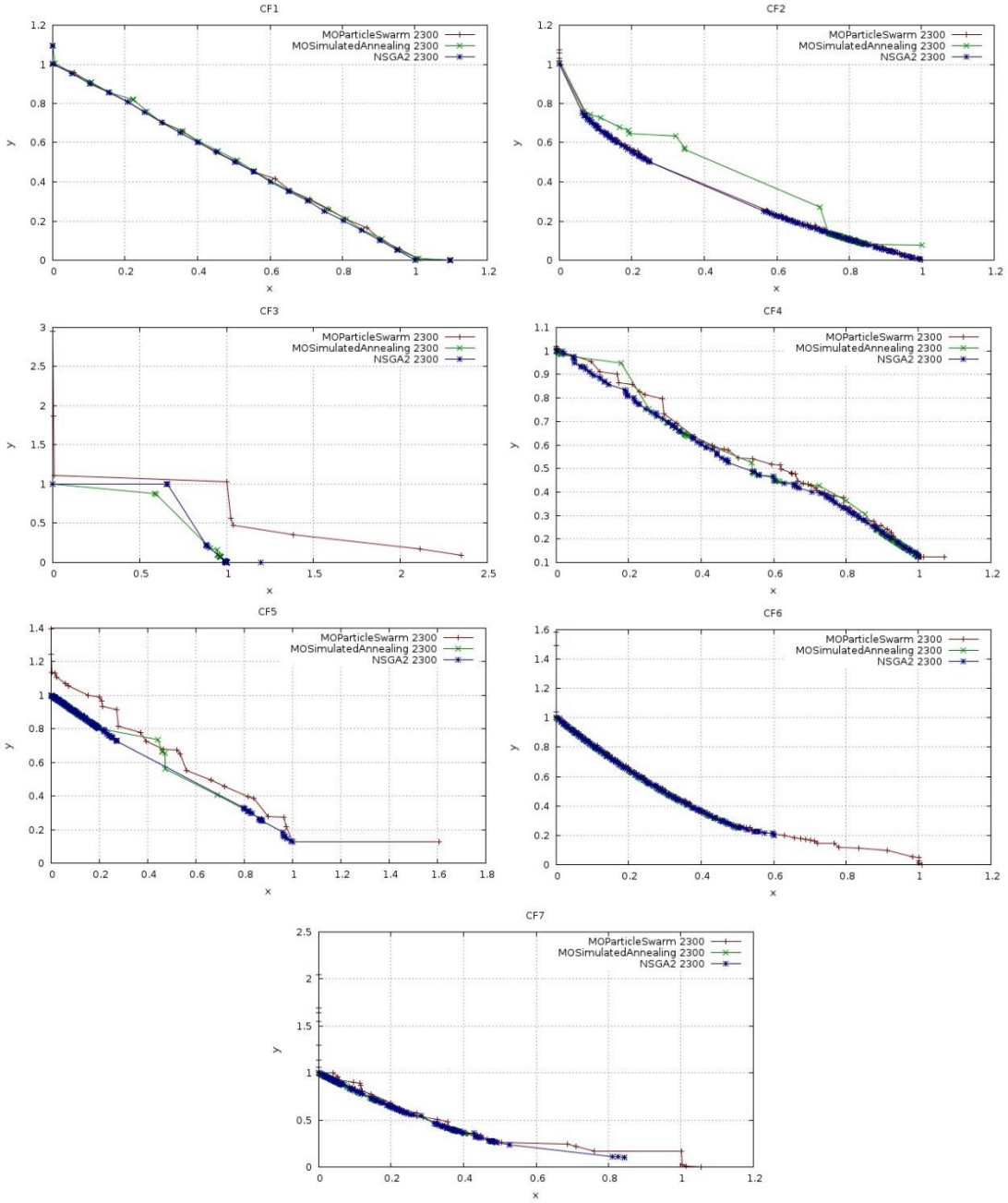


Figure 5-2 - Pareto fronts for problems with 4 variables: CF1 to CF7.

In order to do the simulations in conditions closer to that of the circuit problems, normally defined with 20 to 30 variables, the same problems were (re)defined with n set to 30 and (re)simulated in the same conditions. The result of those simulations is shown in Figure 5-3. The analysis of the obtained fronts shows a notorious degradation of the MOPSO performance, when dealing with problem of larger

dimension. The degradation in the CF1 and CF2 problems is not that large, but is notorious for the other problems (CF3 to CF7).

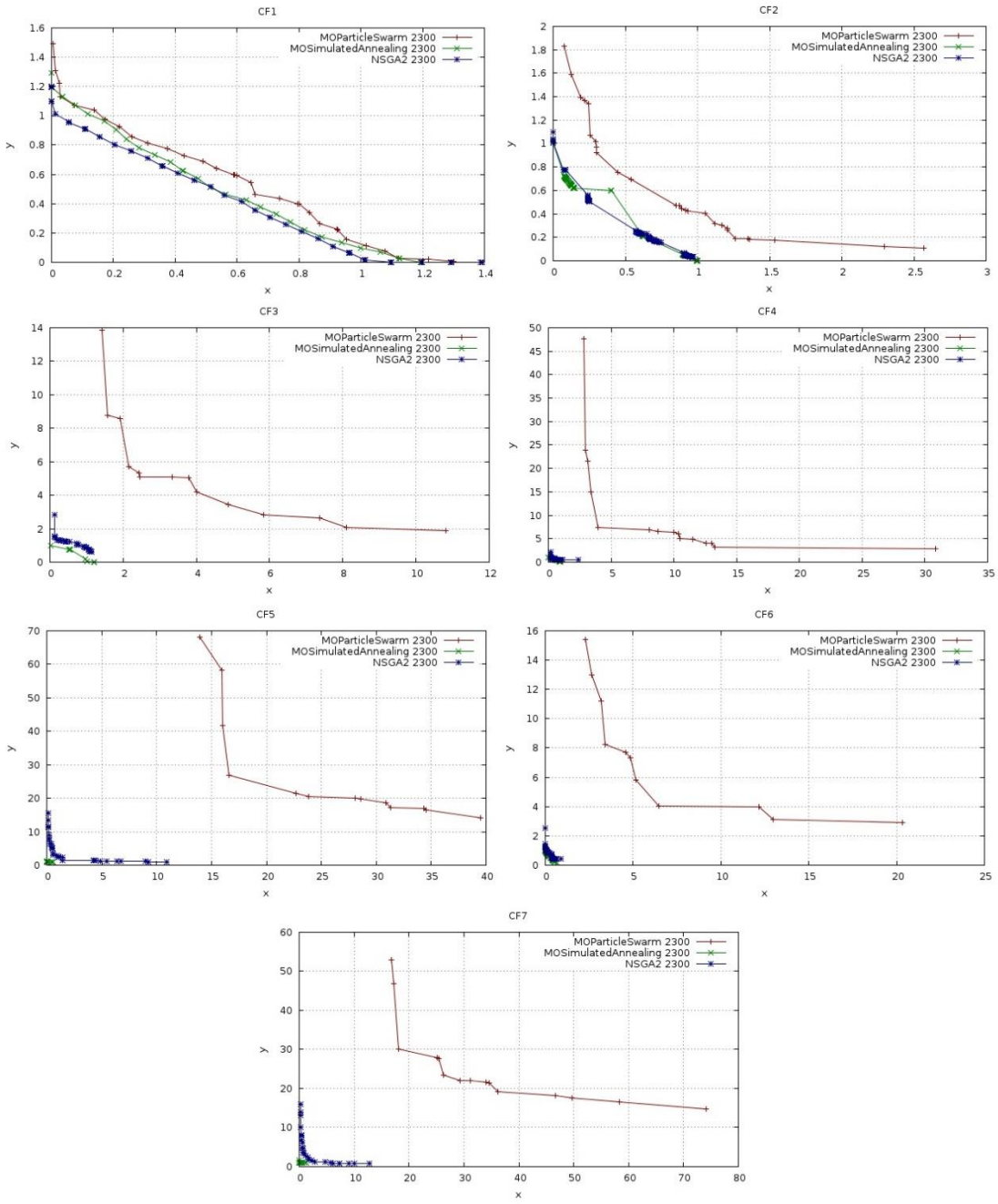


Figure 5-3 - Pareto fronts for problems with 30 variables: CF1 to CF7.

Regarding both MOSA and NSGAII the performance looks similar in Figure 5-3. However if the MOSPO is removed, the closer look shows that the implemented MOSA greatly outperforms the NSGA-II in CF3–CF7, as illustrated in Figure 5-4 for some the CF4-CF7 problems.

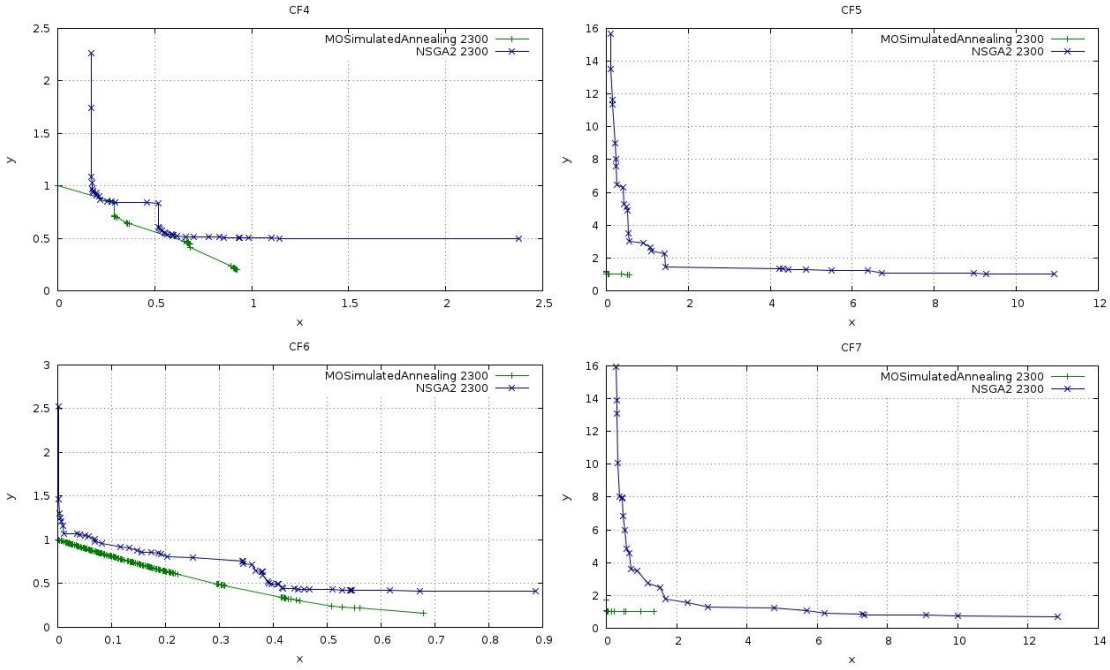


Figure 5-4 - Detail Pareto fronts for CF4-7 problems with 30 variables (MOSA and NSGAII)

5.3 Evaluation of the multi kernel methods

To experiment the multi-kernel methods and evaluate the performance of the merge operator in the parallel multi algorithm kernel, two combinations are considered: first with two NSGAII, and secondly with a MOSA and a NSGAII. The resultant POF shown in Figure 5-5 and Figure 5-6, the configurations used are described in Table 5-1

Table 5-1 - Merge operator parallel configurations

	Kernel1	**Kernel2**	**Merge**	**Strategy**
pNSGA²	NSGAII(64,2300)	NSGAII(64,2300)	step%30 == 0	Parallel-Split Sorted
pNSGA²	NSGAII(64,2300)	NSGAII(64,2300)	step%30 == 0	Parallel-Split Shuffle
pNSGA²	NSGAII(64,2300)	NSGAII(64,2300)	step%30 == 0	Parallel-Select Best
pMOSA-NSGAII	MOSA(64, 2300)	NSGAII(64,2300)	step%30 == 0	Parallel-Split Sorted
pMOSA-NSGAII	MOSA(64, 2300)	NSGAII(64,2300)	step%30 == 0	Parallel-Split Shuffle
pMOSA-NSGAII	MOSA(64, 2300)	NSGAII(64,2300)	step%30 == 0	Parallel Select Best

