1

Review of the literature

Part I: Microelectronics design

1.1 Introduction

1.2 3D integration

The benefits of using 3D-ICs are numerous and have been already pointed out in the literature very often over the past few years [1]. First, by adding vertical dimension to the construction of the physical IC we can increase the IC packaging density. This means more gates for the same circuit footprint, that is much higher functional complexity of the final circuit for the same packaging volume. Secondly, the 3D-SICs are expected to have much better computing/power dissipation ratio. The integration in the 3rd dimension allows the design of circuits with different parts being closer to each other, resulting in less and shorter wires [2]. Lowering wire delays and allowing higher operating frequencies will result in increased bandwidths between nodes satisfying data hungry applications. Also, less and shorter wires mean lower total parasitic capacitance and inductance of the circuit, resulting in lower power dissipation. Finally, the 3D-SICs will enable the design of really heterogeneous systems, embedding not only traditional digital circuits such as processors and memories, but also analogue circuits such as sensors, antennas and power supplies [3].

Currently different technologies for fabrication of 3D-SICs have been proposed in the literature. Proposed methods have been used for implementation of complete

systems going far beyond simple proof-of-concept or feasibility demonstrators. One can mention the implementation of a processor and memory in a single 3D chip dedicated for video coding applications [4] and a processor with multiple levels of memory hierarchy dedicated for high-throughput server applications [5]. Finally, the first commercial 3D-SIC products have been already announced by IBM [6] and companies specialized in 3D semiconductor industry such as Tezzaron [7].

3D Integration is taken into account in the roadmaps of almost all key players in the field of integrated circuit design and manufacturing.

Interconnection length

The 3D integration allows to design circuits with components closer to each other. Wire of a few millimetres long can be replaced by TSV of a few tens of microns [1], as shown in Fig. 1.1. These shorter interconnections will introduce shorter delays, hence allowing higher working frequencies.

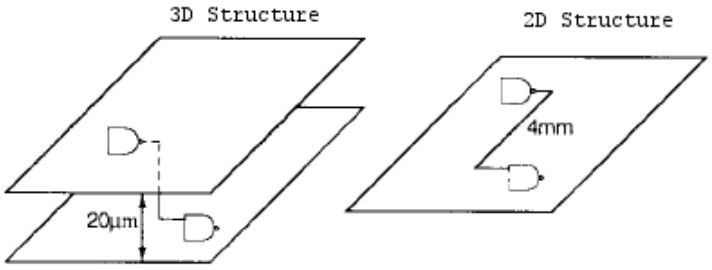


Figure 1.1: Shorter interconnections

Silicon efficiency and accessibility

Adding a vertical dimension allows to increase the integration density. It is therefore possible to have more logic gates than a 2D-IC for the same footprint, hence a more efficient use of the silicon as shown in Fig. 1.2. For instance, compared to the footprint of a 2D-IC, the 3D-SICs can double the integration for a 50% use of a 2D footprint [1].

In addition, the 3D integration allows a better accessibility for the components, as shown in Fig. 1.3. Indeed, for a 2D structure, 8 accessible neighbours can be considered for a central element (Fig. 1.3 (a)), whereas for a 3D structure, the number of accessible neighbours can reach 116 (Fig. 1.3 (b)) [1].

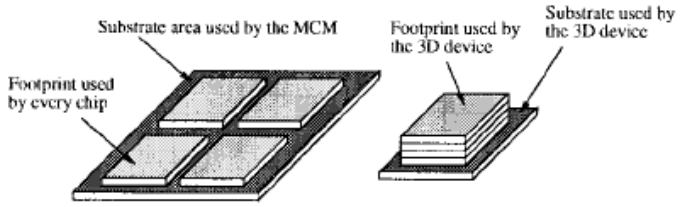


Figure 1.2: Silicon efficiency

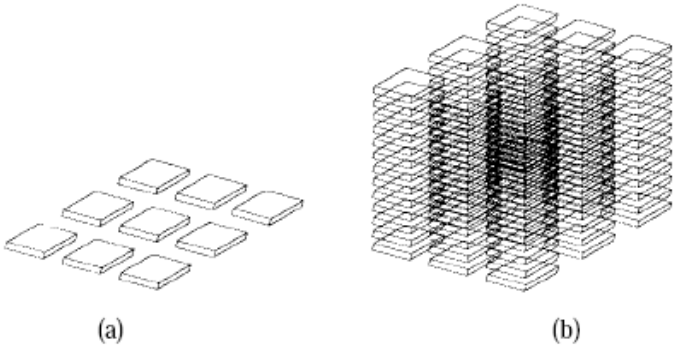


Figure 1.3: Components accessibility

Bandwidth

The use of TSVs on 3D-SIC can significantly increase the bandwidth of a circuit. Indeed, as shown in Fig. 1.4, the interconnections are not only limited to peripheral connections but can also make use of the circuit's surface. This increase of the bandwidth allows higher working frequencies so that it is easier to satisfy data-heavy applications.

Consumption and noise

Shorter interconnections generally translates into lower capacitance and inductance parasites. This means a decrease of the numbers of repeaters, hence a better consumption, less noise and less jitter.

Heterogeneous circuits

The 3D technologies allow truly heterogeneous designs. For instance, it is possible to integrate, in addition to traditional digital circuits of different technologies, analogical circuits such as sensors or antennas, as well as power supply, which give

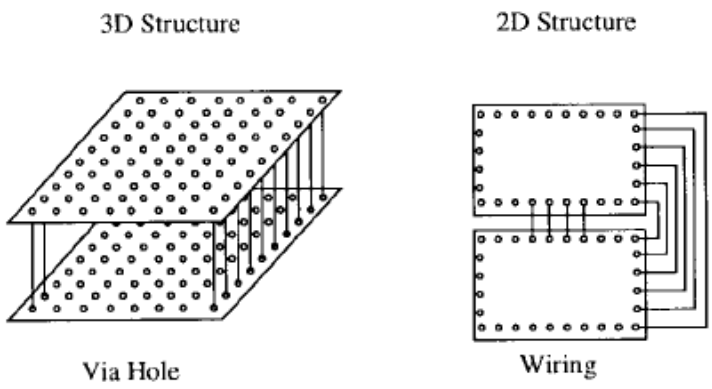


Figure 1.4: Bandwidth improvement

3D-SIC a high degree diversity [3]. The Fig. 1.5 shows a schematic view of a 3D-SIC developed by IMEC for biomedical purposes that contains antennas, DSPs, EEG/ECG sensors, a power supply and solar cells [8].

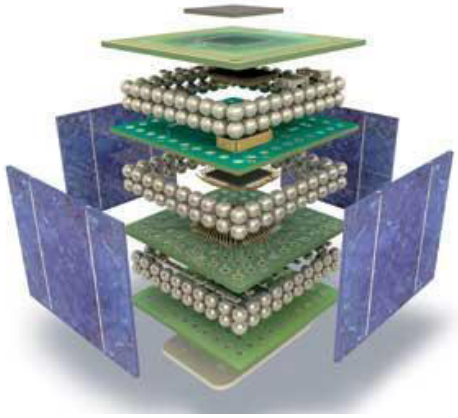


Figure 1.5: Schematic illustration of an heterogeneous 3D-SIC (developed by IMEC) [8]

1.2.1 Technologies de fabrication

Several 3D manufacturing technologies have been proposed and have been used to implement complete systems. Among the existing possibilities, four major categories of methods that illustrate 3D integration can be cited [1, 9].

Chip stacking

This methods consists in stacking components that have been designed and tested separately to produce a system-in-package (SiP). The vertically-stacked chips are interconnected with traditional wirings. The principal advantage of this method is an improvement in terms of size. The wirings are shorter however the components integration density is not increased compared to a 2D system.

Transistor stacking

The transistor stacking consists in creating several transistors level on one substrate. This should be the better way to manufacture 3D circuits although the success rate are currently limited due to thermal issues. The required temperatures to create a layer of high-performance transistors would provoke the destruction of the copper and aluminium already laid down on the previous layer.

Die-on-wafer stacking

In this method, known good dies (KGD), which are functional tested chips, are connected to a host wafer containing other KGDs. These KGDs can be interconnected with organic glues, oxide or metal bonding. The wafer and the bonded KGDs are then shaped to create the interconnections. Different substrates can be combined if the required temperature is low enough to minimize non-homogeneous expansion effects.

The die-on-wafer stacking can use interconnections on the edges of the chips or through-die. Depending on the interconnection type, this method can produce a better integration level than the chip stacking, with a better cost per connection ratio and a higher interconnection density, while holding the advantages of the KGDs.

The quality of the stacking depends on the pick-and-place equipment which is used to position the dies on the wafer. The placement accuracy will determine the possible interconnection density. Also, current equipments are supposed to handle fully buffered chips, not naked circuits so it does not provide protection to static discharge.

