FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Optimization Design Flow of Integrated Circuits based on Machine Learning Approaches

Tiago da Costa Abreu

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Manuel Cândido Duarte dos Santos (Ph.D)

External supervisor: Miguel Andrade Martins (Ph.D)

July 14, 2019

Abstract

Integrated analog circuit design is more complex than its digital counterpart, since every variation in their synthesis can effectively detune these circuits and render them useless.

As such, it is crucial to take into account the interactions between components such as resistors, inductors, capacitors and transistors while sizing them, and their effects in the final behaviour of these circuits.

Considering that the average modern analog circuit encapsulates dozens, hundreds and sometimes thousands and millions of these components, along with dozens of specifications that must be met in order to assure good circuit performance, this sizing task is very quickly rendered unfeasible without the help of automated sizing tools.

This work aims at analyzing the current options in automated, intelligent sizing of components and applying modern state-of-the-art nature-inspired heuristics to enhance the sizing process, addressing some of the limitations in today's tools.

Resumo

A síntese de circuitos analógicos integrados é considerada, de forma geral mais complexa que a síntese de circuitos digitais, uma vez que pequenas variações no seu processo de fabrico podem inutilizá-los rapidamente.

Assim sendo, é crucial ter em conta as interações entre componentes como resistências, indutores, condensadores e transístores aquando do seu dimensionamento, e o seu efeito no comportamento final destes circuitos.

Assim, e considerando que o típico circuito analógico moderno contém dezenas, centenas e por vezes milhares ou milhões de componentes, e dezenas de especificações a cumprir por forma a assegurar um bom funcionamento em quaisquer condições, esta tarefa de dimensionamento é muito rapidamente tornada dantesca quando realizada sem o auxílio de ferramentas de desenho automático.

Este trabalho procura analisar as opções atuais em ferramentas de dimensionamento automatizadas e inteligentes, e aplicar o estado de arte de heurísticas inspiradas na natureza para melhorar o processo de dimensionamento, adereçando algumas das limitações das ferramentas atuais.

Acknowledgements

I would like to thank my family, who did all they could and so much more, stepping over everything that fell upon them and moving forward.

To my dad, my mom, my uncle Inácio, my grandmothers, who managed to push through and find the health and strength of character to watch me jump head first into the abyss.

I would like to thank my supervisor Professor Candido Duarte for assuring the whole process went smoothly and recognizing the subtle difficulties of writing a dissertation in a corporate environment.

Kind regards to Miguel, Luis, Blaz, Marko, Ali, Dinesh and the whole Intel family who always supported me in whichever way they could and Intel Corporation Germany GmbH for providing the right conditions that enabled this work.

Further regards to Alexandre Truppel and Nuno Fernandes for keeping me sane without even realizing it.

Thank you Chuva and Farida, for blindly believing me.

And last but certainly not least, my biggest word of appreciation to Pedro Silva for helping me find the better person inside of me and quite literally rendering this whole adventure possible.

I owe you the world.

Tiago Abreu

"If you always do what you always did, you'll always get what you always got"

Contents

List of Figures

List of Tables

Abbreviations and acronyms

ASA	Adaptive simulated annealing
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
CAD	Computer-Aided Design
CF	Cost function
GA	Genetic algorithm(s)
IC	Integrated Circuits
IP	Intellectual property
MOO	Multiobjective optimization
MOPSO	Multiobjective particle swarm optimization
MOS	Complementary metal–oxide–semiconductor
NSGA-II	Non-dominated sorting genetic algorithm
OF	Objective function
PSO	Particle swarm optimization
PVT	Pressure, Volume, Temperature
RF	Radio Frequency
SPICE	Simulation Program with Integrated Circuit Emphasis
TTM	Time-to-market
VLSI	Very Large Scale Integration

Chapter 1

Introduction

The biggest breakthrough of the twentieth century was arguably the invention of the integrated circuit. Since then, virtually every active electronic circuit is relying on one or more of these, fulfilling digital, analog or mix-signal processing tasks, such as the systems-on-chip (SoC) present in modern cellphones, which have modules capable of dealing with both analog and digital signals within the same chip.

SoCs with analog parts already account for more than 75% of worldwide SoCs, and the tendency seems to point to an increase, as transistor sizes decrease and Very-Large Scale Integration (VLSI) of these circuits becomes more available. By 2016, Intel could already fit 8 billion 14 nanometer transistors into a single chip. To put things into perspective, it takes 10 seconds for a healthy human nail to grow 14 nanometers.

This integration capacity seems to be increasing exponentially alongside circuit complexity, slowing productivity rate, SoC Time-To-Market (TTM), and circuit reliability.

Although analog circuits usually make up for a smaller part of SoCs than digital circuits, their synthesis process is by far more complex to fulfill and verify. The key of this work is to enhance one of the crucial steps of this process – the circuit sizing.

The tools required to test the current methodology, as well as the circuits and IT were provided by Intel Germany GmbH, which was where this thesis was developed. All confidential information was omitted in order to comply with Intel Corporation Confidentiality Policy.

1.1 Context and motivation

Analog-based Integrated Circuit (IC) synthesis is particularly complex because it requires topology and layout synthesis common to digital circuit design, and additional component sizing (resistances, capacitors, inductors, transformers, pads, etc.) [1].

Understanding the effects of varying each parameter in the final specification values is beyond the scope of the average human mind, and circuit designers often rely on experience from the past to fulfill this complex task [2].

Currently, most industry-standard techniques for circuit sizing optimization include decades-old algorithms, which are associated with a series of limitations and require lots of simulations until the desired specifications are met.

These approaches include gradient-based algorithms, that get easily "trapped" in sub-optimal solutions (local minima) due to their focus on local information and volatile memory of the specifications they are optimizing, stochastic-based algorithms that demand lots of different simulations to reach good solutions (optimal regions of the design space) and fail to explore them efficiently when found, and Newton/quasi-Newton based algorithms which involve completely inverting or approximating the inversion of very big Hessian matrices.

All of these approaches entail more or less big computational costs – evaluated through criteria such as amount of processing units, simulation time, memory usage, etc. –, be it because they explore the design hyperspace in a sub-optimal manner or because they take too many iterations and thus simulations to reach the best possible point, and do not rely so much on parallel information obtained from these.

By addressing the component sizing problem armed with new optimization paradigms, we can drastically reduce the amount of time required, even for a team of experienced designers, to correctly size circuit components in order to meet all the specifications even in the worst situations.

Components are getting smaller in VLSI, as predicted by Moore's Law, and circuits' complexity is increasing accordingly; thus the usage of faster and innovative optimization tools, aiming to reliably increase the pace at which this process is happening [1].

1.2 Objectives

Based on the current state-of-the art in optimization algorithms for analog circuit synthesis, namely nature-inspired ones, the aim of this work was to implement a dynamic, multiobjective-oriented, constrained optimization tool, ideally supporting statistical process variation corners of these circuits [3], and integrate it within Intel's flow for complex analog circuit synthesis, allowing it to make use of the vast resources Intel has at hand.

In other words, the tool must be able to autonomously size a given circuit's n variables in order to achieve the best possible values for m specifications, without any prior knowledge of the circuit.

While algorithms like Simulated-Annealing (SA) and gradient-based algorithms simulate only a handful of possible solutions at the same time and even discard the less than optimal results they do not intend to explore in some cases, nature-inspired algorithms such as the Genetic Algorithms-based (GAs) and Particle Swarm Optimization-based (PSO) algorithms derive info from bad and unfeasible solutions to orient the search towards richer regions of the design hyperspace, working with sets of design scenarios together at the same time to search for the best solutions - the elite.

GAs, for example, are based on the concept of discarding the worst half of all possible solutions – known as individuals – tested together and generating offspring by intertwining the good half, simulating reproduction.

This process happens for every iteration and it has demonstrated good convergence [4]

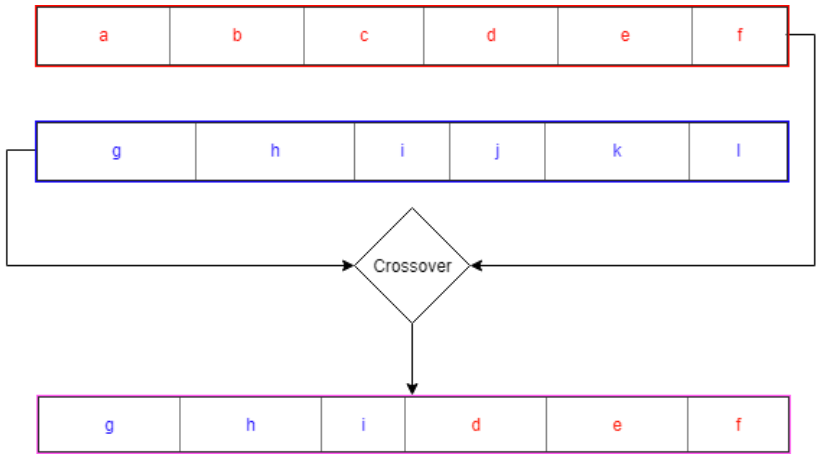


Figure 1.1: Crossover mechanism in GAs

PSO on the other hand, is inspired on swarm behaviour observed in nature, for example in pigeon flocks or fish schools.