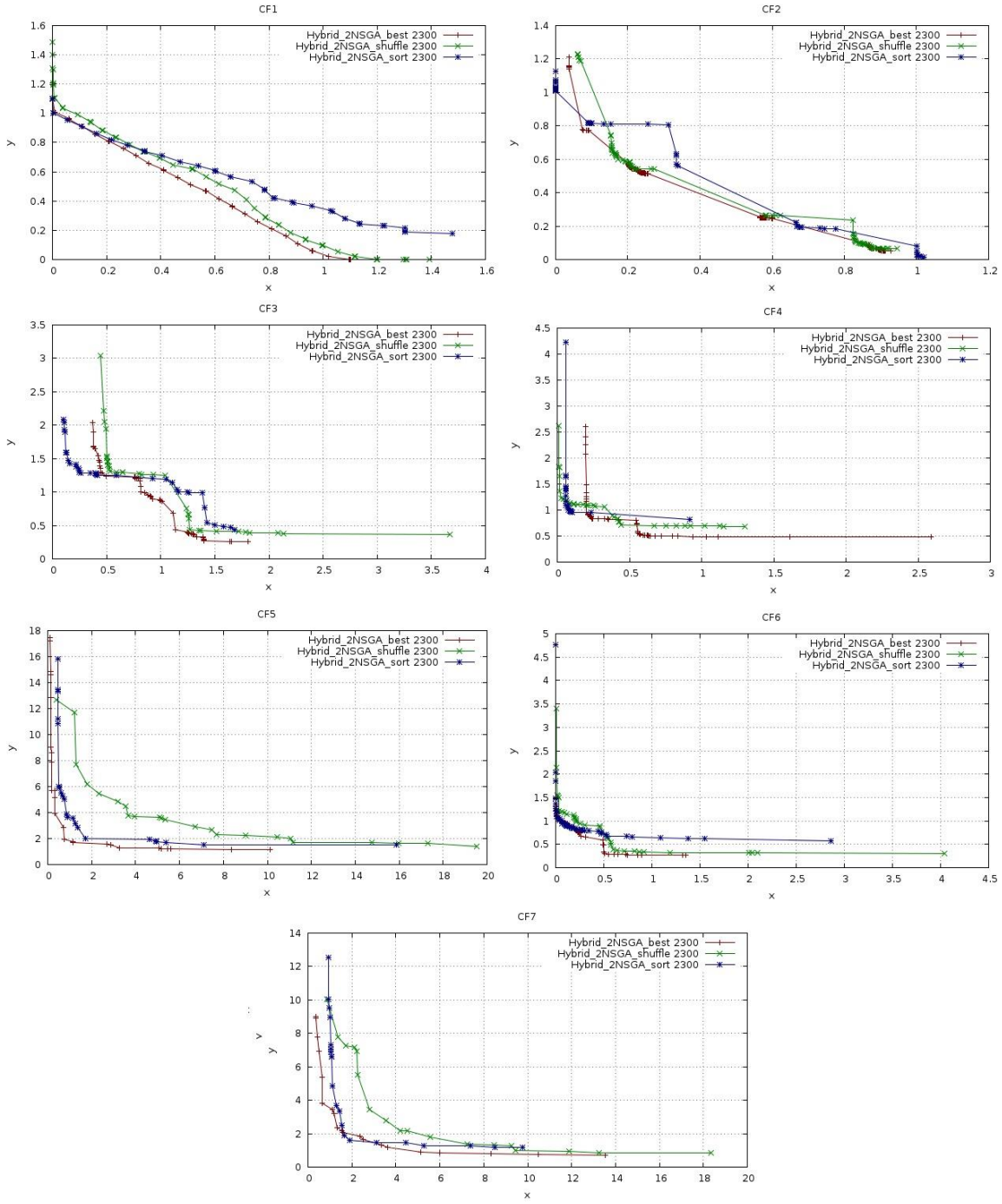


Figure 5-5 - Two parallel NSGA-II: CF1 to CF7.

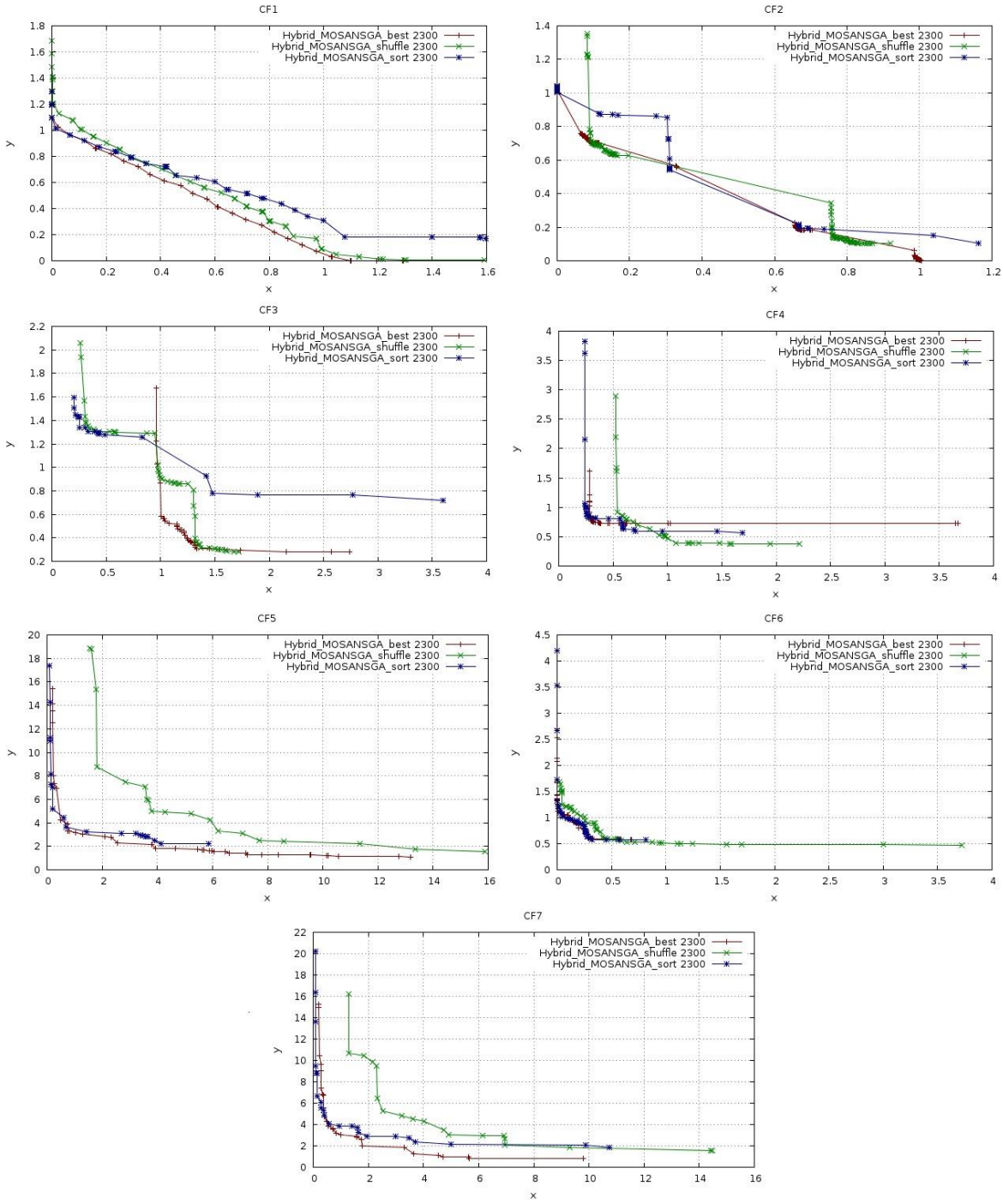


Figure 5-6 - Parallel MOSA and NSGA-II: CF1 to CF7.

From previous results can be concluded that choosing the best elements of each kernel it is the best approach. Also it can be observed the best between shuffle and sort by objective is dependent of the problem, where in CF1, CF2 and CF3 the shuffle method is better and in CF5, CF6 and CF7 the sort by objective is better.

To evaluate the performance of the multi kernel algorithms four combinations of two kernels were tested. The configurations used in the test are described in Table 5-2. On the parallel execution of kernels, two ideas are experimented, one is to segment the space focusing the search done by the different kernels

in different areas of the search space, and other is to use different methods to explore the same space. The first idea leads to the pMOSA[2] and pNASGII[2] combinations, while the second to the pMOSA-NSGAII. Having a GA followed by SA is common in the literature due to the global nature of the GA that favors the exploration of the search space, and the local nature of the SA, that favor the exploitation of the global solutions, hence the sequential sNSGAII-MOSA combination. In Figure 5-7 the results obtained for the four multi-kernel combinations are presented.

Table 5-2 - Hybrid configuration parameters

	Kernel1	Kernel2	Merge	Strategy
pMOSA[2]	MOSA(64, 2300)	MOSA(64, 2300)	step%300 == 0	Parallel-Split Sorted
pNSGA[2]	NSGAII(64,2300)	NSGAII(64,2300)	step%300 == 0	Parallel-Split Sorted
pMOSA-NSGAII	MOSA(64, 2300)	NSGAII(64,2300)	step%300 == 0	Parallel Select Best
sNSGAII-MOSA	NSGAII(128,2000)	MOSA(128, 300)	step%2000== 0	Seq.–All elements

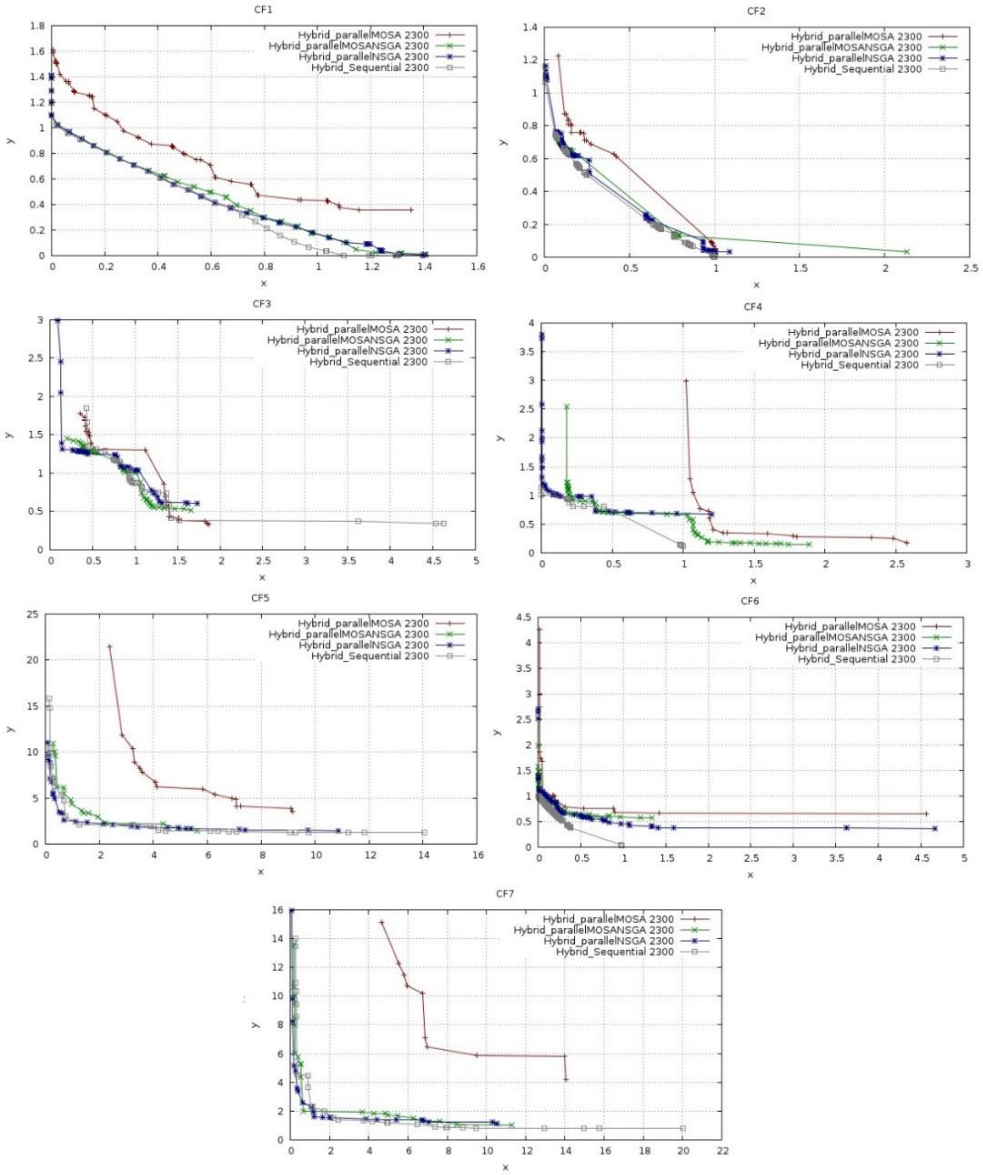


Figure 5-7 - Hybrid Pareto fronts for problems with 30 variables

One interesting and unexpected fact when mixing the kernels was that mixing two MOSA, who deliver the best Pareto fronts for many problems when using a single kernel, deliver the worst Pareto for all the problems in the multi kernel case. Another point is that NSGAII produces good results when mixed with either another NSGAII or with MOSA. From the four tested combinations the best results are obtained with the sequential combination of NSGAII followed by MOSA. This result could be explained by the efficient exploration of the search space performed by the NSGAII and then applying the MOSA local search to further optimize the already "good" global solution.