Wafer-level stacking

This methods consists in bonding entire wafers into a stack. The vertical through-wafer connections are made directly trough each substrate to the next wafer and it transistors layer. Similarly to the previous method, the interconnection density rely on the precision of the alignment, which is however currently better than the die-on-wafer stacking. This greater accuracy implies a better cost per connection ratio and a higher interconnection density compared to the die-on-wafer stacking.

The use of mixed substrates is also possible, only limited by the process temperatures. All the processing is done at the wafer level so wafer handling equipments are used. Since these provide protection to static discharge so there is no need to include buffering between the layers. The methods to bind two wafers are the same that are available for the die-on-wafer method.

One drawback to wafer-level stacking is its efficiency, since the chips on a wafer are not all KGDs.

2

Review of the literature

Part II: Operations research

Chapter abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

2.1 Introduction

In this chapter, we briefly present the basics of multi-objective optimization and multi-criteria decision aid, in order to justify our choice to use such a paradigm. As stated in Chapter 1, the 3D integration can offer new perspectives but designing 3D-SICs includes two major distinctive features: multiple criteria and a huge number of possible solutions. When facing such problems, two main methods exist: the uni-

criterion paradigm and the multi-criteria paradigm. For optimization problems, these paradigm will refer to the terminology mono-objective/multi-objective optimization while for decision aid, the terminology uni-criterion/multi-criteria will be used.

In the following, we will briefly describe each paradigm, showing some of the main approaches alongside illustrative examples. We will first present the uni-criterion methodology, show why it can be limited in our contect and explain why a multi-criteria paradigm can be more suitable.

2.2 The uni-criterion paradigm

2.2.1 Problem formulation

An optimization problem can be formulated, without loss of generality, as [10]

$$\begin{aligned} &\min f(x) \\ &x \in A \end{aligned} \tag{2.1}$$

where f is a real-valued function evaluating the solutions denoted x, and A is the set of solutions, f is also called the *criterion* on which x is evaluated. Let us note that the equation 2.1 expresses a *minimization* problem. A *maximization* problem can be seen as a minimization problem with the identity

$$\max_{x \in A} f(x) = -\min_{x \in A}(-f(x))$$

so that there is no loss of generality by using only *minimization* formulation.

In order to give a more precise idea of what an optimization problem is, we will describe in the next section some typical examples taken from the reference book [10, 11].

2.2.2 Examples of typical optimization problems

Linear programming

Linear programming (LP) is a problem formulation where the aim is to optimize a linear function, subject to linear inequality constraints. This can be formulated as follows:

$$\min \mathbf{c}^\intercal \mathbf{x} \tag{2.2}$$

subject to

$$\begin{aligned} A\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned}$$

where $\mathbf{x}$ is a vector of continuous, integer or boolean variables to be determined, $\mathbf{c}$ and $\mathbf{b}$ are vectors of coefficients, A is a matrix of coefficients.

Efficient exact methods for solving LP problem exist such as, among the most knowns, the simplex algorithm [12] or the interior point method [13].

Example 2.1 (Linear programming). *A given company produces two electronic boards $Board_1$ and $Board_2$ based on two kinds of memories M_1 and M_2. The objective consists in finding the most profitable product mix, given the availability of each memory M_1 and M_2, and for each board $Board_i$ the used amount of memories and the profit, as shown in Table 2.1. The decision variables are x_1 and x_2 that represent respectively the amount of $Board_1$ and $Board_2$. The objective is to maximize the profit. The problem can be formulated as an LP:*

$$\textit{max profit} = 5x_1 + 4x_2 \qquad \blacksquare$$

subject to the constraints

$$\begin{aligned} 192x_1 + 128x_2 &\leq 1024 \\ 32x_1 + 64_x2 &\leq 192 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Table 2.1: Data associated with the LP problem

	Usage for $Board_1$	Usage for $Board_2$	Availability
M_1	192	128	1024
M_2	32	64	192
Profit per unit	€5	€4	

Integer linear programming

Integer linear programming deals with linear problems where the variables are restricted to be integers:

$$\min \mathbf{c}^\intercal \mathbf{x} \tag{2.3}$$

subject to

$$A\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq \mathbf{0}$$
$$\mathbf{x} \in \mathbb{N}$$

where $\mathbf{c}$ and $\mathbf{b}$ are vectors and A is a matrix of coefficients.

When the decision variables are both discrete en continuous, the problem refers to **mixed integer programming** (MILP).

Other particular ILP problems which deals with variables that are restricted to be either 0 or 1 are called **0-1 linear programming**.

Example 2.2 (Travelling salesman problem (TSP) [11]). *This is one of the most known optimization problem. It can be formulated as follows: given n cities and the distance between each pair of cities, we have to find the shortest tour that visits each city once and returns to the origin city. This problem can be formulated as an ILP problem.*
Let d_{ij} be the distance between the city i and the city j, S be the set of solutions (tours) and define:

$$x_{ij} = \begin{cases} 1 & \textit{if the path goes from city } i \textit{ to city } j \\ 0 & \textit{otherwise} \end{cases}$$

The ILP formulation is then:

$$\min \sum_{i=0}^{n} \sum_{j=0, j\neq i}^{n} d_{ij} x_{ij}$$

s.t.

$$\begin{aligned}
&\sum_{i=0, i\neq j}^{n} x_{ij} = 1 && j = 0, \ldots, n \\
&\sum_{j=0, j\neq i}^{n} x_{ij} = 1 && i = 0, \ldots, n \\
&\sum_{i\in S, j\notin S} x_{ij} \geq 1 && \forall S \subset \{1, \ldots, n\} \\
&0 \leq x_{ij} \leq 1 && \forall i, j \\
&x_{ij} \in \mathbb{N} && \forall i, j
\end{aligned}$$

■

Non-linear programming

Non-linear programming (NLP) models deal with mathematical problems where some of the constraints and/or the objective function are non linear:

$$\min f(x) \tag{2.4}$$

where

$$f : \mathbb{R}^n \to \mathbb{R}$$
$$x \in \mathbb{R}^n$$

subject to

$$g_i(x) \leq 0, i \in J = 1, \ldots, m$$

where $g_i : X \to \mathbb{R}^n$ are the inequality constraints.

NLP are generally more difficult to solve than LP [11] and metaheuristics (see Section 2.4.2) are commonly used to solve this class of problems.

2.3 From the uni-criterion paradigm to the multi-criteria paradigm

With a uni-criterion paradigm, the optimization of one criterion is generally performed while considering that this single criterion synthesizes all the characteristics of the problems or that the other criteria already satisfy an acceptable level. This methodology will try to give a solution which is supposed to be optimal according to this criterion. However, most problems encountered in the field of IC design, and more generally in other industrial fields, contains several conflicting criteria as it will be illustrated in Section **??**. Finding a solution that simultaneously optimizes all the criteria is only possible in rare cases and if optimality can be reached.

For instance, when designing ICs, a manufacturer will try to simultaneously maximize the performance while minimize the cost of the circuit. However, we can already guess that those two objectives are conflicting. Also, producing high-end ICs can be subject to more difficulties in terms of thermal dissipation. In addition, a criterion based on ecological standards may have impacts on the cost and the performance of an IC.

This example shows that a uni-criterion approach cannot always be applied since there is no achievable optimum as several criteria have to be simultaneously taken into account. A solution that optimizes one criterion will likely to affect another.

In order to deal with the multiple criteria of a problem, another paradigm consists in taking into account all the criteria simultaneously. This is the aim of the multi-criteria paradigm which aim to:

1. find the solutions that are efficient on all the criteria simultaneously with the multi-objective optimization;
2. provide support to a decision maker facing several conflicting solutions with multi-criteria decision aid (MCDA) that allows to highlight such conflicts and therefore obtain a compromise with a transparent process.

2.4 The multi-criteria paradigm

2.4.1 Problem formulation

A multi-criteria problem can be formulated without loss of generality as follows [10]:

$$\begin{gathered}\min\{f_1(x), f_2(x), \ldots, f_m(x)\} \\ x \in \mathcal{A}\end{gathered} \tag{2.5}$$

where $\{f_1(x), f_2(x), \ldots, f_m(x)\}$ is a set denoted $\mathcal{F}$ of m evaluation criteria that needs to be minimized and x is a solution of the set $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$.

As explained in Section 2.3, an optimal solution can be impossible to find for a multi-criteria problem. However, compromise solutions can exist and in order to identify them, a dominance relation has been defined [10]:

Definition 2.3 (Dominance). *A solution a_1 dominates a solution a_2 if:*
- *a_1 is as least as good as a_2 on all criteria;*
- *a_1 is strictly better than a_2 on at least one criterion.*

From this dominance relation, it is then possible to filter the solutions in order to keep only the non-dominated ones. This set of *efficient* solutions is called the Pareto frontier. Let us note that the *efficient* solutions refer to the decision space while the Pareto frontier refers to the evaluation space.