The behaviour fundamentally states that individuals build their own knowledge of the search space together in a swarm.

This characteristic is particularly useful while searching for food or avoiding predators.

1.3 Thesis structure

This thesis is organized as follows: Chapter 1 describes the motivation for this work and its placement within the automated analog design landscape.

Chapter 2 describes the different angles from which the analog device sizing has been addressed and some history on the matter.

Chapter 3 describes the issues present in commercially available device sizing optimization tools, while chapter 4 details how these limitations were tackled.

Finally, the results obtained from testing this algorithm are displayed in chapter 5, followed by some suggestions for further development and future works on this matter in chapter 6.

Chapter 2

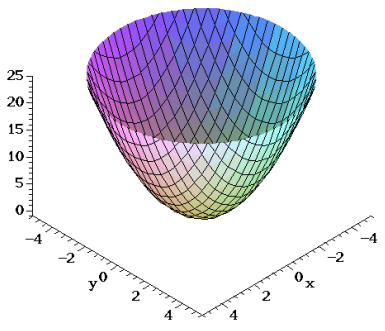
Literature review

Before the emergence of reliable VLSI analog design automation methods and tools, most of the component sizing was done along with the circuit topology selection and layout definition by the hands of an expert or group of experts. This knowledge-based approach relied heavily on the designers' experience and know-how, presenting major drawbacks like the large overhead required to define a new design plan, the reformulation of the entire design plan when expanding the system to new topologies or the migration to other technologies [2].

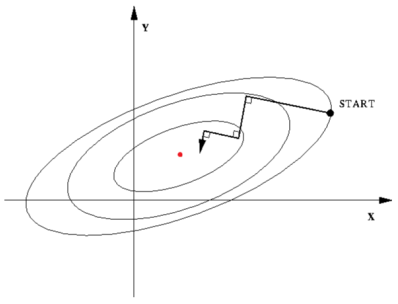
Decades old literature [5] shows the desire of pursuing efficient and effective automated circuit sizing optimization tools has been around for long, and a vast range of gradient-based and Newton-method-based algorithms have been developed over the years, alongside more abstract metaheuristics, like the Nelder-Mead algorithm [6].

Although mathematically robust, these rather old optimization algorithms are fast to converge to local minima and do not have reliable ways of continuing their search in this case, becoming slaves of their own deterministic nature; furthermore, convergence toward a local minima is only assured in convex design problems, such as the function in Figure 2.1a [1, 7].

Figure 2.1b illustrates the zig-zagging, orthogonal movement displayed by the steepest-descent algorithm, rendering it very slow to converge toward minima (red dot in this example).



(a) Plot for $z^2 = x^2 + y^2$, a simple convex function with a single minimum



(b) Steepest-descent

Improvements shown in computer science, with emphasis on nature-inspired metaheuristics [8] and computational power alone entice the pursuit of new solutions.

Nature-inspired algorithms cover a huge volume of the design hyperspace, and have their own techniques to preserve solution diversity, assure hill-climbing capabilities (3.2.2.2) and drive the search with a combination of stochastic components and deterministic ones.

Particle swarm, for instance, has consistently proven to be very effective, showing fast convergence towards global optima at a very low computational cost, for a wide variety of test cases [9].

2.1 Multiple objective world

Over the years, the quality of the results obtained in solving complex mathematical problems using these nature-inspired metaheuristics and their implementation simplicity have also encouraged researchers to pursue applications for these in the real world, where multiobjective problems are much more frequent than the single-objective counterparts.

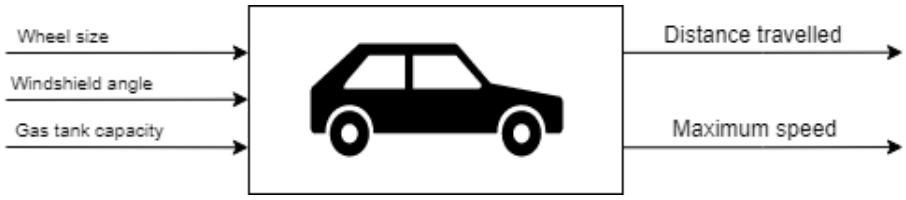


Figure 2.2: Real multiobjective optimization problem

The multiobjective (3.2.3) problem starts when more than one of the system's objective functions are dependent on one or more of the design variables.

Figure 2.2 illustrates a common example: when sizing a vehicle, depending on the design values chosen, for example wheel size and gas tank capacity, different values are obtained for functions that ultimately depend on these, for example distance travelled and the vehicle's maximum speed.

A good rule of thumb for the formulation of a multiobjective algorithm is to apply a balance mechanism to the objective functions being optimized (See section 3.2.3).

The good convergence demonstrated by single-objective PSO algorithms has sprouted a myriad of different multiobjective variations fundamentally based on this algorithm, the so-called Multiobjective Particle Swarm Optimization (MOPSO) algorithms briefly summarized in [10]. The same happened for other simple and essentially single-objective-oriented nature-inspired algorithms, from which genetic algorithm is the worthiest mention.

To avoid getting lost in a sea of different approaches to solving the multiobjective optimization problem, this work's foundation is built upon one of the multiobjective particle swarm optimization algorithms, proposed in [11], due to its fast convergence rate, crucial in engineering optimization, specially within the simulation-based approach adopted (3.1.4.2), its ability to cover the entirety of the Pareto front in the test-cases documented, when compared to other multiobjective, nature-inspired, state-of-the-art counterparts, namely Non-dominated Sorting Genetic Algorithm

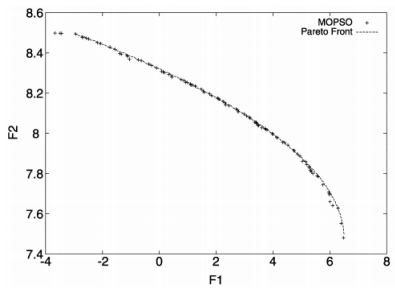
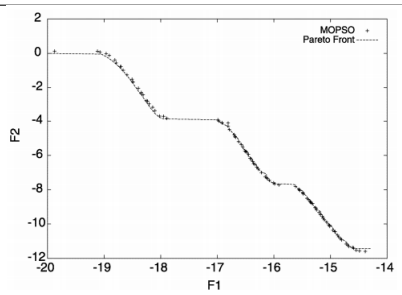
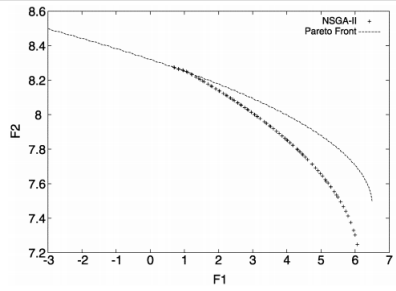
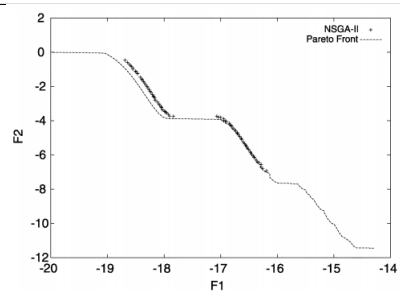
Algorithm	Problem 1 [17]	Problem 2 [18]
MOPSO		
NSGA-II		

Table 2.1: Pareto-fronts comparison

II [12] (NSGA-II), based on the older NSGA proposal, and micro-Genetic Algorithm [13] (micro-GA), and its relative competitiveness when applied to analog circuit sizing [14, 15, 16].

Table 2.1 compares the performances of MOPSO and NSGA-II algorithms for Pareto-front approximation in two different multiobjective optimization problems. The lines represent the true Pareto fronts, while the dots represent the possible solutions used to approximate it.

Since circuit sizing optimization is usually a constrained type of problem, meaning there's certain values for circuit specifications that must be met, the implemented MOPSO was enhanced with constraint-handling (4.2) techniques inspired by the ones presented in [19] and [11].

2.2 Conclusions

Analog circuit design is considered a hard optimization problem and has been used by researchers in classical artificial intelligence, classical optimization, and intelligent systems as a testbench for their methods, and several commercial tools [2].

From all the previously reviewed works we can derive some interest in implementing a constrained multiobjective swarm optimization tool for the automated sizing of analog circuits.

Chapter 3

Problem formulation

3.1 Analog circuit synthesis

Analog design is fundamentally harder than digital circuit design, the latter being able to tolerate higher noise levels due to the binary nature of the signals flowing through them.

Since analog design deals with continuous specification values that need to be optimized, for the circuit to work in a tuned, feasible manner over every possible scenario, it is crucial to take into account factors like device mismatch, layout parasitic effects and changes in environmental conditions, such as temperature and process variations.

The increased complexity and integration of this type of circuits has lead to divide-and-conquer design methodologies, from system level, to block, circuit and device level.

A general design flow is described in the following section, as well as different synthesis approaches and evaluation methods for the designs obtained.

3.1.1 General design flow

A general analog design flow is divided in a range of abstraction levels, the more abstract being the first to be addressed and the more concrete, physical ones the last.