These tests were made to confirm the performance of the algorithms that were implemented as made to prove the potential of the hybrid multi-objective multi-kernel methods, further study is required to each kernel individually, and also, in the combinations of them. The test was also crucial to identify that the results of the combination of kernels cannot easily be foreseen, as illustrated by the case with the pMOSA[2].

5.4 Conclusions

In this chapter mathematical problems optimizations are presented, some examples of hybrid combinations are presented and is shown that mixing technics is a non-trivial exercise. As shown in the case of MOSA albeit being the best in solo, the combination of two MOSA in parallel gives the worst result for the experimented multi-kernel combinations of algorithms. It was also shown that the use of one algorithm to initial explore the search space and then use another one to further optimize a good solution is a possible and viable approach. This tests should be considered a starting point for future research on the combinations of optimization strategies applied to analog circuit.

6 Results for analog IC design

This chapter presents and discusses the results obtained with the implemented kernels to 4 different analog circuits. First two oscillators, an LC-Voltage Controlled Oscillator (VCO) and an LC-Oscillator, are used to evaluate the performance of the single kernel methods. The tests with the oscillators were conducted before the implementation of the multi-kernel methods, this is the reason why multi-kernel methods were not considered within these cases studies.

To evaluate the multi kernel methods, two amplifiers, a single stage differential amplifier that uses voltage combiners, and a two stage miller amplifier were considered. The algorithms applied were the single kernel MOSA, NSGA, and the multi-kernel combination sMOSA-NSGAII.

6.1 LC-Voltage controlled amplifier

Before moving to the LC-VCO circuit optimization, a brief preliminary note on oscillator design is here given. In general the design of oscillators aims at minimizing two objectives: the phase noise and power consumption for a given oscillation frequency. Traditionally these conflicting objectives are reflected in the Figure-of-Merit (FOM) defined by Kinget [63] and shown in (1), where ω_0 is the oscillation frequency, P_{dc} is the power consumption, $\Delta\omega$ is the offset from the output frequency and $L(\Delta\omega)$ is the oscillator phase noise, given by (2), the original Leeson's equation [64], where Q is the loaded quality factor of the oscillator, k is the Boltzmann's constant, T is the absolute temperature, P_{sig} is the oscillation output power, F is the noise factor of the amplifier and $\Delta\omega_{1/f3}$ is the corner frequency between $\omega_{1/f2}$ and $\omega_{1/f3}$ portion of the phase noise spectrum [65]. Finally, the more negative value of FOM (or higher absolute value), the better the performance of an oscillator.

$$\mathrm{FOM} = L(\Delta\omega) + 10\log\left[\frac{P_{\mathrm{dc}}}{1\mathrm{mW}} \times \left(\frac{\Delta\omega}{\omega_0}\right)^2\right] \tag{1}$$

$$L(\Delta\omega) = \frac{P(\Delta\omega)}{P(\omega_0)} = 10\log\left\{\frac{2FkT}{P_{\mathrm{sig}}} \times \left[1 + \left(\frac{\omega_0}{2Q\Delta\omega}\right)^2\right] \times \left(1 + \frac{\Delta\omega_{1/f3}}{|\Delta\omega|}\right)\right\} \tag{2}$$

This FOM is commonly used by designers to access the quality of the achieved solution, and is used in this work to compare the obtained solutions to other state of the art oscillator designs.

The measures of the circuit performance are done using a test bench and circuit simulator. A test bench is a circuit that is used only in simulation, i.e. is not intended to be fabricated. It is used to simulate, load, and provide the means to measure the performance figures of the circuit under test. In the case of the LC-Oscillator, a typical 10 MΩ resistive load was considered, while in the case of the LC-Voltage Controlled Oscillator a 1 pF load capacitance was used.

The performance figures of both the oscillators are measured using Synopsis' HSPICE/RF® circuit simulator using an operating point analysis and a steady state analysis with harmonic balance of oscillation. In the case of the harmonic balance analysis, the tone was defined as 2.4 GHz, calculating a total of 20 harmonics. The extracted measures are listed and described in Table 6-1. Note that in the

case of the LC-Oscillator, the measures related to the PMOS devices are not considered, as there are no PMOS devices in the circuit. In both cases, the power consumption is calculated not considering the power drained by the external current source.

Table 6-1 - Oscillators' performance figures measured.

ID	Units	Description
PN	dBc/Hz	Phase Noise Measured @ 1 MHz
P	mW	Power Consumption
OF	GHz	Effective Oscillation Frequency
OVS	mV	Differential Output Probe Voltage
OVP^n	mV	Overdrive Voltages of the PMOS nth Device (VTH-VGS)
OVN^n	mV	Overdrive Voltages of the NMOS nth Device (VGS-VTH)
DP^n	mV	Saturation Margin of the PMOS nth Device (VDSat-VDS)
DN^n	mV	Saturation Margin of the NMOS nth Device (VDS-VDSat)

The electrical schematic of the LC-VCO circuit is in Figure 6-1. The circuit optimization design flow starts by the definition of the design variables, the design objectives and the design constraints.

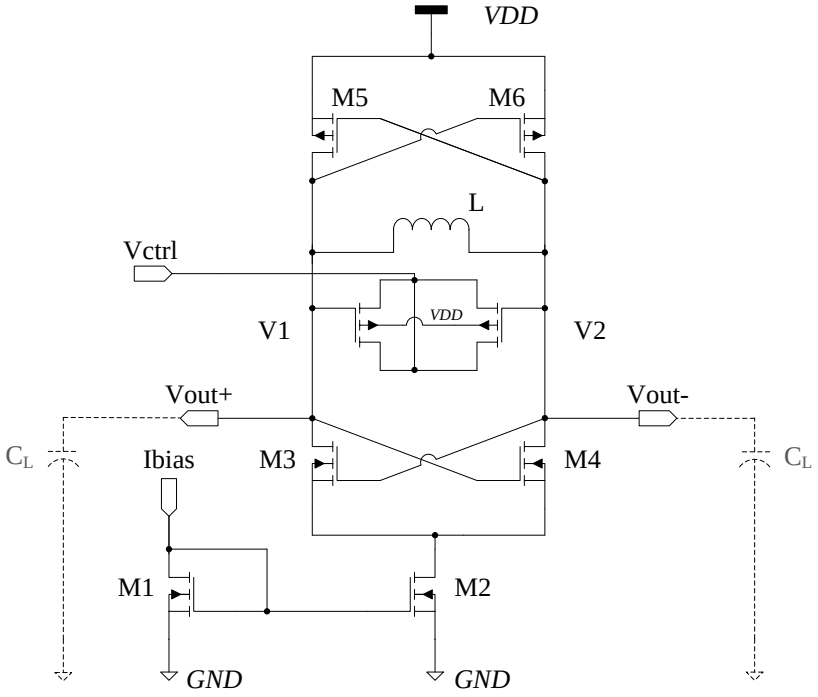


Figure 6-1 - LC-Voltage controlled oscillator circuit schematic.

The devices' sizes constitute the variables in the optimization process, and their ranges define the search space that is explored during the sizing procedure. The variable names and ranges, considering both maximum and minimum values, as well as the precision of the search grid are presented in Table 6-2. The optimization variables are the following device model's parameters: finger widths, lengths, and number of fingers of the transistors (and varactors (V1 and V2)), outer diameter of the inductors, and the external biasing current *Ibias*. The index number in each variable is according to the device names in Figure 6-1. Note that a pair of twin varactors is used, *M1* is equal to *M2*, *M3* is equal to *M4*, and *M5* is equal to *M6*, and all the variable ranges respect the technology available limits, in order to provide physically implementable solutions.

Table 6-2 - LC- VCO optimization variables and ranges.

Variable (Unit)	Min.	Grid Unit	Max.
w_{var}(µm)	1.0	0.1	10.0
w_1(µm)	1.0	0.1	10.0
w_3(µm)	1.0	0.1	10.0
w_5(µm)	1.0	0.1	10.0
nf_{var}	4	2	16
nf_2	4	2	16
nf_3	4	2	16
nf_5	4	2	32
l_{var}(nm)	120	10	920
l_1(nm)	120	10	920
l_3(nm)	120	10	920
l_5(nm)	120	10	920
outd(µm)	90.00	0.01	290.00
ib(mA)	0.1	0.1	5.0

As stated before, the design objectives are the minimization of both phase noise at 1 MHz and power consumption and the design constraints that are considered in all optimization processes are presented in Table 6-3.

Table 6-3 - LC- VCO design constraints.

Circuit Performance	Constraint
Power Consumption	≤ 6 mW
Oscillation Frequency	≥ 2.4 GHz; ≤ 2.4835 GHz
Output Signal Amplitude	≥ 100 mV
Phase Noise@ 1 MHz	≤ -100.0 dBc/Hz
Vth-VGS of M5 & M6	≥ 80 mV
VGS-Vth of M1, M2, M3 & M4	≥ 80 mV
VDSat-VDS of M5 & M6	≥ 50 mV
VDS-VDSat of M1, M2, M3 & M4	≥ 50 mV

The criteria used by the designer to set the constraints were the following: the LC-VCO amplification stage (namely *M3*, *M4*, *M5* and *M6*) are set to operate in the moderate inversion region, therefore, their overdrive voltage is required to be greater or equal to 80 mV, while all devices are to be working in saturation hence their saturation margin is required to be greater or equal to 50 mV. The acceptable range for the oscillation frequency was chosen considering practical applications (being an ISM operating frequency band, according to the Federal Communications Commission rules). Finally, both power consumption and phase noise limitations were set taking into consideration the available and most recent publications on LC-VCOs.

Applying the procedure described in Chapter 3 to describe the circuit design as a multi-objective optimization problem, leads to the following problem formulation where the two objectives to be minimized are:

$$f_1(x) = PN$$
$$f_2(x) = P$$

The constraints are:

$$g_1(x) = \frac{-100dBc/Hz - PN}{|-100dBc/Hz|}$$

$$g_2(x) = \frac{6mW - P}{|6mW|}$$

$$g_3(x) = \frac{OVS - 100mV}{|100mV|}$$

$$g_4(x) = \frac{OF - 2.4GHz}{|2.4GHz|}, g_5(x) = \frac{2.4835GHz - OF}{|2.4835GHz|}$$

$$g_{5+d}(x) = \frac{OVN^d - 80mV}{|80mV|}, g_{11+d}(x) = \frac{DN^d - 50mV}{|50mV|} d = 1:4$$

$$g_{9+d}(x) = \frac{OVP^d - 80mV}{|80mV|}, g_{15+d}(x) = \frac{DP^d - 50mV}{|50mV|}$$

And the search space is:

$$[1.0,\ 1.1,...10]^4 \times [4,6,..,16]^3 \times [4,6,...,32] \times [120,130,...,920]^4 \times [90,\ 90.1,...,290] \times [0.1, 0.2,...,5]$$

This is the optimization problem in the study of the multi-objective optimization of the LC-VCO. To study the three single kernel multi-objective strategies available in AIDA-CMK, a fixed number of evaluations (circuit simulations) was used, to provide a fair ground for the comparison of the different optimization strategies.