Two approaches can be used to establish this set [14]:
- *Exact methods* which aims to compute the Pareto frontier directly [15, 16].
- *Approximate methods* which are based on metaheuristics to quickly explore the solution space and approach as best as possible the Pareto optimal frontier [11].

As explained in Section **??**, designing 3D-SICs includes a huge solution space to deal with in the optimization process. The solution (that is to say the most-suitable 3D-SIC architecture) is unknown and an exhaustive search would take a prohibitive time. Also, due to the nature of the criteria (discrete and continuous variables, linear and non-linear criteria) that will be defined in Section **??**, we have few hopes to be able to develop an exact method. For those reasons, approximate methods with metaheuristics for multi-objective optimization will be used. Let us also remind that the aim of this thesis is to show the applicability of a multi-criteria paradigm to the design of 3D circuits. Therefore developing exact methods has been kept out of the scope of this work.

2.4.2 Metaheuristics for multi-objective optimization

Metaheuristics are a family of approximate optimization methods. They aim to provide "acceptable" solutions in reasonable time for solving complex problems [11]. As stated previously, the optimal solution of a multi-objective optimization problem (MOP) is not a single solution but a set of solutions defined as Pareto optimal solutions. The main goal is therefore to obtain this set.

In our study, due to the heterogeneous nature of the criteria, there are few hopes to find the exact Pareto optimal solutions. In such cases, metaheuristics are commonly used and the goal is then to find an approximation of this set. Two properties has to be respected in order to ensure good approximations: convergence to the Pareto optimal front and uniform diversity. The first property allows to have solutions that are closed to the Pareto set whereas the second property shows a good distribution around the Pareto front.

Numerous metaheuristics have been developed since the 50s. Among the most known, let us cite genetic algorithm [17], scatter search [18], simulated annealing [19], tabu search [20], memetic algorithms [21] and ant colony optimization [22].

In this work, we will focus on genetic algorithms (GA) as they are quick to implement for a first approach and are suitable to heterogeneous variables problems. More details about other metaheuristics can be found in reference books such as [11,23,24].

2.4.3 Genetic algorithm

Genetic algorithm has been developed by Holland in the 1970s [17]. It is a metaheuristic that reproduces the properties of a natural selection process as described by Charles Darwin. GA is based on the principle of the improvement of the gene pool of a population over generations. GAs will mimics the natural evolution with techniques such as selection, crossover and mutation. In the following, we will briefly describe the general methodology of a GA without considering a multi-objective case since the key steps are similar. Afterwards, we will describe one of the most popular multi-objective genetic algorithms: NSGA-II (Non-dominated Sorting Genetic Algorithm).

Genetic algorithms rely on a population that is evolved toward better solutions or individuals. The evolution is an iterative process and starts usually with a randomly-generated solutions. At each iteration, every individual is evaluated to define its fitness. The fitter ones are more likely to be selected for genetic modifications (crossover and possibly mutation). The produced solutions constitutes the new generation that will be used for the next iteration. The algorithm is commonly terminated when a maximum number of generations has been produced or when a certain fitness level has been satisfied. The general pseudo-code for genetic algorithms is shown in Algorithm 1.

Algorithm 1: General pseudo-code for genetic algorithms

```
1 CHOOSE initial population;
2 EVALUATE each individual's fitness;
3 repeat
4 |   SELECT parents;
5 |   CROSSOVER pairs of parents;
6 |   MUTATE the resulting offspring;
7 |   EVALUATE the new candidates;
8 |   SELECT individuals for the next generation;
9 until TERMINATION CONDITION satisfied;
```

Representation of a solution

The representation or encoding of a solution is called a chromosome and depends on the problem. Several examples of problems show binary encodings however, in our study we will use a real-valued matrix that will be detailed in Section **??**. Nevertheless, without loss of generality, we will illustrate the principles of a genetic algorithm by using binary-coded solutions.

Initialization

Initially many solutions are generated, usually randomly to form the initial population. Depending on the problem, the generation of the initial population can be guided (seeded) to areas where optimal solutions are likely to be found.

Selection

The selection is a stochastic process usually planned so that the fitter solutions have a higher probability of being selected. This aims to ensure the convergence of the algorithm.

In particular, one can mention the roulette wheel selection method where the fitness level is used to associate a probability of selection to each candidate. If f_i is the fitness of the individual i, its probability to be selected is $p_i = \frac{f_i}{\sum_{j=0}^{n} f_j}$ where n is the number of individuals in the population.

Crossover

Once a pair of individuals has been selected, they will be crossed-over. Typically, two children are created from each set of parents. One method of crossover (one-point crossover) will be explained here but other approaches exists []. A random crossover point will be selected on both parents. Beyond that point, the data will be swapped with the information of the other parent as show in Example 2.4.

Example 2.4 (Crossover example). *Let us consider two individuals x and y of the population:*

$$\begin{array}{cccccccccc} x & = & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ y & = & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$

If the randomly-chosen crossover point is 2 then the obtained offspring is:

$$\begin{array}{lcccc|cccccc} x' & = & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ y' & = & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \end{array}$$ ■

Mutation

Mutation is a genetic operation used to ensure diversity in the generated populations. It changed one or more information in the chromosome of an individual. This alteration depends on how the solution is encoded. If it is a bit string, the most common operation is to apply a bit flip (see Example 2.5) while for float chromosomes, new values can be generated following user-defined rules (see detailed illustration in Section **??**).

Example 2.5 (Mutation example). *Let us consider one individual x' of the population:*

$$\begin{array}{lcccccccc} x' & = & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}$$ ■

If the randomly-chosen mutation point is 3 then x' becomes:

$$\begin{array}{lcccccccc} x' & = & 0 & 1 & \boxed{1} & 0 & 1 & 0 & 1 & 0 \end{array}$$

Termination

The generational process is repeated until a termination condition has been encountered. Common conditions are:

— a certain level of fitness reached;
— fixed number of generations reached;
— simulation elapsed time reached;
— no better results produced after several generations.

2.4.4 Multi-objective genetic algorithm: NSGA-II

While the original genetic algorithms have been developed for mono-objective purposes, they have also been extended to multi-objective optimization and among the most known, one can cite NSGA-II.

NSGA-II stands for Non-dominated Sorting Genetic Algorithm and has been developed by Deb [25] to provide a multi-objective version for genetic algorithms. It is an evolution of the original NSGA proposed in [26]. NSGA-II follows the same

steps as a classical GA and additionally implements techniques, particularly in the selection step, to take into account several objectives simultaneously.

NSGA-II selection

The selection is based on the Pareto dominance principle, particularly the Pareto rank which allows to sort all the solutions of a set following an extended Pareto principle and the crowding distance which estimates how dense the surrounding of a solution is.

Definition 2.6 (Pareto rank [25]). *From a given pool of solutions, the Pareto optimal ones are of rank 1. For the higher ranks the following process is repeated iteratively: to find the solutions of rank $i \geq 2$, the solutions of rank $i-1$ are removed and the Pareto solutions from this subset are of rank i.*

Definition 2.7 (Crowding distance [25]). *The crowding distance is a measure of the density of solutions surrounding a particular point in the population. It is computed by taking the average distance of the two points on either side of this point along each of the objectives (see. Algorithm 2).*

Algorithm 2: Crowding distance for the set of solutions A

$l = |A|$;
foreach *i* **do**
  set $A[i]_{distance} = 0$;
end
foreach *objective* m **do**
  A = sort(A,m);
  $A[1]_{distance} = A[l]_{distance} = \infty$;
  for $i = 2$ *to* $(l-1)$ **do**
    $A[i]_{distance} = A[i]_{distance} + (A[i+1] \cdot m - A[i-1] \cdot m)$
  end
end

The number of solutions per generation is fixed as constant. Between two solutions with different Pareto ranks, the lower rank will be preferred. Otherwise, if both solutions have the same Pareto rank then the one located in a lesser crowded region will be preferred.

2.4.5 Multi-criteria decision aid

Once the Pareto frontier is obtained or approximated, the compromise solutions can be found by establishing a preference model of the decision maker facing several

conflicting solutions. Those models can be classified into three broad categories [14, 27] whose methods will be detailed in Section 2.4.7:

1. *Aggregation methods*: numerical scores are calculated by aggregating the criteria to determine the level of preference for a solution. The most known aggregation methods are the Multi-Attribute Utility Theory (MAUT) [28] and the Analytic Hierarchy Process [29].
2. *Interactive methods*: it is a sequential process composed by alternating computation steps and dialogue with the decision maker. A first compromise is submitted to the decision maker who can accept or deny it. If the solution is denied, the DM can give extra information (e.g. releasing a constraint) about his preferences (dialogue) and a new solution can be calculated, so a new decision process begins. Otherwise, no better solution can be found and the process stops. Among the most known interactive methods, the STEP Method (STEM) [30] or the Satisficing Trade-Off Method (STOM) [31] can be cited.
3. *Outranking methods*: the solutions are compared pairwise which enables the possibility to identify the relationship between the solutions. This shows the preference for a solution in comparison to another one. PROMETHEE [32] and ELECTRE [33] are among the most known outranking methods.