- System level: the overall architecture of the system is designed; different specifications are mapped into intermediate level parameters;
- Block level: high-level functions are translated into individually described blocks.
- Circuit level: where this work's focus will fall upon; device sizing and optimization is conducted, automatically or manually, according to the inherited specifications, and blocks are iteratively tested for compliance.
- Layout hierarchy: schematic is translated into a physical layout with smallest possible area, either automatically or manually, followed by a parasitic verification.
- Fabrication and testing: mask generation and IC production, followed by rigorous quality tests

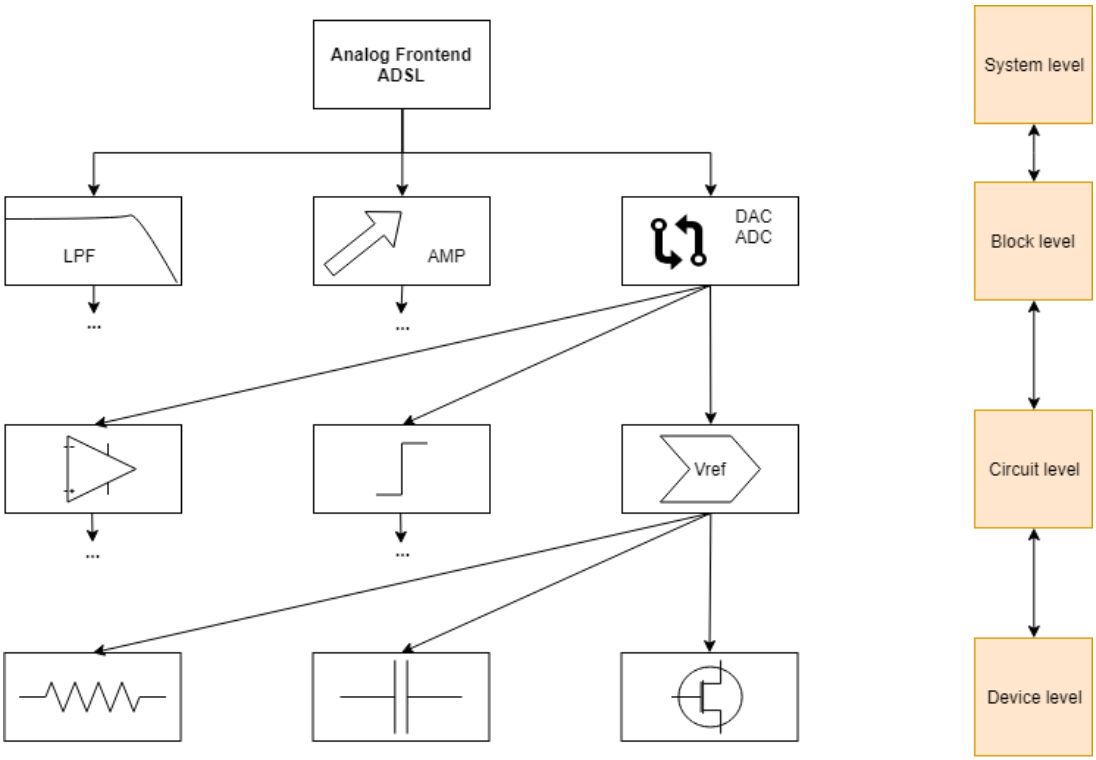


Figure 3.1: Analog synthesis flow example: Analog Front-end ADSL

Taking these abstraction levels into account, the design process usually resorts to a top-down approach while the verification resorts to a bottom-up approach.

This means that the process usually flows from the system level to the block level and circuit level after that, where device sizing sits, while the layout design, parasitic extraction, fabrication and testing is usually performed from the lower levels upward.

A general approach to the design of analog circuit is described in Figure 3.2

3.1.2 Automated design flow

Trends in automated analog design lean towards three main aspects

- Flexibility, allowing the design to interact with every part of the synthesis process
- Modularity, allowing the use of different tools for different design tasks
- Hierarchy, allowing the handling of complex systems

There are tools in place for automated topology selection, layout generation and circuit sizing optimization; this work, however, will only focus on the circuit sizing optimization block, taking in circuits with defined topologies known to work and sizing and verifying on the schematic level only, ignoring repercussions on the layout stage for now.

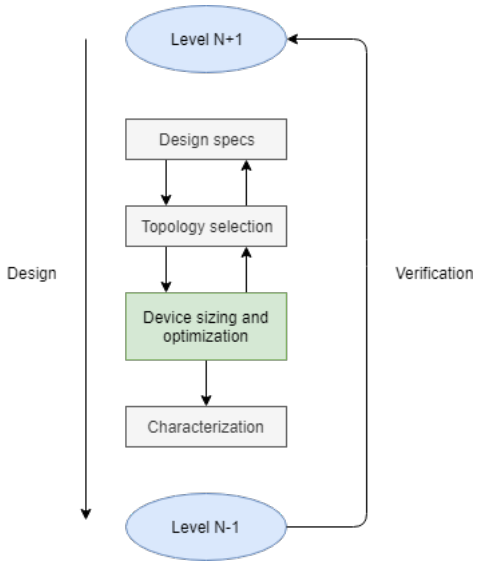


Figure 3.2: Simple top-down analog design flow

3.1.3 Synthesis approaches

3.1.3.1 Knowledge-based

The main purpose of this approach is to encapsulate the designer's knowledge and experience into the design-plan.

When applied to the component sizing task, this means that the designer must know not only how to optimally size every component to achieve the best circuit performance, but also the interactions between the components.

The biggest drawbacks of this approach are the large overhead required to define new design plans and the need to reformulate them when migrating to new topologies or technologies.

3.1.3.2 Optimization-based

This approach, unlike the knowledge-based approach, uses one or a set of optimization engines to iteratively perform the design tasks at hand in an automatic fashion. For circuit sizing, this means that the design variables are iteratively changed – through some optimization mechanism – ideally towards values that guarantee specification compliance and optimal functionality. The circuit's performance is systematically assessed through some evaluation method.

3.1.4 Evaluation methods

3.1.4.1 Equation methods

These methods apply mathematical solution validation to evaluate the design obtained from searching the hyperspace of possible values for circuits' parameters.

For example, in posynomial problems, the objective function, if set with n parameters takes the general format of

$$f(x_1, x_2, ..., x_n) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} ... x_n^{a_{nk}}$$

where x variables represent, for example, resistor, capacitor or transistor length values and

$$c_k \geq 0 \wedge a_{nk} \in \Re$$

.

For modern, highly complex circuits involving multiple objective functions dependent on a lots of variables, the mathematical modelling process can prove daunting, even when resorting to symbolic analysis tools, and the approximations obtained often yield low accuracy, limiting their reliability.

3.1.4.2 Simulation methods

These methods, on the other hand, resort to circuit simulations for performance evaluation.

Presently, the use of SPICE-like simulators is almost generalized and essential to support optimization engines in this field [1].

Moreover, within this approach the same circuit can be optimized several times for different specifications as long as the goal function(s) is(are) adapted; therefore, with this approach, virtually all types of circuits can be sized and optimized with low setup time.

The computational power limitations have been consistently reduced over the years, further underlining the allure of these methods [2].

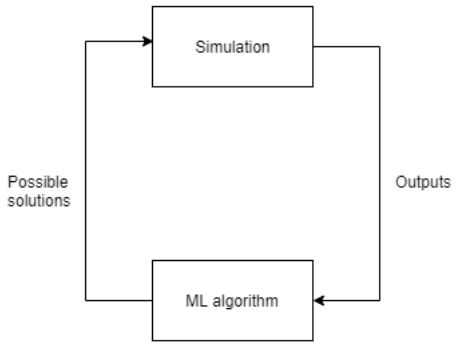


Figure 3.3: Simulation approach

3.1.4.3 Learning methods

As a next step towards fast circuit evaluation, learning mechanisms can, after setup and training, emulate the circuits' response to given sets of design values.

In this context, training means feeding the learning-mechanism with several training samples, that is, pairs of design values and correspondent simulation results obtained previously, with which the mechanism will model a replica of the circuit, as accurate as the amount of data it is fed with.

Although it provides a much quicker evaluation method compared to simulation method, it is heavy to implement and requires large amounts of training data to obtain a reliable model of the system being evaluated.

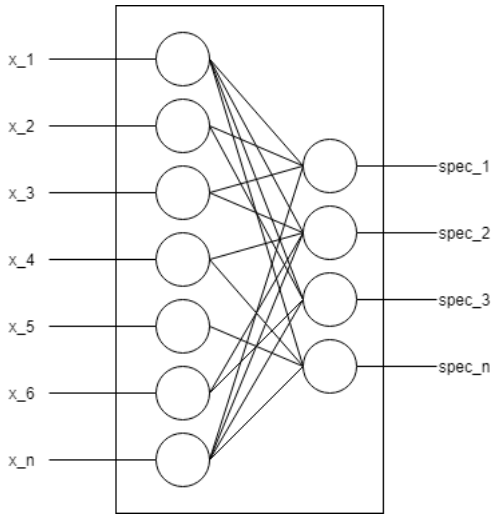


Figure 3.4: Learning-based methods overview. 'x's represent design variables; in this case, circuit parameters, while 'specs' are the circuits' specifications results

In Figure 3.4, an example is given, depicting a two-layer neural network.

3.1.4.4 Overview

Figure 3.5 displays a simple overview of synthesis approaches for any of the analog synthesis flow blocks.