The total amount of simulations in each test run was 64 000. Two combinations of the number of elements the number of iterations were considered, in Runset I these values were {64, 1000} respectively, and in Runset II the values were {128, 500}; Both Runsets include ten independent executions, using a different seed in the random number generator for each run, taken from a common set of seeds, i.e., the same seed was used for the same run of the different algorithms.

To understand the evolution of the best solutions with the number of evaluations, intermediary results were stored. One additional run was executed using only NSGA-II considering a population size of 256 and 5000 generations. The purpose of such run was to find a better approximation of the true Pareto Optimal Front (POF) for this circuit, giving a reference to evaluate the overall quality of the solutions obtained previously.

The evolution of the best power, phase noise and FOM found with the number of simulations for each optimization kernel in both runsets is illustrated in Figure 6-2. Figure 6-3 shows the Pareto fronts for Runset I at different stages of the optimization process, namely at 6400, 12800, 25600, 38400, 51200 and 64000 simulations, while Figure 6-4 shows the same for Runset II.

The analysis of the results show that NSGA-II is much more efficient than MOPSO or MOSA, requiring fewer simulations to achieve the same solutions, and it is more consistent throughout the 10 runs. Interestingly, in Runset I the best FOM of the solutions of the NSGA-II actually gets slightly worse in one of the runs, Figure 6-2 (c). This is a side effect of the indirect optimization of the FOM and happens due to some fluctuations of the oscillating frequency within the feasible range, i.e., even though the

optimization lead to solutions with better phase noise and/or power, at that point small differences in the oscillating frequency actually cause the FOM to get slightly worse. It is also relevant to notice that in one of the runs of the MOPSO, the best value of the phase noise was really good, being found in very few evaluations, Figure 6-2 (b), this is due to the stochastic nature of the techniques used, and that is why 10 runs were executed for each study.

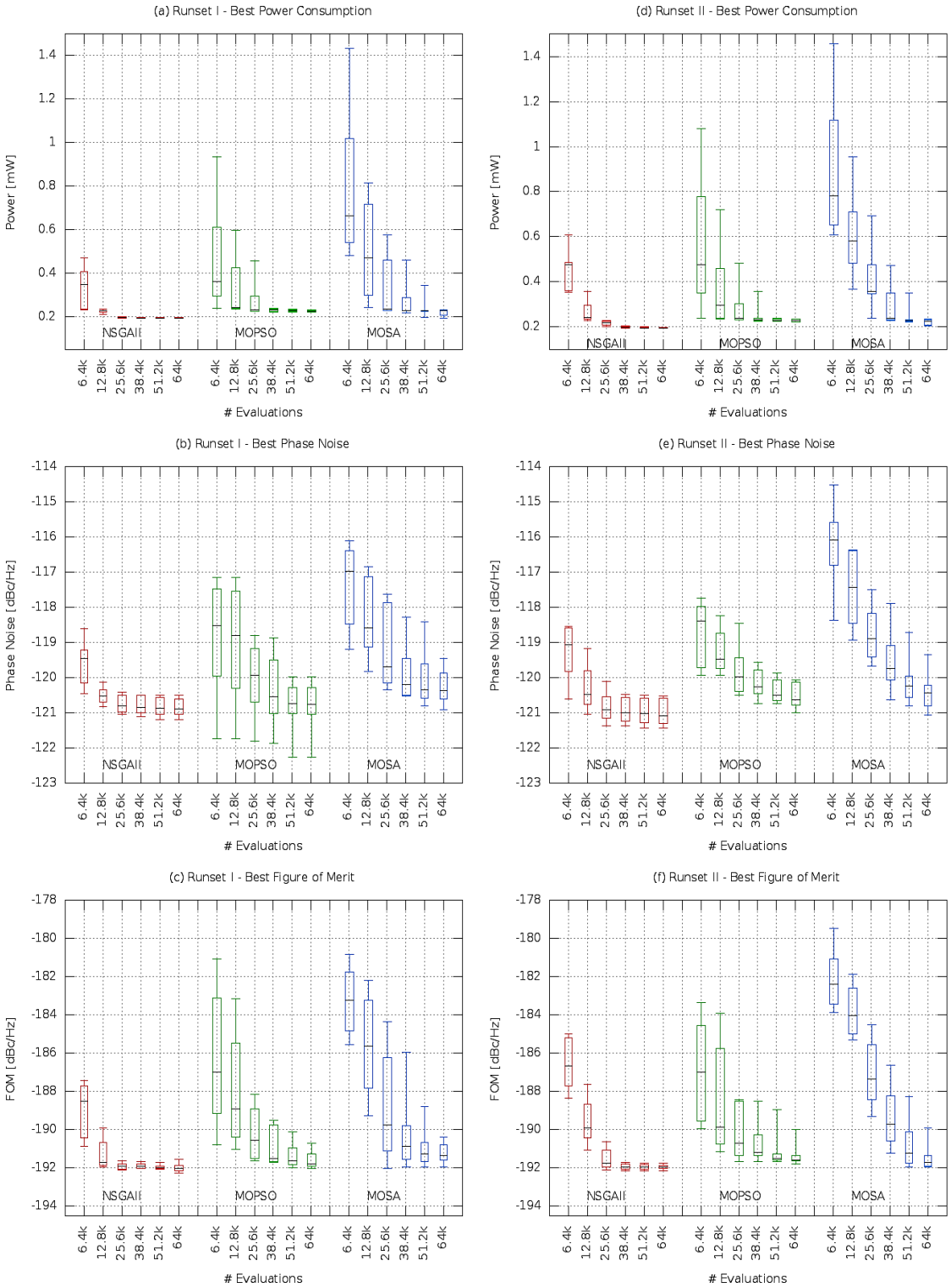


Figure 6-2 - Evolution of the best Power, Phase-Noise and FOM with the number of simulations. (a) Best Power consumption for Runset I; (b) Best Phase-Noise for Runset I; (c) Best FoM for Runset I; (d) Best Power consumption for Runset II; (e) Best Phase-Noise for Runset II; (f) Best FoM for Runset II.

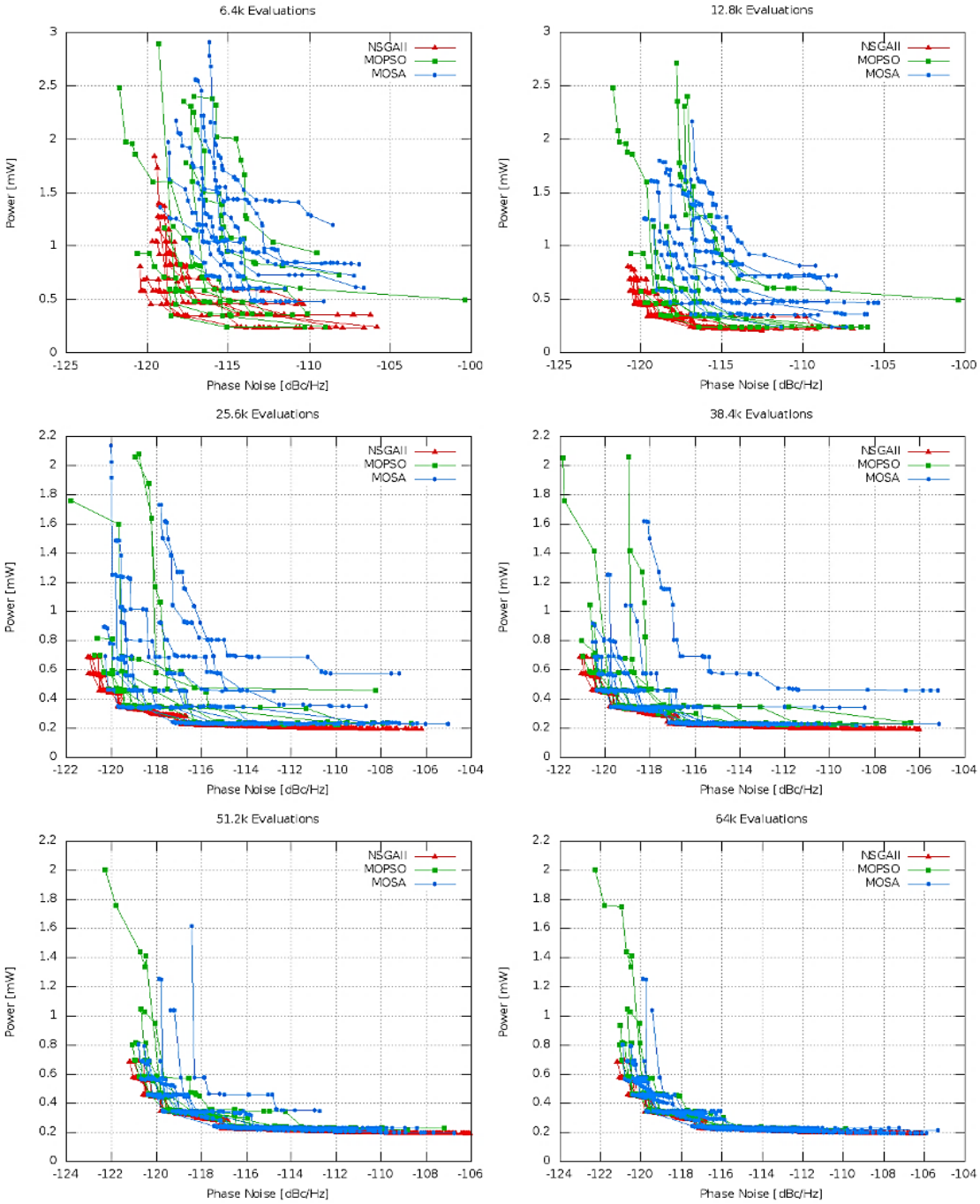


Figure 6-3 - Evolution of the Pareto front with the number of simulations for the different runs of the NSGA-II, MOPSO and MOSA for Runset I.