Generally, the purpose of MCDA is to provide answers for three main problematic [34]:

1. *The choice problematic (P.α)*: the aid aims the selection of a small number of good solutions in such way that one or several compromise solutions can be chosen.

   Example 2.8. *In circuit design, the objective would be to choose the best compromise CPU in terms of performance and price.* ■

2. *The sorting problematic (P.β)*: the aid aims the assignment of each solution to a predefined (ordered) category.

   Example 2.9. *Depending on performance, price, radiation resistance, thermal operational range, electronic components can be sorted for commercial, industrial or military and spatial purposes.* ■

3. *The ranking problematic (P.γ)*: the aid aims the complete or partial preorder of all the solutions.

   Example 2.10. *With a preorder for CPUs based on an assessment of their performance, it is possible to associate a price to each processor depending on their ranking.* ■

2.4.6 Preference modelling definitions

Before introducing some important MCDA methods, let us first define some definitions about preference modelling in order to ease the understanding of the following sections.

When modelling the decision maker's preferences, three binary relations which result from the comparison of two alternatives a_i and $a_j \in \mathcal{A}$ are defined [14]:

$$\begin{cases} a_i P a_j & \text{if } a_i \text{ is prefered to } a_j \\ a_i I a_j & \text{if } a_i \text{ is indifferent to } a_j \\ a_i R a_j & \text{if } a_i \text{ is incomparable to } a_j \end{cases} \tag{2.6}$$

These relations translate situations of preference, indifference and incomparability and it can be assumed that they satisfy the following properties:

$$\forall a_i, a_j \in \mathcal{A} \begin{cases} a_i P a_j \Rightarrow a_i \neg P a_j : & : P \text{ is asymmetric} \\ a_i I a_i & : I \text{ is reflexive} \\ a_i I a_j \Rightarrow a_j I a_i & : I \text{ is symmetric} \\ a_i \neg R a_i & : R \text{ is irreflexive} \\ a_i R a_j \Rightarrow a_j R a_i & : R \text{ is symmetric} \end{cases} \tag{2.7}$$

Intuitively:

— aPb corresponds to the existence of clear and positive reasons that justify significant preference in favour of a

— aIb corresponds to the existence of clear and positive reasons that justify equivalence between the two alternatives

— aRb corresponds to an absence of clear and positive reasons that justify any of the two preceding relations

2.4.7 Some important multi-criteria methods

Multi-Attribute Utility Theory

Multi-Attribute Utility Theory (MAUT) has been introduced by Fishburn [35] and Keeney and Raiffa [36]. This method belongs to the family of aggregation methods that consist in substituting the initial multi-criteria problem

$$\min\{f_1(x), f_2(x), \ldots, f_m(x) | x \in \mathcal{A}\} \tag{2.8}$$

the following uni-criterion problem:

$$min\{U(x) | x \in \mathcal{A}\} \tag{2.9}$$

where $U(x)$ is called the utility function that aggregates all the criteria to a single criterion:

$$U(x) = U[f_1(x), f_2(x), \ldots, f_m(x)] \tag{2.10}$$

One of the most used utility function is the weighted sum:

$$U(x) = \sum_{j=1}^{m} w_j f_j(x) \tag{2.11}$$

where w_j is the weight associated to the criterion j.

With this utility function, it is then possible to compute an aggregated score for each solutions and rank them in order to choose among the best ones.

MAUT has been applied in numerous cases and developments have been provided to axiomatize this method and justify its use [28].

Analytical Hierarchy Process (AHP)

Analytical Hierarchy Process (AHP) has been developed by Saaty [29]. This multi-criteria method is based on mathematics and psychology and allows to face structurally complex choices by decomposing the problem in several sub-problems that can be analysed independently and are easier to understand. Similarly to PROMETHEE and ELECTRE, AHP proceeds by making pairwise comparisons of the alternatives, but on basis of a ordinal scale from 1 to 9. Indeed, one of the distinctive features of this methods is to build a matrix by asking the decision maker to compare all pairs of alternatives and criteria. Therefore, the input for AHP is not an evaluation table but the DM's preference matrix. The normalized right-hand eigenvector of this matrix is then used to compute the score associated to each alternative and the weight associated to each criterion.

In order to illustrate AHP, we will give more details on a particular case where only the criteria are compared. The decision maker will make pairwise comparisons and give an ordinal scale of preference for the criteria. The following matrix can be obtained:

$$A = \begin{pmatrix} 1 & a_{12} & \dots & a_{1j} & \dots & a_{1m} \\ \frac{1}{a_{12}} & 1 & \dots & a2j & \dots & a_{2m} \\ \vdots & & & & & \\ \frac{1}{a_{1j}} & \frac{1}{a_{2j}} & \dots & a_{ij} & \dots & a_{im} \\ \vdots & & & & & \\ \frac{1}{a_{1m}} & \frac{1}{a_{2m}} & \dots & a_{im} & \dots & 1 \end{pmatrix} \tag{2.12}$$

where a_{ij} is expresses the relative importance of the criterion i over the criterion j.

From this matrix, AHP uses a method based on eigenvector to extract the related weights of each criterion that can be used, for instance, as input data for MAUT in a weighted sum.

A comparison matrix is said to be consistent if $a_{ij}a_{jk} = a_{ik} \forall i, j, k$. However, consistency cannot always be reached and AHP's developers have defined a Consis-

tency Index (CI):

$$CI = \frac{\lambda_{max} - m}{m - 1} \tag{2.13}$$

where λ_{max} is the largest eigenvalue of the matrix and m is the matrix size.

This Consistency Index is then compared to Random (consistency) Index (RI) which are considered to be appropriate CIs. These RIs are obtained by randomly generating matrices and taking the average CI values.

A Consistency Ratio (CR) then is defined:

$$CR = \frac{CI}{RI} \tag{2.14}$$

If the value of the Consistency Ratio is lower or equal to 10%, the inconsistency is considered to be acceptable. Otherwise, the decision maker has to revise judgements.

STEP Method (STEM)

The STEP Method has been proposed by Benayoun [30]. STEM is an interactive and iterative exploration procedure that aims to reach the best compromise according the decision maker after a certain number of cycles. Each cycle is composed of a calculation phase and a decision-making phase (discussion with the decision maker):

1. An efficient compromise solution is determined.
2. This solution is submitted to the decision maker. Three cases can then happen:
   (a) The decision maker is satisfied and the procedure ends;
   (b) The decision maker wants to simultaneously improve all the evaluations. This is impossible since the proposed solution is efficient. The procedure ends and cannot help the decision maker.
   (c) The decision maker identifies a particular criterion on which a concession can be made in order to improve other criteria. A new efficient solution can then be determined.
   (d) This new solution is submitted. Go to step 2a.

Satisficing Trade-Off Method (STOM)

The STOM method has been proposed by Nakayama [31]. Similarly to STEM, it relies on a discussion with the decision maker but is based on the setting of an ideal point defined as follows:

Definition 2.11 (Ideal point). *The ideal point* $f^* = (f_1^*, f_2^*, \ldots, f_m^*)$ *is defined such that* $f_i^* = \min\{f_i(x), \forall i = 1, 2, \ldots, m, \forall x \in \mathcal{A}\}$.

The ideal point possesses as coordinates the best values that can be achieved for each criterion separately.

STOM can be summarized in four steps:

1. The first step is to set the ideal point.
2. Then the aspiration level for each criterion is asked to the decision maker; this is the reference point for each criterion of the decision maker.
3. A Pareto solution nearest to the aspiration level is determined.
4. This solution is submitted to the decision maker. If it is satisfactory, the procedure ends. Otherwise, the decision maker is asked to trade off to define another aspiration level. Go to step 3.

The PROMETHEE methods

PROMETHEE (Preference Ranking Organisation METHod for Enrichment Evaluations) has been initiated by Brans [32] and developed with Mareschal [37] and Vincke [38]. In this section, we will only describe the basics of PROMETHEE. More details can be found in [39].