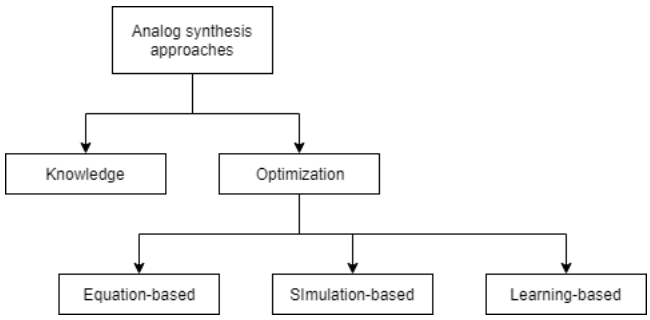


Figure 3.5: Analog synthesis approaches

This work focuses on optimization-based approaches with simulation-based performance evaluation methods – mainly through a SPICE-like simulator.

3.2 Optimization

By definition, optimization is the art of making something better; In mathematical terms, it is generally a hard problem and consists of systematically computing the value or values of x that minimize (or analogously maximize) a given $f(x)$, often referred to as Cost Function (CF) or Objective Function (OF).

This computation can be performed with a myriad of different algorithms, each with its own advantages and disadvantages.

From this point onward, a single variable function will be used for visualization purposes, and any $f(x)$ maximization problem will be considered, without loss of generality, as the minimization of $-f(x)$. The concepts of "minimum" and "optimum" are used interchangeably.

3.2.1 The no-free-lunch theorem

The no-free-lunch theorem states that when considering a broad range of optimization problems, any two optimization algorithms perform equally well on average.

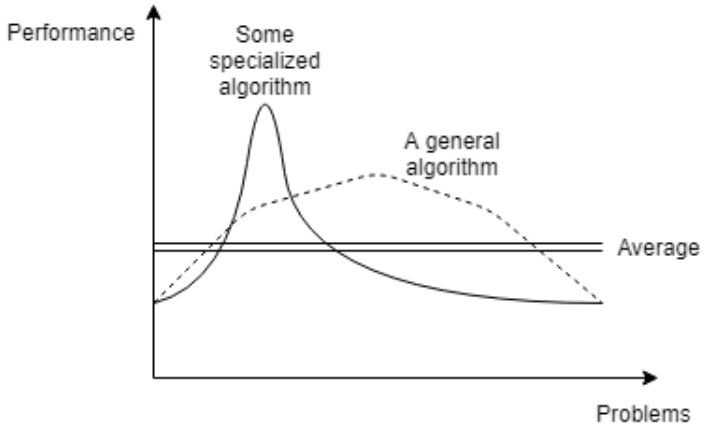


Figure 3.6: The no-free-lunch theorem

Following this line of thought, a good approach to solve any unknown optimization problem is to build a fluid tool that adapts itself to the system it is optimizing as it obtains more information from it.

New nature-inspired algorithms are known for these characteristics as they are usually intuitive and straightforward to implement, and more computationally efficient than their stochastic, Newton, quasi-Newton or gradient-based counterparts. [8].

3.2.2 Optimization fundamentals

3.2.2.1 Local minima trapping and steepest descent

Dating back to the 19th century's Gradient method proposed by Cauchy, gradient-based methods are one of the most explored classes of optimization algorithms.

These first-order, iterative methods rely mostly on first derivative information obtained from the function they pretend to optimize.

The basic iteration principle is

$$x_{i+1} = x_i - \bigtriangledown f|_{x_i} \tag{3.1}$$

For exemplification purposes, we will assume that the feasible design space for these examples verifies $0 < x < 100$.

Figure 3.7 illustrates a simple computation of an arbitrary x_i into x_{i+1}.

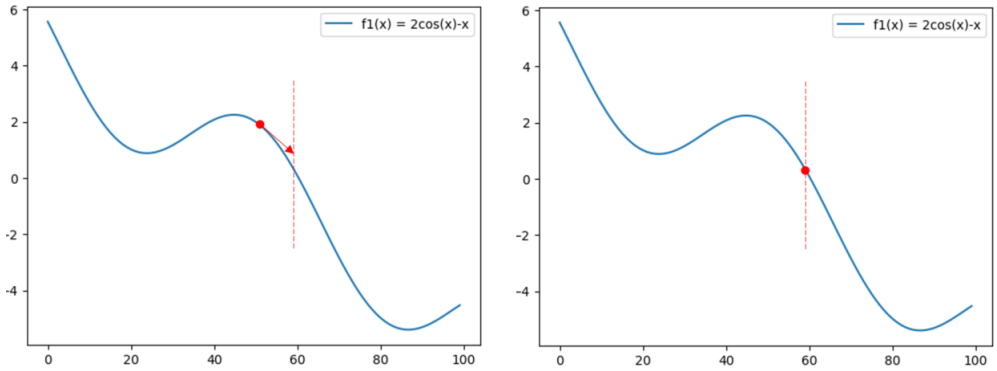


Figure 3.7: Gradient descent behaviour towards minimum

Following this behaviour pattern, it becomes quite simple to foresee a convergence of this specific sample solution toward the global minimum of the function – somewhere in the interval of $80 < x < 100$.

This convergence takes the direction of the so-called steepest descent.

In a different situation however, if the first value x_0 lies within the interval $0 < x < 20$, it is also visible that after iterating, the algorithm leads x towards a value between 20 and 30, corresponding to an $f_x \approx 1.5$, visibly not the best possible solution for this particular case.

This phenomenon is known as local minimum trapping, and poses serious problems in finding globally optimal solutions for more complex problems, involving hundreds, thousand or even millions of variables, and in this case, it emerges not only as a problem of gradient-based algorithms, but also as an initialization problem [20]. In other words, it is clear to see how a well placed initial point – or points, as will be discussed later – can lead to faster and overall better convergence.

In terms of analog circuit sizing, this translates into a higher confidence requirement for the initial circuit design.

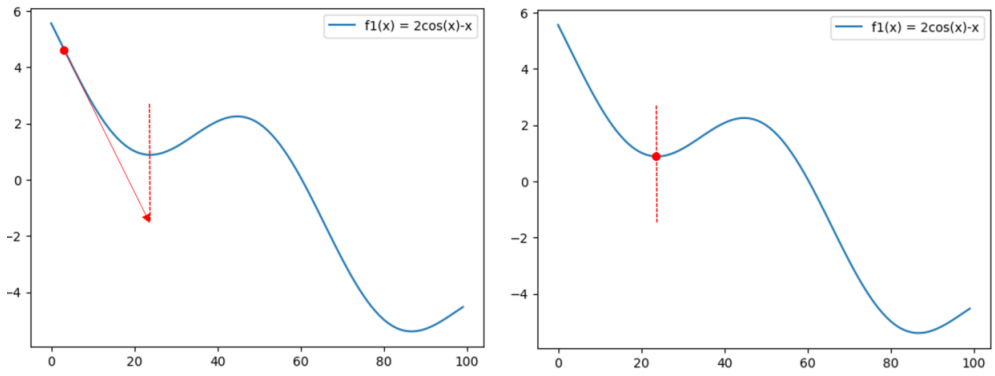


Figure 3.8: Local minimum trapping

An example of a gradient-based algorithm is the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm, which uses first order information of the objective function(s) it is optimizing as well as an iterative approximation of the Hessian of the same function(s).

3.2.2.2 Simulated annealing and stochastic hill-climbing

One of the first and most famous methods to tackle the local minimum trapping issue still being used today, along with its many facets, is simulated annealing, inspired on the physical process of annealing.

In this process, a specific metal object, for example a sword or a vehicle motor part, is cooled down according to a set of temperature plateaus rather than as fast as possible, allowing the metal crystals to form stronger bonds with each other and thus leading to stronger metals.

In simulated annealing, a specified climbing function replaces the crystal energy of the metal [1], unveiling a stochastic hill-climbing ability. In this context, hill-climbing means pushing the search through worse design space regions than previous known ones in hopes of finding a better global optimum.

A brief overview of the algorithm is presented on figure 3.9

The hill-climbing block is originally represented by the inequation

$$rand(0,1) < e^{\frac{\triangle cost}{temp}} \tag{3.2}$$

where $\triangle cost$ is the difference between the old CF value and the new CF value, and $temp$ is the "temperature" coefficient controlled by the algorithm.

The probability of hill climbing is high for high $temp$ and low $\triangle cost$, and low for the analogous case.

Although promising, this algorithm still poses compromises such as the quasi-random or random nature of the point sampling (which can cover undesired areas of the design space), the big amount of steps while hill-climbing, and the overshooting over good minima.