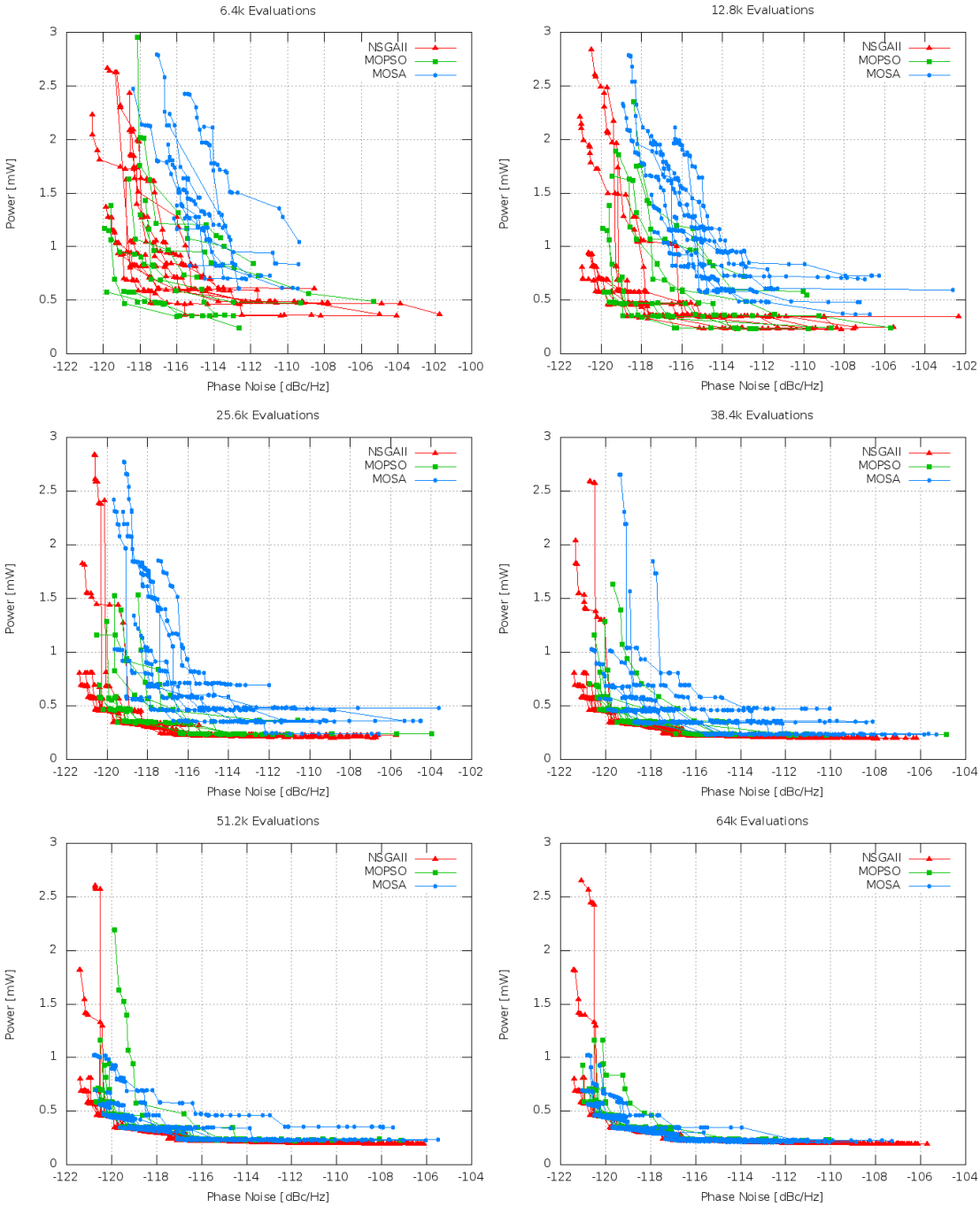


Figure 6-4 - Evolution of the Pareto front with the number of simulations for the different runs of the NSGA-II, MOPSO and MOSA for Runset II.

Nevertheless, even using the other algorithms, after a couple of hours, multiple LC-VCO are designed showing state-of-the-art FOMs even though the FOM is not being explicitly optimized. Figure 6-5, shows a zoom-in in the area of the Pareto fronts where the solutions with best FOM can be found, as stated before the NSGA-II reveal better results than both other two approaches, however all three approaches show results comparable to the POF obtained using 20 times more evaluations.

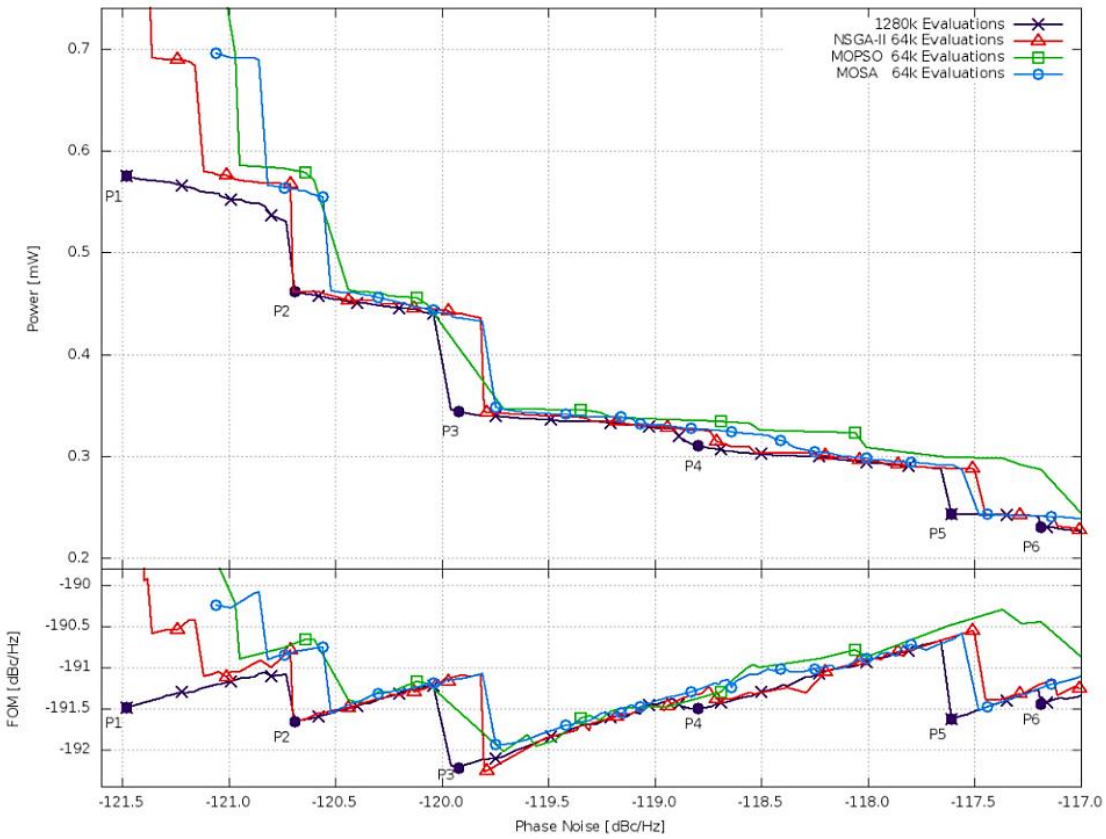


Figure 6-5 - Detailed view of the POF, showing the FOM for each solution.

From the stand point of the RF designer, it is also important to note the local minima found in the FOM, {*P1*, …, *P6}*, these results show the tradeoffs between phase noise and power, all having extremely competitive FOM (≤ -191.5 dBc/Hz) but showing a variation of 147% in power consumption from 0.23 to 0.57 mW, while the phase noise is varies 4.3 dBc/Hz from -121.5 to -117.2 dBc/Hz.

Table 6-4 summarizes the "best" results obtained in this case study, while comparing them to other published works. Here it can be seen that they achieve a FOM that is better when compared with other state-of-the-art LC-VCOs, while explicitly and efficiently exploring the design tradeoff between phase noise and power consumption.

Table 6-4 - Summary of the most competitive LC- VCO solutions, showing other SOA results.

		TECH.	FREQ. [GHz]	POWER [mW]	Δf [MHz]	L(Δf) [dBc/Hz]	FoM [dBc/Hz]
[66] (1)		90 nm	2.42	0.515	1.0	-119.7	-190.3
[67] (2)		180 nm	2.4	0.60	1.0	-103.7	-173.5
[68] (2)		180 nm	2.02	3.15	0.6	-118.9	-186.0
[69] (2)		180 nm	2.63	0.432	0.4	-106.4	-186.4
[70] (2)		90 nm	2.16	0.528	0.4	-106.2	-183.6
[71] (2)		90 nm	2.40	0.564	1.0	-116.8	-186.9
[71] (2)		130 nm	2.40	0.624	1.0	-117.1	-186.8
This work(1)	P1	130 nm	2.40	0.575	1.0	-121.5	-191.5
	P2			0.462		-120.7	-191.7
	P3			0.344		-119.9	-192.2
	P4			0.310		-118.8	-191.5
	P5			0.244		-117.6	-191.6
	P6			0.231		-117.2	-191.5

[1] Simulation;[2] Measurement

6.2 LC-Oscillator

The LC-Oscillator, shown in Figure 6-6 , is a purely reactive feedback circuit widely used in RF front-end circuits.

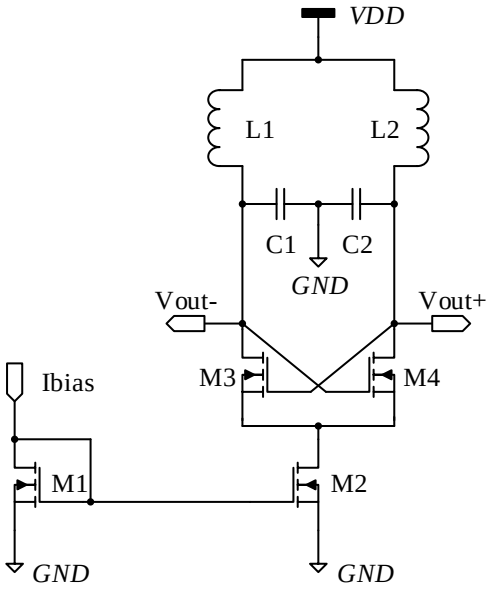


Figure 6-6 - LC- Oscillator: a) circuit schematic; b) small signal equivalent circuit.

The optimization setup of the LC-Oscillator was similar to the setup of the LC-VCO, the devices' sizes that constitute the variables in the optimization processes and their ranges are presented in Table 5. Like in the case of the LC-VCO, the optimization variables are the finger widths, lengths, number of fingers and outer diameter of the inductors. In the case of the LC-Oscillator, a pair of twin inductors is used, a pair of twin capacitors is used, *M1* is equal to *M2*, and *M3* is equal to *M4*, again all the ranges respect the technology limits, in order to provide physically implementable solutions. The optimization target were to minimize both phase noise at 1 MHz and power consumption and the constraints are shown in **Table** 6-5.

Table 6-5 - LC-Oscillator optimization variables and ranges.

Variable (Unit)	Min.	Grid Unit	Max.
w_c (µm)	10.0	0.1	100.0
l_c (µm)	10.0	0.1	100.0
w_1 (µm)	1.0	0.1	20.0
w_3 (µm)	1.0	0.1	20.0
nf_2	4	2	16
nf_3	4	2	16
l_1 (nm)	120	10	920
l_3 (nm)	120	10	920
outd (µm)	90.00	0.01	290.00

The constraints are shown in Table 6-6, in the case of the LC-Oscillator, the active devices are set to operate in the strong inversion region in order to maximize the output voltage amplitude without distortion.

Table 6-6 - LC-Oscillator optimization constraints.

Circuit Performance	Constraint
Phase Noise @ 1 MHz	≤ -113.0 dBc/Hz
Power Consumption	≤ 4 mW
Output Signal Amplitude	≥ 100 mV
Oscillation Frequency	≥ 2.4 GHz; ≤ 2.4835 GHz
VGS-Vth of M1, M2, M3 &M4	≥ 100 mV
VDS-VDSat of M1, M2, M3 & M4	≥ 50 mV

The previous design specifications lead to the optimization problem where the two objectives to be minimized are again:

$$f_1(x) = PN$$
$$f_2(x) = P$$

The constraints are now:

$$g_1(x) = \frac{-113dBc/Hz - PN}{|-113dBc/Hz|}$$
$$g_2(x) = \frac{4mW - P}{|4mW|}$$
$$g_3(x) = \frac{OVS - 100mV}{|100mV|}$$
$$g_4(x) = \frac{OF - 2.4GHz}{|2.4GHz|}, g_5(x) = \frac{2.4835GHz - OF}{|2.4835GHz|}$$
$$g_{5+d}(x) = \frac{OVN^d - 100mV}{|80mV|}, g_{11+d}(x) = \frac{DN^d - 50mV}{|50mV|} d = 1:4$$

And the search space is: $[10.0,\ 10.1,...100]^2 \times [1.0,\ 1.1,...10]^2 \times [4,6,..,16]^2 \times [120,130,...,920]^2 \times [90,\ 90.1,...,290]$

This is the optimization problem considered in the study of the multi-objective optimization of the LC-Oscillator. The total amount of simulations in each test run was 32.000, the number of elements was 64, and the number of iterations was 500. Again, ten executions using a different seed for each run for each algorithm were done.