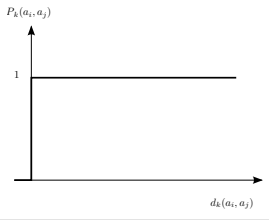
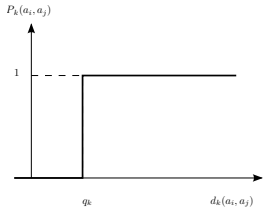
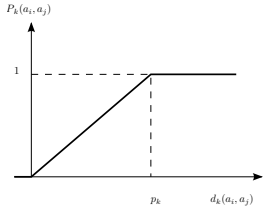
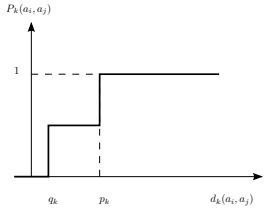
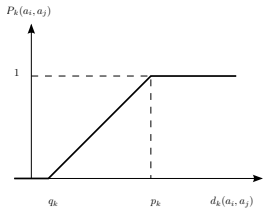
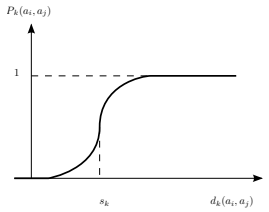
The PROMETHEE methods are based on the three following steps:

— Enriching the preference structure: a preference function is introduced.
— Enriching the dominance relation: a valued outranking relation is determined.
— Decision aid: the valued outranking relations are exploited.

1. *Preference function*
   Since the dominance relation is really poor (binary relation), a preference function $P_k(a_i, a_j)$ will be introduce to enrich it. This function gives the preference degree of an alternative a_i over an alternative a_j with respect to the function $d_k(a_i, a_j) = f_k(a_i) - f_k(a_j)$ which is the difference between the evaluation of a_i and a_j for the criterion k, assuming a non decreasing function.
   Consequently, it is therefore possible to define several types of preference functions based on preference (P) or indifference (Q) thresholds, as shown in Table 2.2. Below the indifference threshold, the decision maker will consider having no preference while above the preference threshold, the decision maker will have no more difference in its preference.
2. *Valued outranking relation*
   Multi-criteria preference index
   The multi-criteria preference index is defined as follows:

$$\pi(a_i, a_j) = \sum_{k=1}^{m} P_k(a_i, a_j).w_k, \forall i \neq j \text{ with } \sum_{k=1}^{k} w_k = 1 \tag{2.15}$$

Table 2.2: Preference functions

Usual		Strict preference
U-shape		Q: indifference threshold
V-shape		P: preference threshold
Level		Q: indifference threshold P: preference threshold
Linear		Q: indifference threshold P: preference threshold
Gaussian		S: preference threshold

where $w_k > 0, k = 1, 2, ..., m$ are the weights on each criterion. $\pi(a_i, a_j)$ represents a measure of the preference of a_i over a_j on all the criteria.

Let us note the following properties of the preference index:

$$\pi(a_i, a_i) = 0 \tag{2.16}$$
$$0 \leq \pi(a_i, a_j) \tag{2.17}$$
$$\pi(a_i, a_j) + \pi(a_j, a_i) \leq 1 \tag{2.18}$$

Outranking flow

An "outranking flow"is then defined on the basis of the preference index. That allows to compare alternatives with each others. Three types of flow are formulated:

— The positive outranking flow: $\phi^+ = \frac{1}{n-1} \sum_{j \neq i} \pi(a_i, a_j)$. This flow expresses how a_i outranks all the other alternatives.

— The negative outranking flow:$\phi^- = \frac{1}{n-1} \sum_{j \neq i} \pi(a_j, a_i)$. This flow expresses how a_i is outranked by all the other alternatives.

— The net flow: $\phi(a) = \phi^+(a_i) - \phi^-(a_i)$. This flow expresses the balance between the positive and negative flows of a_i

Let us note the following properties for these flows:

$$\phi^+, \phi^- \in [0; 1] \tag{2.19}$$
$$\phi \in [-1; 1] \tag{2.20}$$

Based on these flows, the PROMETHEE methods will establish an outranking.

3. *PROMETHEE I*

The positive and negative flows allow to sort the alternatives of A. Let (S^+, I^+) and (S^-, I^-) be the two complete pre-orders obtained from these flows:

$$\begin{cases} a_i S^+ a_j \Leftrightarrow \phi^+(a_i) > \phi^+(a_j) \\ a_i I^+ a_j \Leftrightarrow \phi^+(a_i) = \phi^+(a_j) \end{cases} \tag{2.21}$$

This means that the higher the positive flow is, the better the alternative.

$$\begin{cases} a_i S^- a_j \Leftrightarrow \phi^-(a_i) < \phi^-(a_j) \\ a_i I^- a_j \Leftrightarrow \phi^-(a_i) = \phi^-(a_j) \end{cases} \tag{2.22}$$

This means that the lower the negative flow is, the better the alternative.

PROMETHEE I establishes a partial ranking by taking the intersection of these two pre-orders:

$$\begin{cases} a_i P^{(1)} a_j \Leftrightarrow \begin{cases} a_i S^+ a_j \text{ and } a_i S^- a_j \\ a_i S^+ a_j \text{ and } a_i I^- a_j \\ a_i I^+ a_j \text{ and } a_i S^- a_j \end{cases} \\ a_i I^{(1)} a_j \Leftrightarrow a_i I^+ a_j \text{ and } a_i I^- a_j \\ a_i R^{(1)} a_j \text{ otherwise} \end{cases} \tag{2.23}$$

where $(P^{(1)}, I^{(1)}, R^{(1)})$ represent respectively the preference, the indifference and the incomparability in PROMETHEE I.

— $a_i P^{(1)} a_j$ ("a_i is prefered to a_j"): a_i is simultaneously better and less worse than a_j.

— $a_i I^{(1)} a_j$ ("a_i and a_j are indifferent"): a_i is neither better nor worse than a_j.

— $a_i R^{(1)} a_j$ ("a_i and a_j are incomparable"): a_i is better than a_j on some criteria while a_j is better than a_i on other criteria.

4. *PROMETHEE II*

   In order to obtain a complete ranking, the net flow will be considered:

$$\begin{cases} a_i P^{(2)} a_j \Leftrightarrow \phi(a_i) > \phi(a_j) \\ a_i I^{(2)} a_j \Leftrightarrow \phi(a_i) = \phi(a_j) \end{cases} \tag{2.24}$$

   where $P^{(2)}$ et $I^{(2)}$ represent respectively the preference and the indifference in PROMETHEE II. This means that the higher the net flow is, the better the alternative.

   Let us note that, unlike PROMETHEE I, PROMETHEE II does not give place to incomparability and a complete ranking can directly be obtained.

5. *The GAIA plane*

   While it is impossible to have a visual representation of the solution space when there are more than three criteria, the GAIA (Geometrical Analysis for Interactive Assistance) plane can give a visualization even if there are more than three criteria, by means of the principal component analysis (PCA) of the net flows on the decision maker's preferences for each criterion.

   The PCA allows a projection of the alternatives on a plane that minimizes the loss of information induced by this projection.

   This plan allows to have a visual descriptive analysis with several criteria. It can highlight the conflict between criteria and show the profiles of the alternatives. This will help to identify the potential compromise solutions.

The ELECTRE methods

ELECTRE (*ELimination Et Choix Traduisant la REalité*, or ELimination and Choice Expressing REality) has been developed by Roy [33]. In this section, we will only described the basics of ELECTRE. More details can be found in [40].

1. *ELECTRE I*

   ELECTRE I is a method linked to the $P.\alpha$ problematic that aims to obtain a subset N of alternatives such that all the solutions that do not belong to this set is outranked by at least one alternative of N and the solutions of N do not outrank each other. N is therefore not the set of good alternatives but rather the set where the best compromise can certainly be found.

   The outranking relation is obtained by establishing a weight w_k for each criterion. A concordance index is the associated to each pair (a_i, a_j) of alternatives:

   $$c(a_i, a_j) = \frac{1}{W} \sum_{j: f_k(a_i) \leq f_k(a_j)} w_k, \text{ where } W = \sum_{k=1}^{m} w_k, w_k > 0 \tag{2.25}$$

   The concordance index represents a measure of the arguments favourable to the statement "a_i *outranks* a_j".

   A discordance index can also be defined:

   $$d(a_i, a_j) = \begin{cases} 0 & \text{if } f_k(a_i) \geq f_k(a_j), \forall k \\ \frac{1}{\delta} \max_k [f_k(a_j) - f_k(a_i)] & \text{otherwise} \end{cases} \tag{2.26}$$

   The discordance index is therefore higher if the preference of a_j over a_i is strong on at least one criterion.

   Then concordance $\hat{c}$ and discordance $\hat{d}$ thresholds are defined alongside the outranking relation S:

   $$\forall i \neq j, a_i S a_j \text{ iff } \begin{cases} c(a_i, a_j) \geq \hat{c} \\ d(a_i, a_j) \leq \hat{d} \end{cases} \tag{2.27}$$

   From this definition, a subset N of alternatives is established such that:

   $$\begin{cases} \forall a_j \in A \setminus N, \exists a_i \in N : a_i S a_j \\ \forall a_i, a_j \in N, a_i \overline{S} a_j \end{cases} \tag{2.28}$$

   A subset N of alternatives is established such that all the alternatives that do not belong to this set is outranked by at least one alternative of N and the alternatives of N are incomparable. The decision process will therefore take place within the set N.