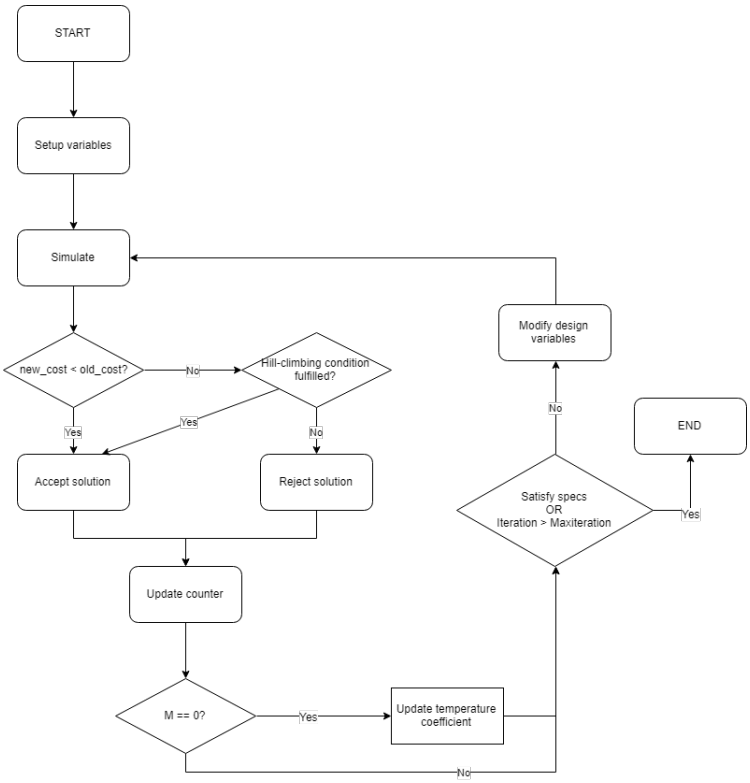


Figure 3.9: Simulated annealing

3.2.3 Multiobjectification

Weighted sum: Over the years, the majority of multiobjective optimization approaches converts the multiple problems into a single objective problem by applying a weighted sum cost function that balances CFs out.

For an optimization of

$$J = J_1, J_2, ..., J_n \tag{3.3}$$

objective functions, weighted sum reduces the problem dimensions to one single-function problem by computing

$$J = \alpha J_1 + \beta J_2 + ... + \phi J_n \tag{3.4}$$

This way, the multiobjective optimization problem becomes the problem of minimizing a single cost function J [21].

Although this approach allows for a direct weighting of the specifications and is generally simpler to implement, there are cases where it does not cover the entirety of the so called "Pareto front", exposed in the next section [22].

Pareto dominance: Another way to look for improvements in solutions is by applying the Pareto dominance concept.

Pareto dominance states that a vector $\vec{u}$ dominates a vector $\vec{v}$ (denoted $\vec{u} \preceq \vec{v}$) if and only if u is partially less than v, that is:

$$\forall i \in \{i,...,k\}, u_i \leq v_i \wedge \exists i \in \{i,...,k\} : u_i < v_i \tag{3.5}$$

For example, when minimizing two specifications, solutions 1, 2, 3 and 4 depicted in Figure 3.10 are said to be nondominated, because none of them guarantees a linear improvement in one of the specifications without damaging the other.

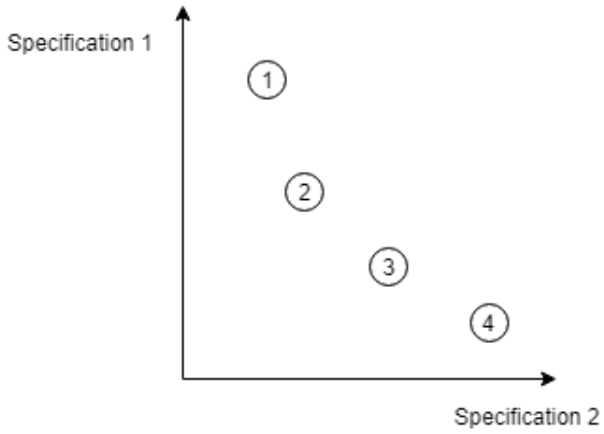


Figure 3.10: Pareto front example

3.2.4 Constraint handling

Although nondominated Pareto-front-based approaches show more promising results than uniquely cost-function based approaches, their purpose is mainly directed towards unconstrained optimization [11, 23].

If however, the system being optimized needs to meet certain criteria, the search should be guided through the best regions of the design space that are also feasible, i.e. that assure specification compliance, or in the worst case scenario, maximize it.

This constraint problem can be formulated as

Minimize

$$f_i(x) = f_i(x_1, x_2, ..., x_n), i = 1, ..., k \tag{3.6}$$

subject to

$$g_j(x) = g_j(x_1, x_2, ..., x_n) < 0, j = 1, ..., q \tag{3.7}$$

$$h_j(x) = h_j(x_1, x_2, ..., x_n) = 0, j = q+1, ..., m \tag{3.8}$$

$$x_j^{min} \leq x_j \leq x_j^{max} \tag{3.9}$$

An efficient and straightforward way to embed these constraints into the algorithm is by applying a healthy balance between some sort of constraint penalties and non dominance when updating the best solution(s) iteratively, since simply rejecting solutions that do not fit the feasible space greatly reduces the algorithms' searching abilities [19].

3.2.5 Cooperation and Coordination

These two intuitive concepts, applied mainly to swarm intelligence, describe some movement rules applied to the particles based on swarm behaviour.

Coordination stands for the particle's ability to remember its best-known position so far and use it as a component when computing future iterations.

Cooperation , on the other hand, stands for the swarm's ability to remember its best known position (or set of positions), otherwise known as as the global best or the elite.

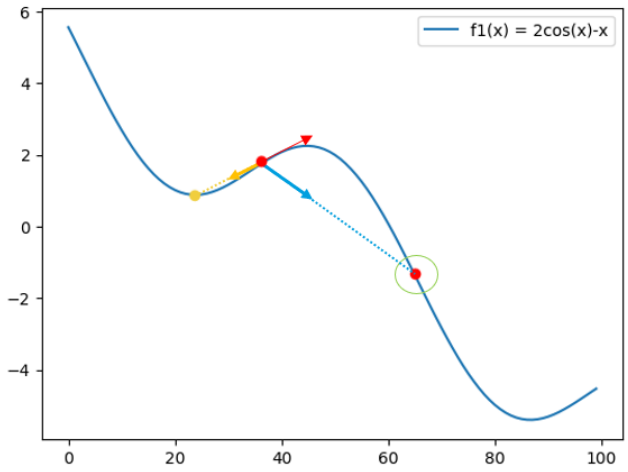


Figure 3.11: Example of cooperation (blue arrow) and coordination (yellow arrow) components in the computation of a new iteration

In particle swarm algorithms, the resulting vector for the particle's dislocation is the weighted sum of the three vectors (red, yellow, blue), where red is the so called inertia vector, pointing in the direction the particle was moving in the last iteration.

For example, by weighting these components with a weight dependent on the number of iterations, the algorithm is forced to focus more on the global component for early exploration, while focusing on the local component for the later stages of the iteration process.

Geographical incidence (Global or Local) is a way of classifying different optimization algorithms, depending on their tendency to either search thoroughly around a given point or explore completely different areas of the design hyperspace.

Figure 3.12 depicts some algorithms' positions in the global-local spectrum.

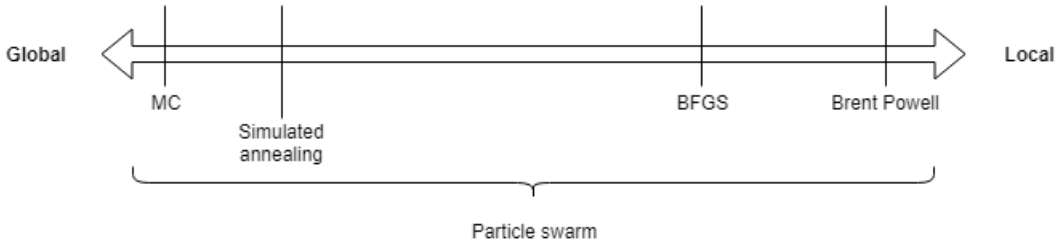


Figure 3.12: Global-to-local spectrum

3.2.6 Multi-threaded optimization

For every optimization problem, the common approach among analog circuit designers is to combine multiple algorithms, hoping to tackle all the different optimization issues. This leads to a very high simulation load and less than optimal resource usage.

Additionally, besides testing a few points per iteration (such as simulated annealing or BFGS algorithms), this cooperation concept is not known to be generalized within the current commercial options.

For big companies like Intel, which usually have powerful simulation resources at hand, it is advantageous to exploit cooperation between multiple simulations.

By employing parallel-oriented algorithms such as genetic algorithms or particle swarm, we can simulate various scenarios at the same time for different particles, extracting information from those particles and iteratively orienting the new population towards what seems to be the best region of the design hyperspace.

This kind of approach allows for a trade-off between time and resources, as most algorithms that simulate one possible solution at a time usually take more iterations to converge but use less resources, while the opposite seems to happen with swarm like and genetic algorithms – naturally parallel-simulation-oriented algorithms.

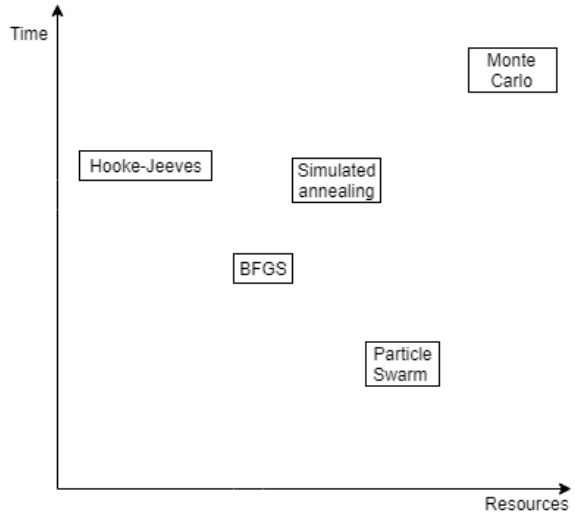


Figure 3.13: Time vs Resources relationship for different optimization algorithms

Chapter 4

Proposed solution

4.1 Particle swarm optimization

Particle swarm optimization is a relatively recent machine learning optimization heuristic.