Figure 6-7 (a) shows a zoom-in in the area of the Pareto fronts where the solutions with best FOM can be found at the end of the optimization, while Figure 6-7 (a), (b) and (c) show the best power, phase noise and FOM found for each optimization kernel respectively. The analysis of the results show again that NSGA-II is more efficient than MOPSO or MOSA. Again, and taking notice that the ranges of plots are small, the multiple runs of all the algorithms yield consistent results, producing design solutions showing state-of-the-art FOMs even though the FOM is not being optimized explicitly. Like in the case of the LC-VCO, the FOM of the LC-Oscillators along the Pareto front shows several local minima, showing the tradeoffs between phase noise and power, all having competitive FOM (≤ -185 dBc/Hz) but showing a variation of 367% in power consumption from 0.588 to 2.746 mW, while the phase noise is varies 7.6 dBc/Hz from -123.3 to -115.7 dBc/Hz

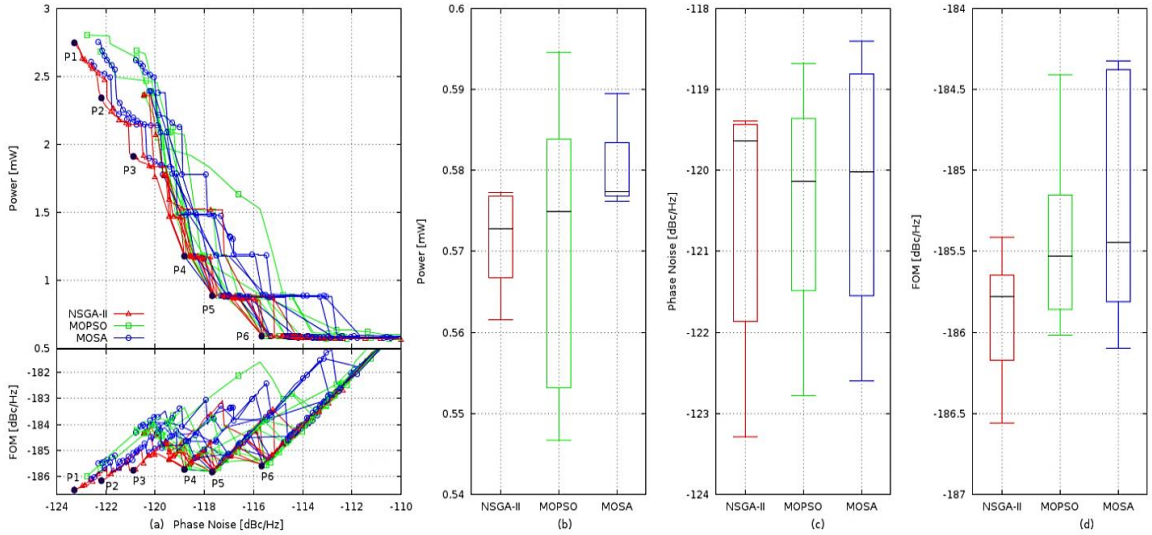


Figure 6-7 - LC-Oscillator optimization results
(a) POF and FOM, (b) the best Power, phase noise, and FOM for the multiple runs.

6.3 Single stage amplifier with gain enhancement using voltage combiner

The amplifier topology using voltage combiners proposed in [72] was optimized to maximize the figure of merit (FOM) and maximize de low-frequency gain (GDC). By maximizing the FOM, the power consumption is minimized as well as the gain-bandwidth product (GBW) is maximized, as described by the FOM is in (1), where C_{load} is the load capacity and *IDD* is the current consumption.

$$FOM = \frac{GBW \times C_{load}}{IDD} \left[\frac{MHz \times pF}{mA}\right] \quad (1)$$

The circuit schematic is shown in Figure 6-8 and the performance figures are measured using Mentor Graphics Eldo® circuit. The extracted measures that can be used to define objectives as constraints are listed and described in Table 6-7.

Table 6-7 - Single stage amplifier performance figures.

ID	Units	Description
IDD	A	Current Consumption
GDC	dB	Low-Frequency Gain
GBW	Hz	Unity Gain Frequency
PM	degree	Phase Margin
FOM	MHz * pF / mA	Figure of Merit
OVP^n	mV	Overdrive Voltages of the PMOS nth Device (VTH-VGS)
OVN^n	mV	Overdrive Voltages of the NMOS nth Device (VGS-VTH)
DP^n	mV	Saturation Margin of the PMOS nth Device (VDSat-VDS)
DN^n	mV	Saturation Margin of the NMOS nth Device (VDS-VDSat)

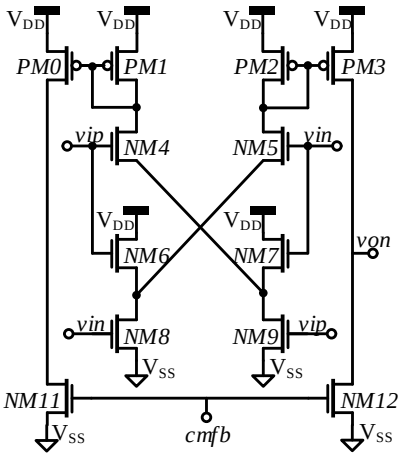


Figure 6-8 - Gain enhanced amplifier schematic.

The optimization setup of the Gain enhanced amplifier was similar to the setup of the LC-VCO, the devices' sizes that constitute the optimization variables were the width, length and the number of fingers of all the MOS devices, and the dimension ranges are presented in Table 6-8.

Table 6-8 - Gain enhanced amplifier optimization variables and ranges.

Variable (Unit)	Min.	Grid	Unit Max.
l0, l1, l4, l6, l8, l10 (nm)	120	10	1000
w0 w1 w4 w6 w8 w10 (µm)	1	0.1	10
nf0, nf1, nf4, nf6, nf8, nf10	1	2	8

The optimization target were to maximize both the figure of merit (FOM) and low-frequency gain (GDC).and the constraints are shown in Table 6-9.

Table 6-9 - Gain enhanced amplifier optimization constraints.

Circuit Performance	Constraint
FOM	≥ 1000 MHz.pF/mA
IDD	≤ 350µA
GDC	≥ 50dB
GBW@6pF	≥ 30MHz
PM	≥ 60º
Vth-VGS of PM0, PM1, PM2, & PM3	≥ 100 mV
VGS-Vth of NM4, NM5, NM6, NM7, NM8, NM9, NM11 & NM12	≥ 100 mV
VDSat-VDS of PM0, PM1, PM2, & PM3	≥ 50 mV
VDS-VDSat of NM4, NM5, NM6, NM7, NM8, NM9, NM11 & NM12	≥ 50 mV

The previous design specifications lead to the optimization problem where the two objectives to be minimized are again:

$$f_1(x) = -FoM$$
$$f_2(x) = -Gdc$$

The constraints are now:

$$g_1(x) = \frac{FOM - 1000\,\text{MHz.pF/mA}}{|-1000\,\text{MHz.pF/mA}|}$$

$$g_2(x) = \frac{350\mu\text{A} - IDD}{|350\mu\text{A}|}$$

$$g_3(x) = \frac{GDC - 50dB}{|50dB|}$$

$$g_4(x) = \frac{GBW - 30MHz}{|30MHz|}$$

$$g_5(x) = \frac{PM - 60^o}{|60^o|}$$

$$g_{6+d}(x) = \frac{OVP^d - 100mV}{|100mV|}, g_{10+d}(x) = \frac{DP^d - 50mV}{|50mV|}\, d = 0:3$$

$$g_{10+d}(x) = \frac{OVN^d - 100mV}{|100mV|}, g_{23+d}(x) = \frac{DN^d - 50mV}{|50mV|}\, d = 4:12$$

And the search space is: $[120,130,...,1000]^6 \times [1.0,\ 1.1,...10]^6 \times [1,2,..,8]^6$

This is the optimization problem considered in the study of the multi-objective optimization of the Gain enhanced amplifier. The total amount of simulations in each test run was 64.000, the number of elements was 128, and the number of iterations was 500. The hybrid configuration used was a sequential NSGA-II-MOSA (sNSGA-MOSA) with 400 NSGA iterations followed by 100 MOSA iterations. This choice was made based on the test results using mathematical functions. A few executions using a different seed for each run for each kernel were done. Figure 6-9 shows the Pareto fronts of the simulations, where the NSGA-II and the Hybrid clearly outperform the MOSA. Both the NSGA-II and the sNSGA-MOSA presented very similar results.

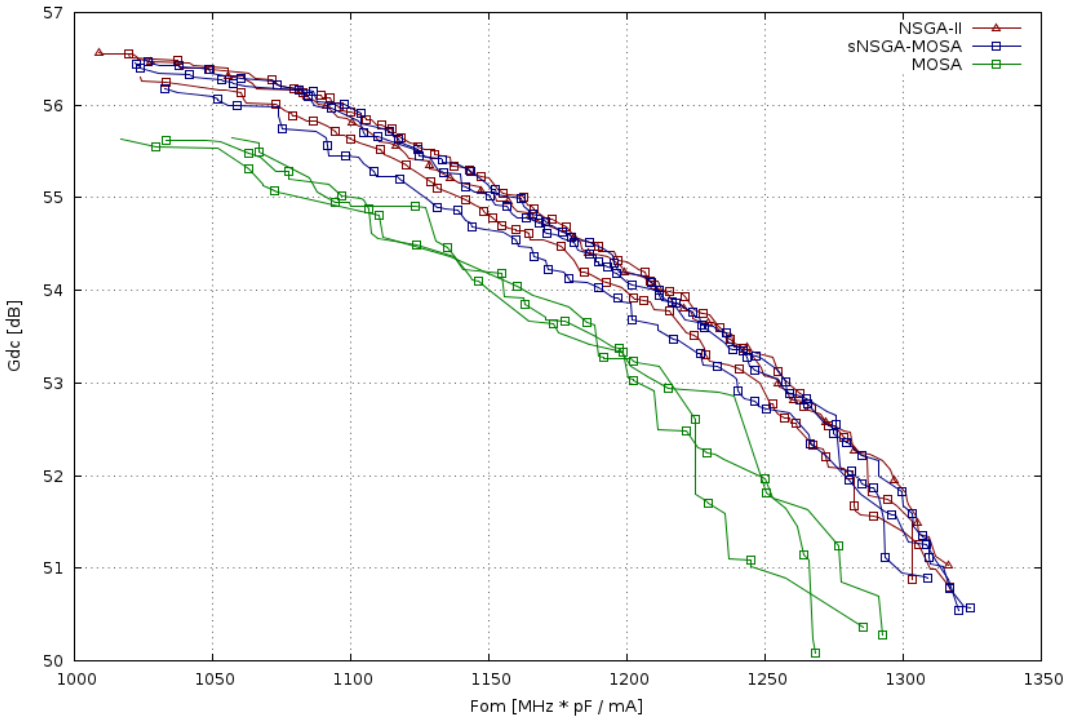


Figure 6-9 - Gain enhanced amplifier optimization results

6.4 Two stage amplifier

The two-stage operational amplifier of Figure 6-10, loaded with a 10MΩ resistor in parallel with 1 pF capacitor and biased with a current of 10µA, is optimized for two objectives: maximize the bandwidth GBW while minimizing the current consumption.

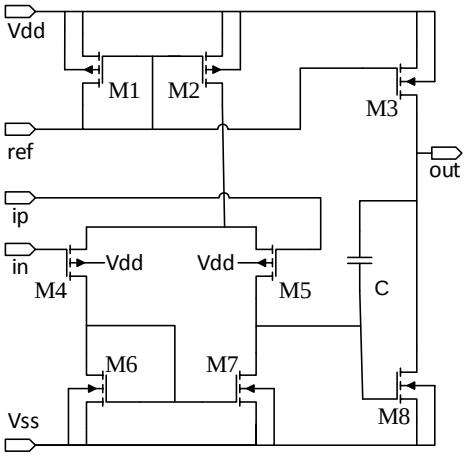


Figure 6-10 - Two-stage operational amplifier schematic

The measures are extracted from the circuit simulation and can be used to define objectives and constraints are listed and described in Table 6-10.

Table 6-10 - Two-stage amplifier performance figures.

ID	Units	Description
IDD	A	Current Consumption
GDC	dB	Low-Frequency Gain
GBW	Hz	Unity Gain Frequency
PM	degree	Phase Margin
NO	V / V	Noise RMS
SN	V / √Hz	Noise Density
Sr		Slew Rate
Voff	V	Offset Voltage
PSRR	V	Power Supply Rejection Ratio
OVP^n	mV	Overdrive Voltages of the PMOS nth Device (VTH-VGS)
OVN^n	mV	Overdrive Voltages of the NMOS nth Device (VGS-VTH)
DP^n	mV	Saturation Margin of the PMOS nth Device (VDSat-VDS)
DN^n	mV	Saturation Margin of the NMOS nth Device (VDS-VDSat)

The optimization variables are the width, length, number of fingers and number of rows of the MOS devices, and the length and number of fingers of the MOM capacitor. The variable ranges considered are indicated in Table 6-11.