2. *ELECTRE II*

   This method aims to rank the alternatives. The outranking relation is defined by fixing two concordance thresholds $\hat{c}_1$ and $\hat{c}_2$ such that $\hat{c}_1 > \hat{c}_2$ and by building a strong outranking relation S^F and a weak outranking relation S^f based on these two thresholds:

$$a_i S^F a_j \text{ iff } \begin{cases} c(a_i, a_j) \geq \hat{c}_1 \\ \sum_{,:f_k(a_i)>f_k(a_j)} w_k > \sum_{k:g_k(a)<g_k(b)} w_k \\ (f_k(a_i), f_k(a_j)) \notin D_k, \forall k \end{cases} \tag{2.29}$$

$$a_i S^f a_j \text{ iff } \begin{cases} c(a_i, a_j) \geq \hat{c}_2 \\ \sum_{k:f_k(a_i)>f_k(a_j)} w_k > \sum_{k:f_k(a_i)<f_k(a_j)} w_k \\ (f_k(a_i), f_k(a_j)) \notin D_k, \forall k \end{cases} \tag{2.30}$$

   The discordance can also induce two levels of relations by building two sets of discordance for each criterion.

   In order to obtain the ranking, a set is determined from S^F. This set B contains the alternatives that are not strongly outranked by any others. From B and S^f, the set A^1 of alternatives that are not weakly outranked by any alternatives of B is determined. The set A^1 constitutes the best alternatives class. A^1 is then removed and the process is repeated to find A^2 and so on until a complete pre-order is obtained.

   Let us note that a second complete pre-order can be obtained by applying the process first with the less good alternatives class and then the best ones.

3. *ELECTRE III*

   This method takes into account the indifference and preference thresholds. It is based on a valued outranking relation that is less sensible to data and parameters variabilities.

   In ELECTRE III, an outranking degree $S(a_i, a_j)$ associated to each pair (a_i, a_j) of alternatives is defined. It can be understood as an "degree of credibility of outranking"of a_i over a_j.

   A weight w_k is associated to each criterion and for each pair (a_i, a_j) of alternatives the concordance index is computed as follows:

$$c(a_i, a_j) = \frac{1}{W} \sum_{k=1}^{m} w_k c_k(a_i, a_j), \text{ where } W = \sum_{k=1}^{m} w_k \tag{2.31}$$

with

$$c_k(a_i, a_j) = \begin{cases} 1 & \text{if } f_k(a_i) + q_k(f_k(a_i)) \geq f_k(a_j) \\ 0 & \text{if } f_k(a_i) + p_k(f_k(a_i)) \leq f_k(a_j) \\ \text{linear} & \text{if } \begin{array}{r} f_k(a_i) + q_k(f_k(a_i)) \leq f_k(a_j) \\ \leq f_k(a_i) + p_k(f_k(a_i)) \end{array} \end{cases} \tag{2.32}$$

where q_k et p_k represent respectively the indifference and preference thresholds.

The definition of discordance is then enriched by the introduction of a veto threshold $v_k(f_k(a_i))$ for each criterion k such that any credibility for the outranking of a_j by a_i is refused if $f_k(a_j) \geq f_k(a_i) + v_k(f_k(a_i))$.

A discordance index is then defined:

$$D_k(a_i, a_j) = \begin{cases} 0 & \text{if } f_k(a_j) \leq f_k(a_i) + p_k(f_k(a_i)) \\ 1 & \text{if } f_k(a_j) \geq f_k(a_i) + v_k(f_k(a_i)) \\ \text{linear} & \text{if } \begin{array}{r} f_k(a_i) + p_k(f_k(a_i)) \leq f_k(a_j) \\ \leq f_k(a_i) + v_k(f_k(a_i)) \end{array} \end{cases} \tag{2.33}$$

The degree of outranking is finally defined:

$$S(a_i, a_j) = \begin{cases} c(a_i, a_j) & \text{if } \begin{array}{r} D_k(a_i, a_j) \leq c(a_i, a_j) \\ \forall k \end{array} \\ c(a_i, a_j) \prod_{k \in \mathcal{F}(a_i, a_j)} \frac{1 - D_k(a_i, a_j)}{1 - c(a_i, a_j)} & \text{otherwise} \end{cases} \tag{2.34}$$

where $\mathcal{F}(a_i, a_j)$ is the set of criteria for which $D_k(a_i, a_j) > c(a_i, a_j)$. The degree of outranking is thus equal to the concordance index when no criterion is discordant, otherwise the concordance index is decreased proportionally depending on the importance of the discordances.

A value $\lambda = \max_{a_i, a_j \in \mathcal{A}, i \neq j} S(a_i, a_j)$ is determined and only the outranking degree that have a value greater or equal to $\lambda - s(\lambda)$, where $s(\lambda)$ is a threshold to be determined, are considered. A ranking can then be determined from a qualification index $Q(a)$ for each alternative a that represents the difference between the number of outranked alternatives by a and the number of alternatives that outrank a. The set of actions having the largest qualification will be called the first distillate D_1.

If D_1 contains only one alternative, the previous procedure is repeated with $A \setminus D_1$. Otherwise the same procedure is applied for D_1 and if the obtained distillate D_2 contains only one alternative, the procedure is repeated with $D_1 \setminus D_2$. Otherwise, it is applied for D_2, and so on until D_1 is completely

used, before starting with $A \setminus D_1$. This procedure produces a first complete preorder.

A second complete preorder can be obtained by applying the opposite procedure where the alternatives with the smallest qualification are first used.

2.5 Conclusion

TODO

3

Problem definition and 3D-stacked integrated circuit model

In the previous chapters, we have presented a review of the literature about the field of microelectronics design. We have highlighted some limitations of the current tools that already occur for current ICs. In this chapter, we will define the problem we tackle and show how we model it in order to propose improvements to design flows.

3.1 Problem definition

As stated in Chapter **??**, the limitations of the current design flows can be summarized in three points:

— Limitation of the design space exploration
— Unicriterion optimization
— No 3D-SIC specific tools

In order to address these limitations, we propose in this thesis a methodology based on multi-objective/criteria tools and taking into account 3D-SIC specificities for an exploration of the design space.

While this methodology could be applied at different levels in a design flow, we have focused our development in the logical design step and the virtual prototyping flow, more specifically the floorplanning with performance assessments.

3.1.1 Design an IC

In order to meet the specifications, a design has to first make choice at a physical level:

— Targeted architecture, e.g. ASIC, FPGA
— Number of functional units
— Number of memories and their size
— The general layout
— ...

Since the 3D-SICs are based on conventional circuits, the options and degrees of freedom coming from 2D-ICs are still present:

— Process technology, e.g. 180 nm to 22 nm CMOS
— Memories technology, e.g. SRAM, DRAM, FLASH
— Communication infrastructure, e.g. bus, Network-on-Chip
— ...

In addition to those options and degrees of freedom coming from 2D-ICs, there are also numerous 3D-SIC's parameters:

— Number of tiers to use
— Place and route of the functional units between the tiers
— Technology to use per tiers (heterogeneity)
— Interconnection and geometry between tiers
— 3D-SIC integration technology
— 3D-SIC assembly technology
— ...

The above mentioned parameters illustrate the numerous possibilities for designing a circuit and how the design space for 2D-ICs becomes much bigger when considering 3D-SICs. The main issue is therefore to choose the most efficient combination among all those options. This can thus be compared to a combinatorial optimization problem which makes the use of metaheuristics more than justified as they are commonly-used tools for such problems. Also, given the multi-criteria nature of designing 3D-SIC, this optimization will have to take into account all the criteria simultaneously. In the next section will define the criteria that a designer can consider.