Its mechanism is inspired by the social and cooperative behaviour displayed by various animal species like birds, fish and humans.

The mechanism consists of a population, hereafter called a swarm, of potential solutions to the optimization problem, called particles. These particles move through the search domain with a dynamic velocity dependent on their personal record of visited positions and their peers', and an inertia value (introduced later on) , in search of an optimal solution for a given objective function or functions.

The underlying mechanics are very straight forwardly described by the two fundamental expressions

$$V_{k+1} = wV_k + c_1 r_1 (P_{best} - X_k) + c_2 r_2 (G_{best} - X_k) \tag{4.1}$$

and

$$X_{k+1} = X_k + V_{k+1} \tag{4.2}$$

where X_k is a matrix containing all particles' design values for iteration k, V_k is a vector containing their correspondent speeds, P_{best} is a matrix containing the best known previous solutions for all particles and G_{best} is a vector containing the best global solution (or solutions, as we'll see ahead). The vectors $P_{best} - X_k$ and $G_{best} - X_k$ represent the so called coordination and cooperation components respectively.

4.2 Multiobjective balance and constraint handling

The multiobjective approach implemented for this work is roughly inspired on the algorithm proposed in [11], and the constraint handling techniques proposed in [19].

A Pareto-ranking scheme is implemented to store the best nondominated solutions in two external, independent repositories, which are analyzed in every iteration of the algorithm.

The first repository, previously introduced as P_{best}, stores the solutions which obtained best specification results for each individual particle. "Best" is defined according to two criteria:

1. Pareto dominance: if the new solution dominates the particle's current best known solution
2. Specification compliance: if more specifications comply with constraints in the new solution

then $P_{ibest} = X_{ik}$, for particle i in iteration k.

The second repository stores the Pareto-front of nondominated global solutions, and accepts new solutions taking into account the Pareto dominance criteria exclusively. If it reaches the maximum number of solutions allowed, however, the first ones to be removed are the ones that failed to comply with the biggest amount of specifications. If all the solutions in the repository meet all specifications, a random one is excluded, although this becomes rarer as the number of specifications increases.

For every iteration and particle, the G_{best} component, hereafter called the *leader*, is randomly selected from the repository.

This way, we can advocate for two different characteristics within the same algorithm.

Integrating solutions that fail to comply with specifications into our repositories allows for a broader exploration of the design hyperspace in early stages of the search [19], and is crucial to a good performance in circuit sizing, where solutions that fit this category are generally hard to come by.

In conclusion, for the local repository, the updating of the "best" known personal solution oscillates between specification compliance and Pareto dominance, while for the global repository, a purely Pareto-dominance acceptance criteria is complemented with a purely specification-compliance criteria for solution disposal.

4.3 Global to local approach

Unlike the majority of commercially available algorithms, in the implemented MOPSO the weights given to the local (coordination and inertia) and global (cooperation) components are not fixed.

Instead, these vary depending on the ration between the current iteration and maximum number of iterations, as well as the number of specs being optimized and the number of solutions in the global repository, emphasizing local search in late stages of the search and global search early on.

4.4 Implementation

The pseudo-algorithm describing the code implemented can be written as such:

1. Setup variables, weights, boundaries

2. Initialize
   (a) V as 0
   (b) X as random.uniform(BOUNDARIES)
   (c) P_{best} as ∞
3. Initial evaluation
   (a) Simulate X
   (b) Get F
   (c) Set $nondominated(F)$ as G_{best}
   (d) Set F as P_{best}
4. Enter WHILE until stopping condition is reached
   (a) Select G_{best}
   (b) Compute speed using $V_{k+1} = wV_k + c_1r_1(P_{best} - X_k) + c_2r_2(G_{best} - X_k)$
   (c) Compute new X_{k+1}
   (d) Check if every X inside the design space
      i. IF not, invert V for said particle
      ii. Set X to the BOUNDARIES value
   (e) Evaluate X
   (f) Get F
   (g) Update G_{best}
      i. Concatenate G_{best} with $nondominated(F)$
      ii. Get $nondominated(G_{best})$
   (h) WHILE G_{best} longer than MAX_SIZE
      i. Remove particle with most failed specs
   (i) Update P_{best}
      i. IF $nondominated(P_{best}, F)$ returns F as dominant
      ii. ELSE IF F has more specs in compliance than current P_{best}
   (j) Check if stopping criteria has been reached
      i. IF yes, break
      ii. ELSE Increment counter

The code was implemented in Python and tested in two different constrained test functions with 100 particles and a *GBest* repository size of 100.

Coefficients w, $c1$ and $c2$ were set to 0.4, 2 and 1 respectively.

The first test case was the Schaffer function N.1 [24] written as

$$Minimize = \begin{cases} f_1(x) = x^2 \\ f_2(x) = (x-2)^2 \end{cases}$$

with

$$-100 < x < 100$$

Figure 4.1 displays the 5^{th} iteration of the algorithm, with a decent spread of solutions across the Pareto front.

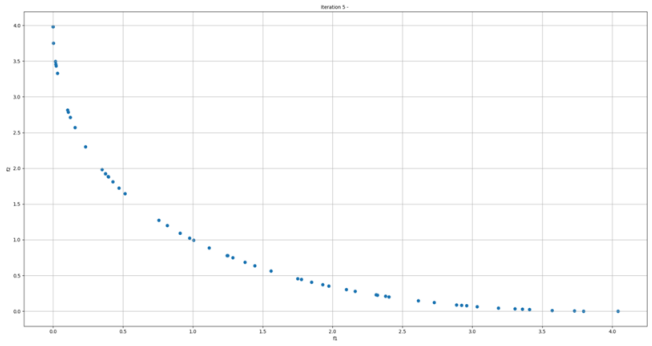


Figure 4.1: 5^{th} iteration of a MOPSO run for the optimization of the Schaffer function N.1

Figure 4.2 displays an overlap between the results obtained and the true Pareto front.

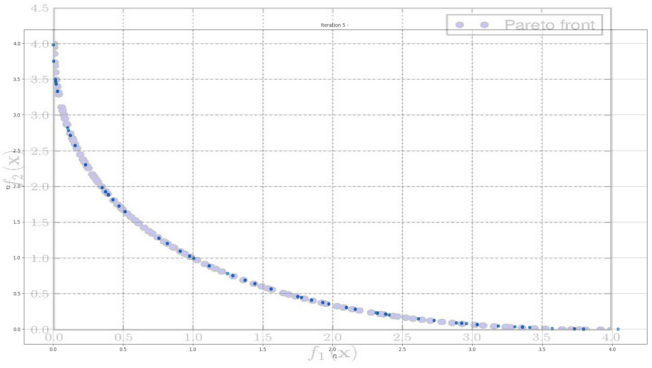


Figure 4.2: 5^{th} iteration of a MOPSO run for the optimization of the Schaffer function N.1 overlapped with the true Pareto front for the same function

The second test case was the Chakong and Haimes function [25] written as

$$Minimize = \begin{cases} f_1(x,y) = 2 + (x-2)^2 + (y-1)^2 \\ f_2(x,y) = 9x - (y-1)^2 \end{cases}$$

and constrained to

$$s.t. = \begin{cases} g_1(x,y) = x^2 + y^2 \leq 225 \\ g_2(x,y) = x - 3y + 10 \leq 0 \\ -20 \leq x, y \leq 20 \end{cases}$$

Figure 4.3 displays the 5^{th} iteration of the algorithm, again with a decent spread of solutions across the Pareto front.

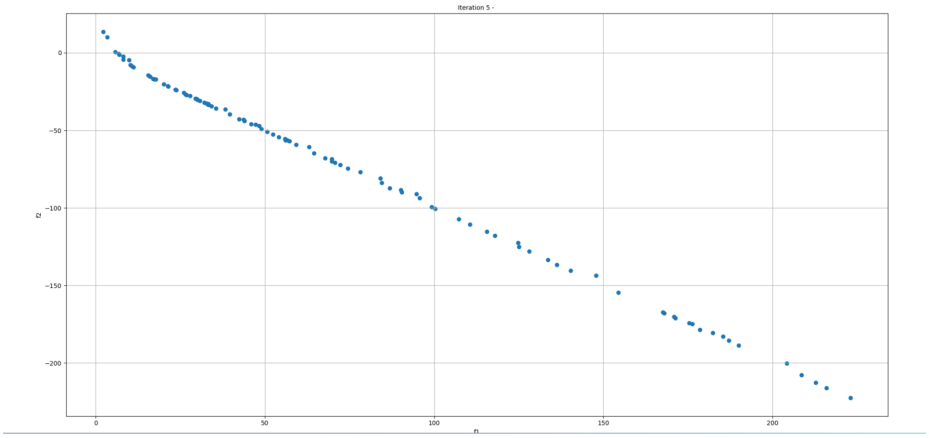


Figure 4.3: 5^{th} iteration of a MOPSO run for the optimization of the Chakong and Haimes function

Figure 4.4 displays once more an overlap between the results obtained and the known Pareto front.