Table 6-11 - 2 Stage amplifier optimization variables and ranges.

Variable (Unit)	Min.	Grid Unit	Max.
l1, l4, l6, l8 (nm)	120	5	1000
w1 w4 w6 w8 (µm)	1	0.1	10
nf1, nf2, nf3, nf4, nf6, nf8	1	2	200
lc(µm)	4.4	0.1	100
nfc	14	2	198

Table 6-12 indicated the design specifications for this circuit working as a versatile low power DC buffer.

Table 6-12 - 2 Stage amplifier constraints.

Circuit Performance	Constraint
IDD	≤ 80 µA
GDC	≥ 50dB
GBW	≥ 1 MHz
PM	≥ 55º
PSRR	≥ 55dB
SR	≥ 0.8 V/µs
Voff	≤ 1 mV
No	≤ 400 µVrms
Sn	≤ 100 nV/√Hz
Vth-VGS of M1, M2, M3, M4 & M5	≥ 100 mV
VGS-Vth of M6, M7 & M8	≥ 100 mV
VDSat-VDS of M1, M2, M3, M4 & M5	≥ 50 mV
VDS-VDSat of M6, M7 & M8	≥ 50 mV

The previous design specifications lead to the optimization problem where the two objectives to be minimized are again:

$$f_1(x) = IDD$$
$$f_2(x) = -GBW$$

The constraints are now:

$$g_1(x) = \frac{80\mu\text{A} - IDD}{|80\mu\text{A}|}$$
$$g_2(x) = \frac{GDC - 50dB}{|50dB|}$$
$$g_3(x) = \frac{GBW - 1MHz}{|1MHz|}$$
$$g_4(x) = \frac{PM - 55^o}{|55^o|}$$
$$g_5(x) = \frac{PSRR - 55dB}{|55dB|}$$
$$g_6(x) = \frac{SR - 0.8V/\mu s}{|0.8V/\mu s|}, g_7(x) = \frac{1\text{mV} - Voff}{|1\text{mV}|}$$
$$g_8(x) = \frac{400\text{uVrms} - No}{|400\text{uVrms}|}, g_9(x) = \frac{100\text{nV}/\sqrt{\text{Hz}} - Sn}{|100\text{nV}/\sqrt{\text{Hz}}|}$$
$$g_{9+d}(x) = \frac{OVP^d - 100mV}{|100mV|}, g_{17+d}(x) = \frac{DP^d - 50mV}{|50mV|}\ d = 1:5$$
$$g_{9+d}(x) = \frac{OVN^d - 100mV}{|100mV|}, g_{17+d}(x) = \frac{DN^d - 50mV}{|50mV|}\ d = 6:8$$

And the search space is: $[120,130,...,1000]^4 \times [1.0,\ 1.1,...10]^4 \times [1,2,4,..,200]^6 \times [4.4,.4.5,...,100] \times [14,16,..,198]$

This is the optimization problem considered in the study of the multi-objective optimization of the two stage amplifier. The total amount of simulations in each test run was 128.000, the number of elements was 128, and the number of iterations was 1000. The sNSGA-MOSA configuration used again, switching the kernels on the 700th iteration, resulting in 700 NSGA iterations followed by 300 MOSA iterations. Figure 6-11 shows the Pareto fronts obtained from a couple of independent runs.

Again the sNSGA-MOSA and NSGA-II outperform the MOSA. The NSGA-II and sNSGA-MOSA presented very similar results on the center of the POF, sNSGA-MOSA clearly outperforms the NSGA in the edges of the POF.

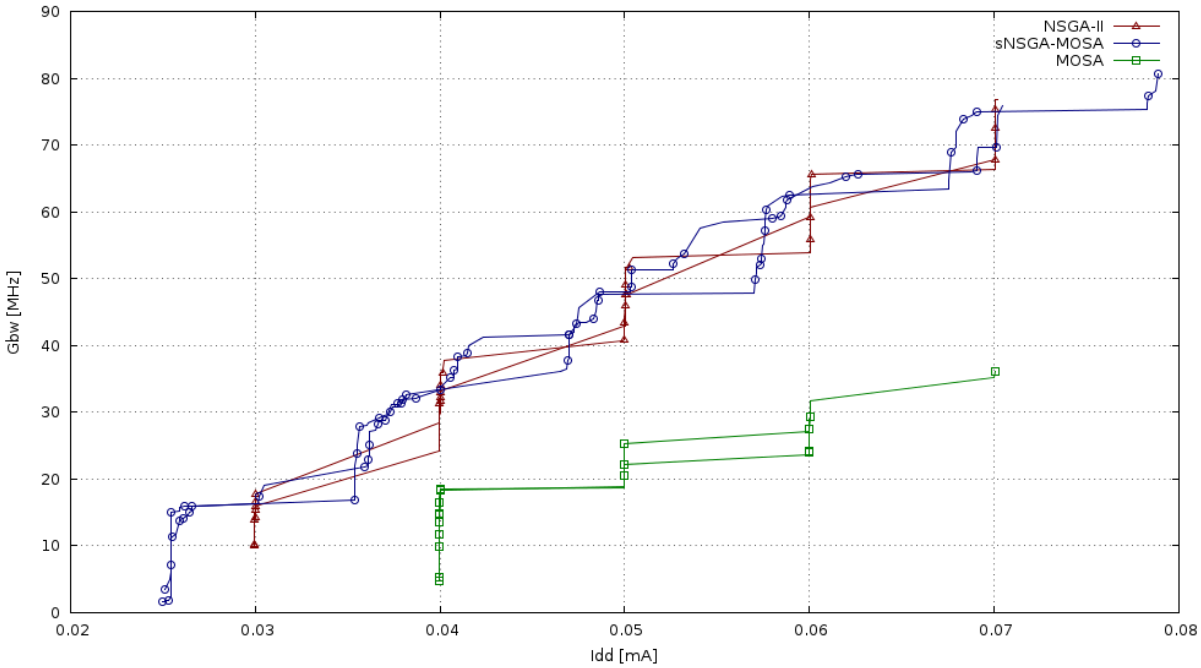


Figure 6-11 - Two-stage operational amplifier results

6.5 Conclusions

In this chapter the circuit's problem optimizations are presented and shown that the optimization of analog circuit is very different from the optimization of the Mathematical functions, considered in the Chapter 5. Although defined similarly, the optimization of these two types of problems is not trivial, and the best strategy to optimize one is not necessarily the best for the other. Given the example of the MOSA, it performed best in solo on mathematical functions, but proved to be the worst option for the circuit's problems. It was also shown that the use of one algorithm to initial explore the search space and then use another one to further optimize until a good solution is found it is a possible and viable approach, but further testing on the merge strategies of the algorithms could be conducted in order to tune the optimization and improve the quality of the results found.

7 Conclusion and future work

This chapter presents the conclusions of all the work executed for this dissertation, and the future directions for the continuous development of AIDA and the circuit optimization framework.

7.1 Conclusions

The work presented in this dissertation corresponds to an innovative IC design automation approach by implementing an abstraction layer between the circuit optimization and the optimization engine. Moreover, by creating a flexible optimization framework that implements an abstraction layer, two important goals were achieved: first it was possible in a very short period to develop comparative studies of three multi-objective optimization methods, namely the MOSA, NSGAII and MOPSO; secondly, two hybridization methods were proposed and tested in real analog circuits showing the potential of the implemented framework to help in the development and tailoring of innovative optimization methods for analog ICs.

The new framework proved to be capable to accelerate and reduce the development time of new optimization methods by providing a seamless integration with the optimization method and potentiating the reuse of common strategies for particular operations in the algorithm. By sharing strategies is easy to experiment combinations of the available strategies and to add new strategies.

Finally, the proposed objectives for this work were achieved and a new optimizer was created.

7.2 Further work

In analog design automation, the developments of new and better approaches are always necessary. There is still a long way to go in this domain; the improvement on productivity of analog design is an economic demand, from this work and its application to analog design there are some suggestions for future research which may improve the development of new and better automation tools. The first and obvious suggestion is to reap the benefits of the implemented framework by experimenting new methods and approaches to the analog circuits sizing and optimization.

Another suggestion is the creation of a public circuit optimization benchmark that is well defined and does not depend on confidential technology information that lay the building blocks for an open access common ground enabling to compare the implemented methods between research groups. While this clearly would be an incredibly valuable tool in the circuit sizing and optimization domain and although conceptually simple, the nature of analog design and the amount of trade secrets and confidential implementation details make it extremely difficult to legally distribute a realistic collection of analog circuits that can be optimized by anyone anywhere in the world.

Finally and although simple unity testing was implemented for many of the developed classes, integration and functional testing with automatic periodic regressions while extremely valuable for the maintenance of the framework, were out of the scope of this work. Creating such an infrastructure, while not necessarily scientifically relevant, would further ease the development of new and innovative methods for analog IC optimization.

References

[1] G. G. E. Gielen, "CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip," *Computers and Digital Techniques, IEE Proceedings ,* vol. 152, no. 3, pp. 317-332, 2005.

[2] G. Gielen and R. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," in *Proceedings of the IEEE (Volume:88 , Issue: 12)* , 2000.

[3] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE,* vol. 86, no. 1, pp. 82-85, 1998.

[4] M. F. M. Barros, J. M. C. Guilherme and N. C. G. Horta, Analog circuits and systems optimization based on evolutionary computation techniques, Berlin: Springer, 2010.

[5] SYNOPSIS INC., "Synopsys HSPICE - Accurate Circuit Simulation," [Online]. Available: http://www.synopsys.com. [Accessed 2014].

[6] Cadence, "Cadence Spectre Accelerated Parallel Simulator," [Online]. Available: http://www.cadence.com/products/cic/accelerated_parallel/pages/default.aspx. [Accessed 2014].

[7] R. Martins, N. Lourenço, S. Rodrigues, J. Guilherme and N. Horta, "AIDA: Automated Analog IC Design Flow from Circuit Level to Layout," in *International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design*, Seville, 2012.

[8] C. A. Makris and a. C. Toumazou, "Analog IC design automation. II. Automated circuit correction by qualitative reasoning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 14, no. 2, pp. 239-254, 1995.

[9] C. Toumazou and C. A. Makris, "Analog IC design automation. I. Automated circuit generation: new concepts and methods," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 14, no. 2, pp. 218-238, 1995.

[10] M. G. R. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B. L. A. G. Goffart, E. A. Vittoz, S. Cserveny, C. Meixenberger, G. V. D. Stappen and H. J. Oguey, "IDAC: an interactive design tool for analog CMOS circuits," *IEEE Journal of Solid-State Circuits,* vol. 22, no. 6, pp. 1106-1116, 1987.

[11] N. Horta, "Analogue and Mixed-Signal Systems Topologies Exploration Using Symbolic Methods," *Analog Integrated Circuits and Signal Processing,* vol. 31, no. 2, pp. 161-176, 2002.

[12] H. Y. Koh, C. H. Sequin and P. R. Gray, "OPASYN: a compiler for CMOS operational amplifiers," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 9, no. 2, pp. 113-125, 1990.

[13] J. P. Harvey, M. I. Elmasry and B. Leung, "STAIC: an interactive framework for synthesizing CMOS and BiCMOS analog circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 11, no. 11, pp. 1402-1417, 1992.

[14] P. C. Maulik, L. R. Carley and D. J. Allstot, "Sizing of cell-level analog circuits using constrained optimization techniques," *IEEE Journal of Solid-State Circuits,* vol. 28, no. 3, pp. 223-241, 1993.

[15] P. C. Maulik, L. R. Carley and R. A. Rutenbar, "Integer programming based topology selection of cell-level analog circuits," *IEEE Transactions on omputer-Aided Design of Integrated Circuits and Systems,* vol. 14, no. 4, pp. 401-412, 1995.