3.2 Model and criteria definition

Typically, the criteria that have to be optimized simultaneously can be the performance, the power consumption, the cost, the package size, the heat dissipation, etc. In this model, we will define six criteria which are among the most important parameters while designing a circuit [41]. Among the six presented criteria, the first five are implemented and the latter one is discussed:

1. *The interconnection global length*: this parameter can reflect the global performance of a system. The objective is to minimize it in order to have, for instance, a short delay and low power consumption. It will be calculated using the Manhattan distance:

$$d_{i,j} = |x_i - x_j| + |y_i - y_j| \tag{3.1}$$

where (x_n, y_n) is the geometrical coordinates of the n^{th} block. As a first approximation, the center point of each block will be selected as reference coordinate. Also, since it is more interesting to place close to each other two blocks that require a large bandwidth (BW) to communicate, we will balance the values as follows:

$$d'_{i,j} = \frac{d_{i,j}}{BW_{i,j}} \tag{3.2}$$

So the global interconnection length D will be the sum of $d'_{i,j}$ for all communicating blocks:

$$D = \sum d'_{i,j} \tag{3.3}$$

2. *The cost*: an economical factor is obviously an important criteria for a design. This criteria has been estimated with the aid of an expert in 3D-SIC manufacturing. While a circuit can be more efficient with many layers, it will also be more expensive. This criteria has to be minimized. Due to the confidential nature of the cost of a 3D-SIC, we will consider a simplified model where the cost is proportional to the area and increasing exponentially with the number of tiers:

$$cost = a(tech).S + b(tech)^{layer\ number} \tag{3.4}$$

where $a(tech)$ and $b(tech)$ are coefficient depending on the technology assigned. Let us note that this criterion includes both discrete and continuous variables.

3. *The package volume*: this can be an important criteria when designing embedded circuits. The package volume is calculated as follows:

$$volume = largest\ layer\ size\ *\ stack\ thickness \tag{3.5}$$

A large appoximation of 200 μm will be made for the thickness of one tiers. Let us note that this criterion includes both discrete and continuous variables.

4. *The clock tree position*: in this model, we consider a synchronous system so the objective is to minimize the distance between each block and the clock tree in order to have a high frequency. We choose arbitrarily to approximate the reference point as a fixed point located at the upper left corner of the middle tier of the 3D-SIC.

5. *The thermal dissipation*: thermal dissipation is one of the major issues when designing 3D-SICs. It can be more appropriate to place two blocks underneath each other in successive tiers but a high heat dissipation may happen in intensive computational process. This criterion is a research topic on its own [42, 43]. Here we will use a simplified evaluation model with finite elements. This model will consider that the dissipated power, intra- or inter-tiers, is inversely proportional to the distance to the heat source:

$$P_{diss} = \sum_i \frac{1}{R_{th,i} r} \tag{3.6}$$

where r is the distance to the heat source and $R_{th,i}$ the thermal resistance depending on whether the dissipation is intra- or inter- tiers. This criteria is still on early development stage and we can generate thermal maps of a floorplan as shown in Fig. 3.1 but this is currently based on finite elements [44] which require quite a long computational time even for a simplified thermal model. This criterion in its current development stage is difficult to integrate to the exploration process, due to the computation time of finite elements methods. In the current work, we will simply compute the peak power of a circuit which can be done more quickly.

6. *The power consumption*: the objective is to minimize the power consumption which can be a crucial criteria for embedded systems. Generally, the power consumption can be defined as the sum of a static (P_{stat}) (given data from the component datasheet) and a dynamic (P_{dyn}) consumption:

$$P = P_{stat} + P_{dyn} \tag{3.7}$$

$$P_{stat} = [given\ data] \tag{3.8}$$

$$P_{dyn} = \alpha.C.V_{dd}^2.f.[tech] \tag{3.9}$$

where α represents the toggle rate, C the capacitance, V_{dd} the voltage, f the frequency and $[tech]$ a rectification factor due to the technology assigned.

At first, we will focus on the three first criteria in order to be able to have a visualization of the design space. We will also arbitrarily introduce some limitations in term of degrees of freedom to analyse what happens if we release a degree of freedom. This will be done while considering the three same criteria, in order to keep a visualization and show how the flexibility of MOO will improve the information and the results. Then we will analyse our methodology with the five first criteria that have been presented. As stated, the sixth criterion is currently unused. Actually, it has been simulated and tested but we need data from the manufacturers which are not easily available.

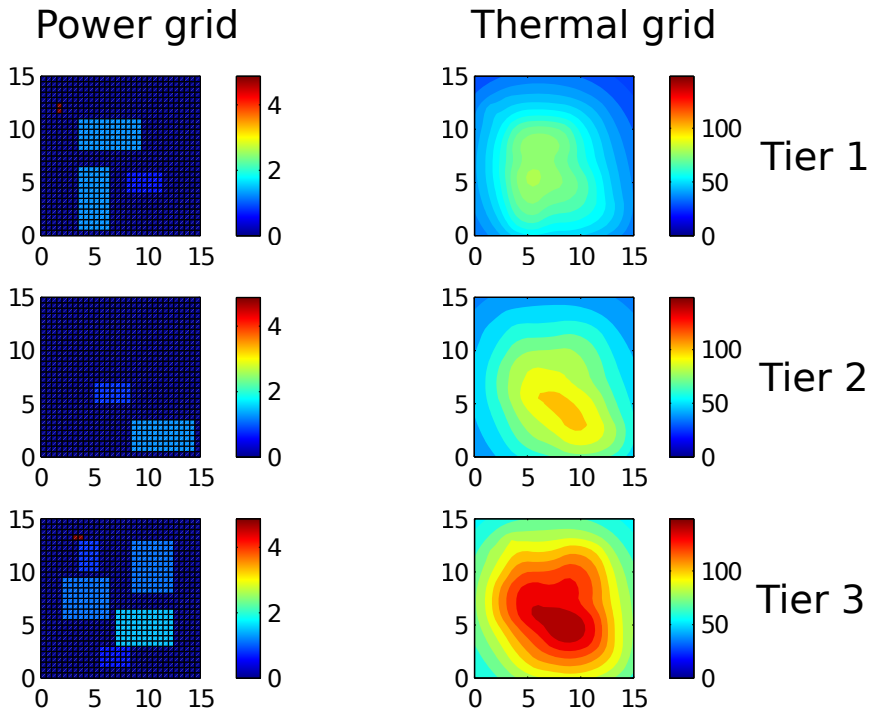


Figure 3.1: Power grid and thermal map of a floorplan (3 tiers)

3.3 Design methodology

In summary, the problem we are facing is to place several blocks that have to be assigned in many tiers while considering multiple conflicting criteria. Now that our model has been defined, we will present a proposition of a new design methodology based on multi-objective optimization.

As explained previously, designing ICs implies numerous choices. At the moment, with this growing complexity, the current design flows can show their limits. For instance, most of the time, the designers will be likely to freeze a certain amount of choices on basis of their experience, and then begin the optimization process with the remaining parameters. This will therefore limit the exploration of the design space and good solutions may be ignored. In addition, the fixed choices can be questionable since they are based on the designer's experience though they could also be based on more objective facts.

In order to enable an efficient design space exploration, we propose a method in four steps based on MCDA which is illustrated in Fig. 3.2. The implementation will be briefly presented in the next section.

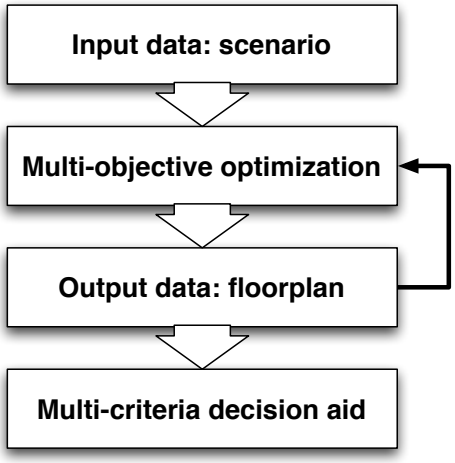


Figure 3.2: MCDA-based design methodology

For the problem we consider, the input data will contain the information about the scenario:

— Type and number of blocks: computational units, memories, etc.
— Size of the blocks: inherent to the block.
— Minimum aspect ratio: we consider a degree of freedom where a block can have its dimensions varying within an aspect ratio range. This means that a block does not have to be square, as shown in Fig. 3.3. This parameter can influence the delay in a block.
— Size variability of a block: we add this degree of freedom considering that the specified size of a block can be fixed by the designer but this fixed size can restrict the design space exploration. The variability of a block's size can have effect on the performance and the global footprint.

In addition, the bandwidth requirements are needed as they will indicate which are the important interconnections and prevent two blocks that require a large bandwidth from being too far from each other. The available manufacturing technologies are also useful to enable the design of heterogeneous systems.

The combination of all the parameters described in the model are the possible alternatives for a 3D-SIC design and will provide output data after design space exploration. For a floorplanning problem the required output data are generally the geometrical layout of the circuit [41]:

— The geometrical coordinates for each block and the assigned layer.
— The size of each block (if it can vary from the specified size).
— The aspect ratio for each block.
— The technology assigned to each tier: this will reduce the size of each block.