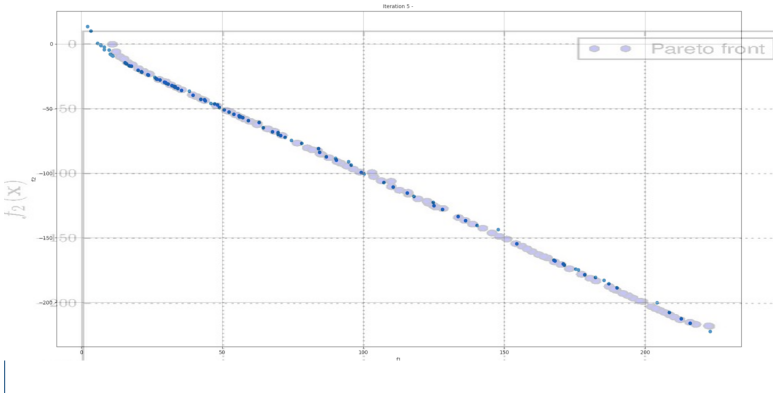


Figure 4.4: 5^{th} iteration of a MOPSO run for the optimization of the Chakong and Haimes function overlapped with the true Pareto front for the same function

Chapter 5

Results

The algorithm was field-tested to size a cascaded operational amplifier, in the context of an analog to digital converter block, bearing 4 objective functions across 5 previously extracted process corners [20], practically handled as 20 constrained objective functions, and 6 design variables (gate length and finger width for 3 different transistors).

After testing all the algorithms available, a best common point was found empirically, written as α point in table 5.1.

The algorithms demanding a mandatory reference point were fed suggested design values.

The following table 5.1 depicts some of the results obtained.

α is the number of iterations until the α point was obtained and overshoot is the likelihood of the algorithm being tested to jump over good points towards worse ones.

PPI stands for particles per iteration, and can translate to the number of simulations being run in parallel.

The overshoot column states a subjective evaluation of the overshooting behaviour observed in the correspondent run - that is, the amount of iterations it took for the algorithm to return to a convergent behaviour after diverging from a known best point.

Run	Algorithm	PPI	# simulations	α	Overshoot
1	MOPSO	30	240	7^{th}	-
2	MOPSO	50	350	6^{th}	–
3	MOPSO	30	120	3^{rd}	-
4	MOPSO	20	100	4^{th}	-
5	Global stochastic (w/o ref)	1-5	180+	180^{th}+	++
6	Global stochastic (w/ ref)	1-5	450+	450^{th}	+
7	Local gradient-based (w/ ref)	2-4	70+	70^{th}+	-
8	Local non-gradient-based (parallel w/ ref)	1-5	50+	50^{th}+	–

Table 5.1: Run results

To be noted that although, in run 6, the global stochastic algorithm tested was initialized with a fairly good starting point, it took longer to find the α point than the same algorithm without a reference point, emphasizing the stochastic nature of the algorithm.

Image 5.1 depicts the first five iterations from the 3^{rd} run of table 5.1.

Orange squares represent the first iteration and blue Xs represent the 5^{th}.

For visualization purposes, only the two specifications that failed for all algorithms were plotted. Although hard to visualize a clear trade-off curve, the tendency points toward the region which provided best specs (around 51 for specification 1, which is being maximized, and 114×10^{-6} for specification 2, which is being minimized).

Even though there is some overshoot within the MOPSO, cooperation between the particles seems to downplay its influence.

This behaviour is prominent in run 5, in which one could observe cases of overshooting taking around 40 to 50 iterations to converge back into the good track – toward the α point.

The role cooperation partakes on this algorithm seemingly renders it much more robust and less prone to divergence than the stochastic global algorithm tested.

Finally, although outperformed by the local algorithms, the MOPSO provides a unique feature consisting of its ability to vary convergence and search detail depending on the size of the swarm set by the user, i.e. the number of possible solutions being simulated concurrently per iteration.

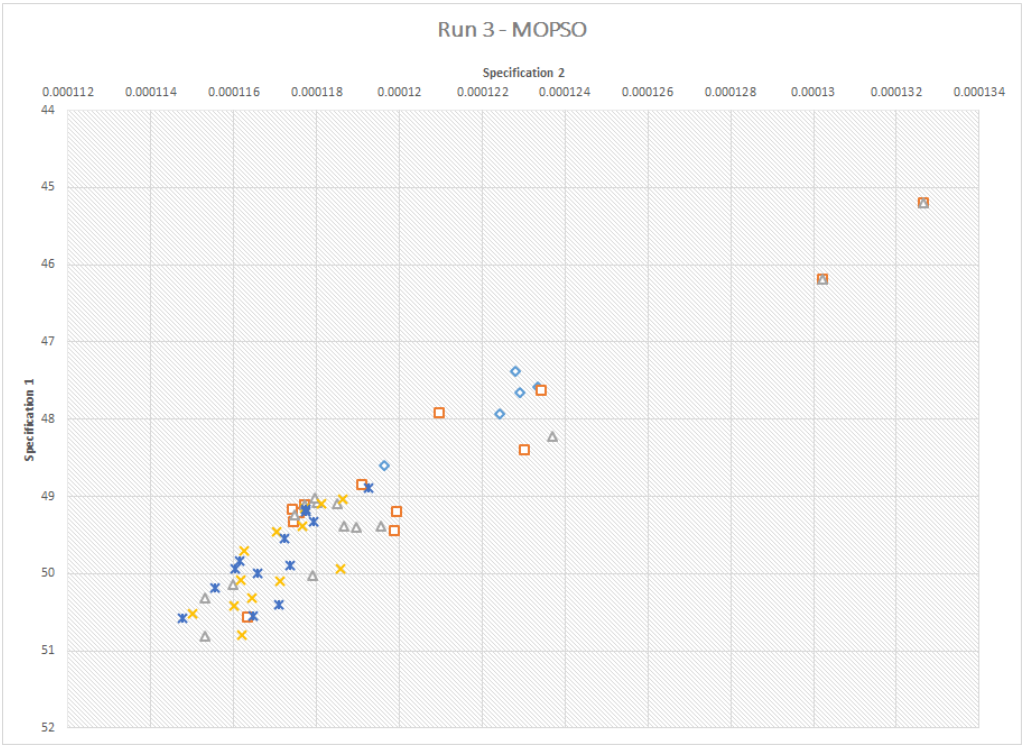


Figure 5.1: First 5 iterations - 3^{rd} run

Table 5.2 shows the first 6 iterations of the first run of in table 5.1.

Once again, only the two specifications that failed for all algorithms were plotted.