[16] K. Matsukawa, T. Morie, Y. Tokunaga, S. Sakiyama, Y. Mitani, M. Takayama, T. Miki, A. Matsumoto, K. Obata and S. Dosho, "Design methods for pipeline & delta-sigma A-to-D converters with convex optimization," in *Asia and South Pacific Design Automation Conference*, 2009.

[17] M. d. M. Hershenson, S. P. Boyd and T. H. Lee, "GPCAD: a tool for CMOS op-amp synthesis," in *IEEE/ACM International Conference on Computer-Aided Design*, San Jose, 1998.

[18] M. Kuo-Hsuan, P. Po-Cheng and C. Hung-Ming, "Integrated hierarchical synthesis of analog/RF circuits with accurate performance mapping," in *International Symposium on Quality Electronic Design*, Santa Clara, 2011.

[19] A. J. Torralba, J. Chavez and L. G. Franquelo, "Fuzzy-logic-based analog design tools," *Micro, IEEE,* vol. 16, no. 4, pp. 60-68, 1996.

[20] A. Torralba, J. Chavez and L. G. Franquelo, "FASY: a fuzzy-logic based tool for analog synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 15, no. 7, pp. 705-715, 1996.

[21] G. G. E. Gielen, H. C. C. Walscharts and W. M. C. Sansen, "Analog circuit design optimization based on symbolic simulation and simulated annealing," *IEEE Journal of Solid-State Circuits,* vol. 25, no. 3, pp. 707-713, 1990.

[22] E. S. Ochotta, R. A. Rutenbar and L. R. Carley, "Synthesis of high-performance analog circuits in ASTRX/OBLX," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 15, no. 3, pp. 273-294, 1996.

[23] W. Kruiskamp and D. Leenaerts, "DARWIN: CMOS opamp synthesis by means of a genetic algorithm," in *Design Automation Conference*, 1995.

[24] A. Doboli, N. Dhanwada, A. Nunez-Aldana and R. Vemuri, "A two-layer library-based approach to synthesis of analog systems from VHDL-AMS specifications," *ACM Transactions on Design Automation of Electronic Systems,* vol. 9, no. 2, pp. 238-271, 2004.

[25] L. W. Nagel, "SPICE2: A Computer Program to Simulate Semiconductor Circuits," EECS Department, University of California, Berkeley, 1975.

[26] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by Simulated Annealing," *Science,* vol. 220, no. 4598, pp. 671-680, 1983.

[27] W. Nye, D. Riley, A. Sangiovanni-Vincentelli and A. Tits, "DELIGHT.SPICE: an optimization-based system for the design of integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 7, no. 4, pp. 501-519, 1988.

[28] L. Cheng-Wu, S. Pin-Dai, S. Ya-Ting and C. Soon-Jyh, "A bias-driven approach for automated design of operational amplifiers," in *International Symposium on VLSI Design, Automation and Test*, Hsinchu, 2009.

[29] F. Medeiro, F. Fernandez, R. Dominguez-Castro and A. Rodriguez-Vazquez, "A Statistical Optimization-based Approach For Automated Sizing Of Analog Cells," in *International Conference Computer-Aided Design*, 1994.

[30] R. Castro-Lopez, O. Guerra, E. Roca and F. Fernandez, "An Integrated Layout-Synthesis Approach for Analog ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 27, no. 7, pp. 1179-1189, 2008.

[31] M. Barros, J. Guilherme and N. Horta, "Analog circuits optimization based on evolutionary computation techniques," *Integration, the VLSI Journal,* vol. 43, no. 1, pp. 136-155, 2010.

[32] M. Barros, J. Guilherme and N. Horta, "GA-SVM feasibility model and optimization kernel applied to analog IC design automation," in *ACM Great Lakes symposium on VLSI*, Stresa-Lago Maggiore, 2007.

[33] G. Alpaydin, S. Balkir and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," *IEEE Transactions on Evolutionary Computation,* vol. 7, no. 3, pp. 240-252, 2003.

[34] M. Krasnicki, R. Phelps, R. Rutenbar and L. Carley, "MAELSTROM: efficient simulation-based synthesis for custom analog cells," in *Design Automation Conference*, New Orleans, 1999.

[35] R. Santos-Tavares, N. Paulino, J. Higino, J. Goes and J. P. Oliveira, "Optimization of Multi-Stage Amplifiers in Deep-Submicron CMOS Using a Distributed/Parallel Genetic Algorithm," in *International Symposium on Circuits and Systems* , Seattle, 2008.

[36] R. Phelps, M. Krasnicki, R. Rutenbar, L. Carley and J. Hellums, "Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 19, no. 6, pp. 703-717, 2000.

[37] J. R. Koza, F. I. Bennett, D. Andre, M. A. Keane and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Transactions on Evolutionary Computation,* vol. 1, no. 2, pp. 109-128, 1997.

[38] T. Sripramong and C. Toumazou, "The invention of CMOS amplifiers using genetic programming and current-flow analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 21, no. 11, pp. 1237-1252, 2002.

[39] Y. Hongying and H. Jingsong, "Evolutionary design of operational amplifier using variable-length differential evolution algorithm," in *International Conference on Computer Application and System Modeling*, Taiyuan Shanxi, 2010.

[40] S.-C. Chu, H.-C. Huang, J. F. Roddick and J.-S. Pan, "Overview of Algorithms for Swarm Intelligence," in *Computational Collective Intelligence. Technologies and Applications*, Berlin Heidelberg, Springer , 2011, pp. 28-41.

[41] H. Gupta and B. Ghosh, "Analog Circuits Design Using Ant Colony Optimization," *International Journal of Electronics, Computer & Communications Technologies,* vol. 2, no. 3, pp. 9-21, 2012.

[42] B. Benhala, A. Ahaitouf, M. Fakhfakh and A. Mechaqrane, "New Adaptation of the ACO Algorithm for the Analog Circuits Design Optimization," *International Journal of Computer Science Issues,* vol. 9, no. 3, pp. 360-367, 2012.

[43] S. Kamisetty, J. Garg, J. Tripathi and J. Mukherjee, "Optimization of Analog RF Circuit parameters using randomness in particle swarm optimization," in *World Congress on Information and Communication Technologies*, 2011.

[44] P. P. Kumar and K. Duraiswamy, "An Optimized Device Sizing of Analog Circuits using Particle Swarm Optimization," *Journal of Computer Science,* vol. 8, no. 6, pp. 930-935, 2012.

[45] M. Fakhfakh, Y. Cooren, A. Sallem, M. Loulou and P. Siarry, "Analog circuit design optimization through the particle swarm optimization technique," *Analog Integrated Circuits and Signal Processing,* vol. 63, no. 1, pp. 71-82, 2010.

[46] N. Lourenço and N. Horta, "GENOM-POF: Multi-Objective Evolutionary Synthesis of Analog ICs with Corners Validation," in *Genetic and Evolutionary Computation Conference*, Philadelphia, 2012.

[47] N. Lourenço, R. Martins, M. Barros and N. Horta, "Analog Circuit Design based on Robust POFs using an Enhanced MOEA with SVM Models," in *Analog/RF and Mixed-Signal Circuit Systematic Design*, Berlin, Springer, 2013, pp. 149-167.

[48] T. McConaghy, P. Palmers, M. Steyaert and G. Gielen, "Trustworthy Genetic Programming-Based Synthesis of Analog Circuit Topologies Using Hierarchical Domain-Specific Building Blocks," *IEEE Transactions on Evolutionary Computation,* vol. 15, no. 4, pp. 557-570, 2011.

[49] A. Pradhan and R. Vemuri, "Efficient Synthesis of a Uniformly Spread Layout Aware Pareto Surface for Analog Circuits," in *International Conference on VLSI Design*, New Delhi, 2009.

[50] E. Deniz and G. Dundar, "Hierarchical performance estimation of analog blocks using Pareto Fronts," in *Ph.D. Research in Microelectronics and Electronics*, 2010.

[51] R. Castro-Lopez, E. Roca and F. V. Fernandez, "Multimode Pareto fronts for design of reconfigurable analogue circuits," *Electronics Letters,* vol. 45, no. 2, pp. 95-96, 2009.

[52] E. Roca, M. Velasco-Jiménez, R. Castro-López and F. V. Fernández, "Context-dependent transformation of Pareto-optimal performance fronts of operational amplifiers," *Analog Integrated Circuits and Signal Processing,* vol. 73, no. 1, pp. 65-76, 2012.

[53] G. Gielen, T. McConaghy and T. Eeckelaert, "Performance space modeling for hierarchical synthesis of analog integrated circuits," in *Design Automation Conference*, 2005.

[54] C. Shoou-Jinn, H. Hao-Sheng and S. Yan-Kuin, "Automated passive filter synthesis using a novel tree representation and genetic programming," *IEEE Transactions on Evolutionary Computation,* vol. 10, no. 1, pp. 93-100, 2006.

[55] M. Graphics, "Eldo Classic - Foundry Certified SPICE Accurate Circuit Simulation," [Online]. Available: http://www.mentor.com/products/ic_nanometer_design/analog-mixed-signal-verification/eldo/. [Accessed 2014].

[56] R. Martins, N. Lourenço and N. Horta, LAYGEN II: Analog ICs Layout Generator, Berlin: Springer, 2013.

[57] M. Graphics, "IC Verification and Signoff Using Calibre," [Online]. Available: http://www.mentor.com/products/ic_nanometer_design/verification-signoff/. [Accessed 2014].

[58] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *Evolutionary Computation, IEEE Transactions on,* vol. 6, no. 2, pp. 182-197, 2002.

[59] S. Bandyopadhyay and S. Saha, "Some Single- and Multiobjective Optimization Techniques," in *Unsupervised Classification*, Berlin, Springer Berlin-Heidelberg, 2013, pp. 17-58.

[60] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *International Conference on Neural Networks*, 1995.

[61] R.-S. M. and C. Coello, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," *International Journal of Computational Intelligence Research,* vol. 2, no. 3, pp. 287-308, 2006.

[62] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu and S. Tiwari, "Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition," 2009.

[63] P. Kinget, "Integrated GHz voltage controlled oscillators," in *Analog Circuit Design*, Springer US, 1999, pp. 353-381.

[64] D. B. Leeson, "A Simple Model of Feedback Oscillator Noise Spectrum," *IEEE,* vol. 54, no. 2, p. 329–330, 1966.

[65] T. Lee, The design of CMOS radio-frequency integrated circuits, Cambridge: Cambridge University Press, 1998.

[66] P. K. Rout, U. K. Nanda, D. P. Acharya and G. Panda, "Design of LC VCO for optimal figure of merit performance using CMODE," in *1st Int. Conf. Recent Advances in Information Technology*, Dhanbad, 2012.

[67] B. Perumana and R. Mukhopadhyay, "A Low-Power Fully Monolithic Subthreshold CMOS Receiver With Integrated LO Generation for 2.4 GHz Wireless PAN Applications," *IEEE Journal on Solid-State Circuits,* vol. 43, no. 10, pp. 2229-2238, 2008.

[68] J. Lin, M. Jian-Guo, Y. Kiat-Seng and A. Do, "A novel methodology for the design of LC tank VCO with low phase noise," in *IEEE International Symposium on Circuits and Systems*, Kobe, 2005.

[69] H. Lee and S. Mohammadi, "A subthreshold low phase noise CMOS LC VCO for ultra low power applications," *IEEE Microwave and Wireless Components Letters,* vol. 17, no. 11, pp. 796-798, 2007.

[70] R. Fiorelli, E. Peralias and F. Silveira, "LC-VCO design optimization methodology based on the gm/ID ratio for nanometer CMOS technologies," *IEEE Transactions on Microwave Theory and Techniques,* vol. 59, no. 7, pp. 1822-1831, 2011.

[71] K. Siwiec, T. Borejko and W. Pleskacz, "LC-VCO design automation tool for nanometer CMOS technology," in *IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits & Systems*, Tallinn, 2012.

[72] R. Póvoa, N. Lourenço, N. Horta, R. Santos-Tavares and J. Goes, "Single-Stage Amplifiers with Gain Enhancement and Improved Energy-Efficiency employing Voltage-Combiners," in *21st IFIP/IEEE International Conference on Very Large Scale Integration*, Istambul, 2013.