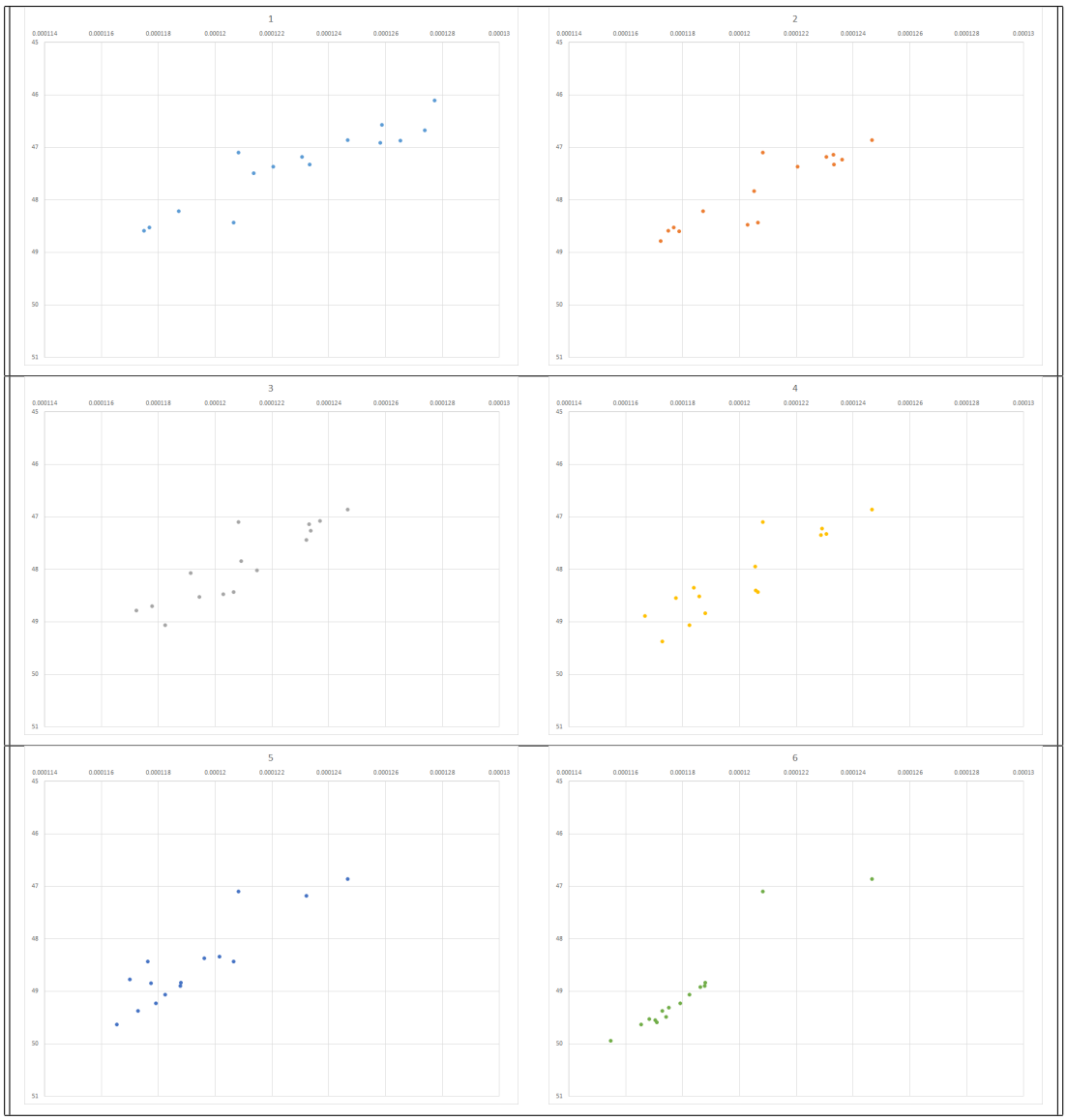


Table 5.2: First 6 iterations - 1^{st} run

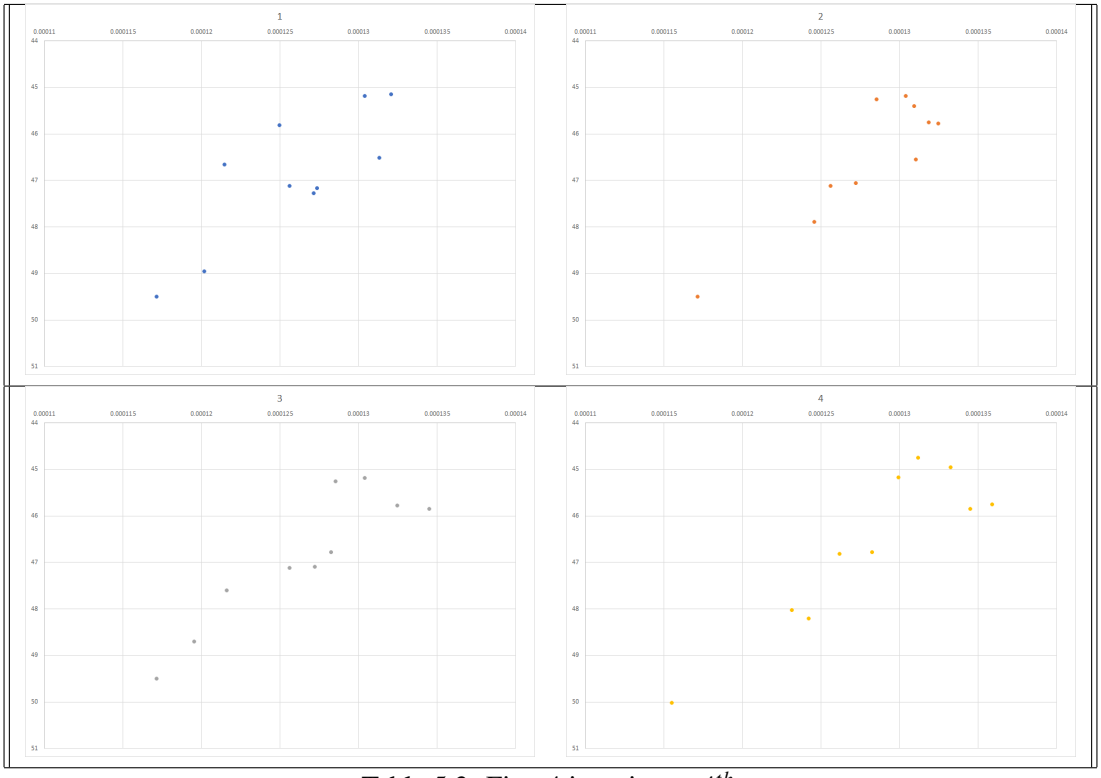


Table 5.3: First 4 iterations - 4^{th} run

Table 5.3 depicts an even less obvious case, with a clear "wandering" behaviour for most particles, but in which one of them finds the correct way toward the optimal point.

The implemented approach will guarantee that solution is not discarded ever unless better ones are found consistently.

Chapter 6

Summary and future work

As solutions stabilize iteratively (meaning particles take increasingly smaller steps over time), a way to continue the search and explore further options could be implemented, for example saving half of the solutions from the repository and applying a genetic algorithm heuristic to the rest. Another promising extension would be the adaptive grid (with quasi-random leader selection), which shows some improvements in efficiently pushing the Pareto front [23].

Although functional, this approach relies on theoretical values for the transistor parameters, not taking into the account the available standard dimensions provided by foundries, and does not take into account FinFET technology.

The specification compliance is performed by the analog design environment tool and not by the algorithm, which can lead to implementation setbacks in the future, specially when migrating to new Intellectual Property (IP).

Algorithm parameter modulation and testing was explored only superficially. More informed parameter tweaking (such as weights, repository size, swarm size, number of iterations, etc.) could lead to a better performance of the tool.

The initialization process is set as uniformly random between each design variable's boundaries. Better sampling processes have been proven more efficient and documentation strongly suggests they could improve the tool's performance [20].

Taking into account the panoply of different options to improve an algorithm that is already competitive, I'm very satisfied with the outcome of this work and am eager to see future improvements being pursued.

References

[1] D. Allstot, J. Park and K. Choi. *"Parasitic-Aware Optimization of CMOS RF Circuits"*. Springer, 2003.

[2] M. Barros, J. Guilherme and N. Horta. *"Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques"*. Springer, 1st edition, 2011.

[3] S. Weber and C. Duarte. *Circuit Design: Anticipate, Analyze, Exploit Variations: Statistical Methods and Optimization*. 04 2017.

[4] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998.

[5] J. Bandler and S. Chen. "Circuit Optimization: The State of the Art". *IEEE transactions on microwave theory and techniques, Vol. 36, No. 2*, February 1988.

[6] X.S. Yang, editor. *Nature-Inspired Optimization Algorithms*. Elsevier, Oxford, 2014. doi:https://doi.org/10.1016/B978-0-12-416743-8.00016-6.

[7] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[8] J. Brownlee. *"Clever Algorithms: Nature-Inspired Programming Recipes"*. LuLu, 1st edition, January 2011.

[9] R. Eberhart, Y. Shi and J. Kennedy. *"Swarm Intelligence"*. Morgan Kaufmann, 1st edition, March 2001.

[10] M. Reyes-Sierra 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*, 2006.

[11] C. Coello, G. Pulido, and M. Lechuga. Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004. doi:10.1109/TEVC.2004.826067.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002. doi:10.1109/4235.996017.

[13] C. Coello and G. Pulido. Multiobjective optimization using a micro-genetic algorithm. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, GECCO'01, pages 274–282, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[14] M. Fakhfakh, A. Sallem, M. Bougharíou, S. Bennour, E. Bradai, E. Gaddour and M. Loulou. *Analogue Circuit Optimization through a Hybrid Approach*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. doi:10.1007/978-3-642-21705-0_11.

[15] 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*, 63(1):71–82, Apr 2010. doi:10.1007/s10470-009-9361-3.

[16] S. Kamisetty, J. Garg, J. Tripathi, and J. Mukherjee. Optimization of analog rf circuit parameters using randomness in particle swarm optimization. In *2011 World Congress on Information and Communication Technologies*, pages 274–278, Dec 2011. doi:10.1109/WICT.2011.6141257.

[17] H. Kita, Y. Yabumoto, N. Mori and Y. Nishikawa. Multi-objective optimization by means of the thermodynamical genetic algorithm. In H. Voigt, W. Ebeling, I. Rechenberg and H. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN IV*, pages 504–512, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[18] F. Kursawe. A variant of evolution strategies for vector optimization. In H. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 193–197, Berlin, Heidelberg, 1991. Springer Berlin Heidelberg.

[19] Y. Woldesenbet, G. Yen, and B. Tessema. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525, June 2009. doi:10.1109/TEVC.2008.2009032.

[20] L. Abreu. "Study of new paradigms of integrated circuit simulation". Master's thesis, Faculdade de Engenharia da Universidade do Porto, s/n, R. Dr. Roberto Frias, 4200-465 Porto, Portugal, 2019.

[21] I. Kim and O. de Weck. Adaptive weighted sum method for multiobjective optimization: A new method for pareto front generation. *Structural and Multidisciplinary Optimization*, 31:105–116, 02 2006. doi:10.1007/s00158-005-0557-6.

[22] W. Jakob and C. Blume. Pareto optimization or cascaded weighted sum: A comparison of concepts. *Algorithms*, 7:166–185, 03 2014. doi:10.3390/a7010166.

[23] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, June 2000. doi:10.1162/106365600568167.

[24] J. Schaffer. Some experiments in machine learning using vector evaluated genetic algorithms. 01 1985.

[25] V. Chankong and Y. Haimes. *Multiobjective Decision Making: Theory And Methodology*. 01 1983.