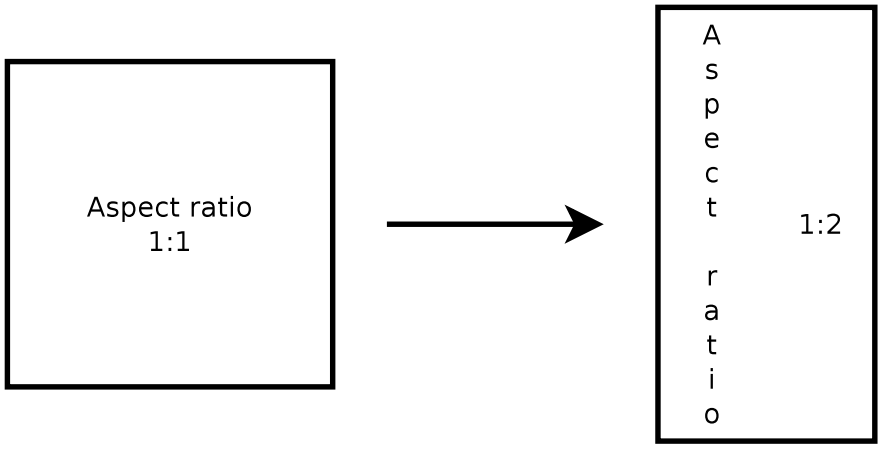


Figure 3.3: Example of aspect ratio degree of freedom

The size of a block will define the number of transistors inside using a given technology, for example 180 nm. For a constant number of transistors, if the block is manufactured with a smaller technology, let us say 45 nm, then its size will be divided by a $(180/45)^2$ factor, as shown in Fig. 3.4. Please note, that this factor is an approximation which is not always met with real physical design.

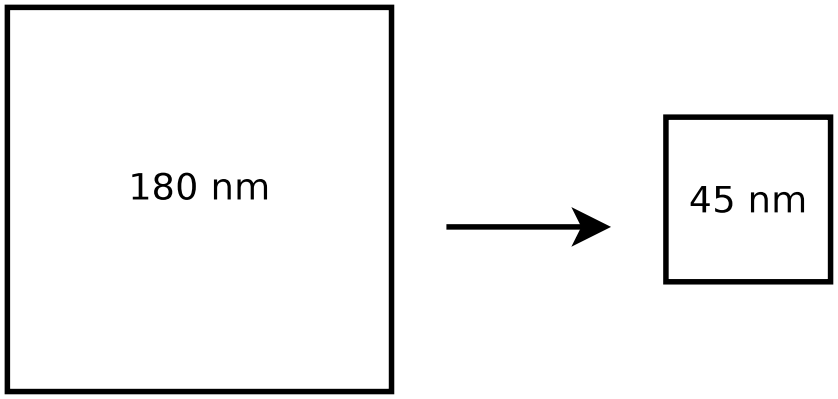


Figure 3.4: The use of different manufacturing technologies results in a size variation

Table 3.1: Scenario input matrix example

Component	ID	Size (90 nm)	Min aspect ratio	Size variability
ADRES 1~6	1~6	18.6 mm^2	0.5	0.1
FIFO	7	0.54 mm^2	0.5	0.1
L2D1-2	8-9	6.74 mm^2	0.5	0.1
L2Is1-1	10-11	6.62 mm^2	0.5	0.1
EMIF	12	0.66 mm^2	0.5	0.1
ARM	13	0.89 mm^2	0.5	0.1

ID: Component identification number

3.4 Case study and implementation

In this section, we will briefly explain the implementation of our design method and show some experimental results based on this first approach. We consider a case study based on the 3MF MPSoC platform developed at IMEC [45].

3.4.1 Description of the case study and modeling

The 3MF MPSoC is made of 13 blocks as shown on Fig. 3.5:
— 6 ADRES processors ([46])
— 2 data memories (L2D#)
— 2 instruction memories (L2Is#)
— 1 external memory interface (EMIF)
— 1 input/output processor (FIFO)
— 1 ARM processor (ARM)

Details about the area required for each component is given in Table 3.1 for a 90 nm technology. This table is also the input matrix required to specify the scenario.

The 3MF MPSoC can be configured for three use cases which have specific bandwidth requirements. For the following results, we will base our simulation on the "data split scenario" configuration which possesses the communication specifications shown in Table 3.2. This information is implemented, as shown in Table 3.3, in an input matrix which is built by specifying the communication structure: the first column will contain the ID of the source block and each next pairs of columns will contain the ID of the target blocks and the bandwidth required.

The input data are thus shown in Tables 3.1 and 3.3. The available technologies are also needed to take advantage of the heterogeneity. An example matrix for this input data is given in Table 3.4. Since no information is given about the ARM unit bandwidth usage, we will simplify our problem and not include it in the implementation. We consider therefore 12 blocks to assign.

Table 3.2: "Data split scenario" bandwidth requirements

Source	Target	Bandwidth (MB/s)
FIFO	EMIF	39.6
EMIF	ADRESi	6.6
L2D1	ADRESi	26.4
L2D2	L2D1	52.7
ADRESi	FIFO	1.2
ADRESi	L2D2	6.6
ADRESj	L2Is1	300
ADRESk	L2Is2	300

Index: $i, j, k \in \mathbb{N}^+$;
$1 \leq i \leq 6; 1 \leq j \leq 3; 4 \leq k \leq 6$

Table 3.3: Bandwidth input matrix

S	T	B	T	B	T	B	T	B	T	B	T	B
1	7	1.2	9	6.6	10	300	0	0	0	0	0	0
2	7	1.2	9	6.6	10	300	0	0	0	0	0	0
3	7	1.2	9	6.6	10	300	0	0	0	0	0	0
4	7	1.2	9	6.6	11	300	0	0	0	0	0	0
5	7	1.2	9	6.6	11	300	0	0	0	0	0	0
6	7	1.2	9	6.6	11	300	0	0	0	0	0	0
7	12	39.6	0	0	0	0	0	0	0	0	0	0
8	1	26.4	2	26.4	3	26.4	4	26.4	5	26.4	6	26.4
9	8	52.7	0	0	0	0	0	0	0	0	0	0
12	1	6.6	2	6.6	3	6.6	4	6.6	5	6.6	6	6.6

S: source block ID
(T, B): target block ID and required bandwidth

Table 3.4: Available technologies input matrix example

Technology (nm)				
90	60	45	32	22

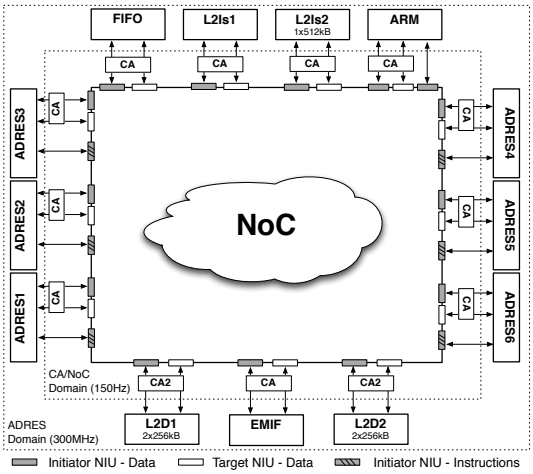


Figure 3.5: Architecture of the 3MF MPSoC platform [45]

In summary, the problem we consider is to place 12 blocks while taking into account several criteria (5). We will also consider a scenario where the blocks can be placed on 1 up to 5 tiers. The input data will be processed to generate floorplans. Those output data will be encoded using the matrix model following the example shown in Table 3.5. They will be generated through a multi-objective optimization.

As explained earlier, we will use a metaheuristic to approximate the Pareto optimal frontier. For that purpose, we choose to use NSGA-II [25]. The algorithm was run from a sample of 10 000 generated solutions from 1 up to 5 tiers. This size of random solutions is chosen arbitrarily since it is actually quite difficult to estimate the size of solution space, due to the heterogeneous nature of the criteria. As stated in Section 2, even for a really simplified problem, the solution space is huge (more than 10^{18}) so, taking 10 000, 100 000 or 1 000 000 randomly-generated solutions does not really imply any difference. Also, taking too few solutions (e.g. 100) is not interesting since we have empirically observed that our algorithm will take a longer time to begin to converge. 10 000 randomly-generated solutions seems to us a good compromise of time and workable solutions.

In the following section, we will present some details about the implementation of the NSGA-II algorithm.

3.4.2 Implementation of the exploration algorithm: NSGA-II

As shown in Table 3.5, we choose to encode our data in real or integer values, so that they can be used directly by design tools:

— The component identification number (ID) is a fixed integer value linked to

the component.
— The assigned layer (L) is a discrete value ranging from 1 to 5 in the case study.
— The geometrical coordinates (X,Y) are real values that depends on the dimension of the circuits and the aspect ratio of a block, so that the component cannot be placed outside the chip.
— The size (S) is a fixed real value linked to the component.
— The aspect ratio (AR) is a real value ranging from AR_{min} to $1/AR_{min}$ where AR_{min} is given as a specification as explained in Section 4.
— The length in X and Y axis (LX, LY) are real values computed from the size and the aspect ratio.
— The assigned technology per layer is a discrete value taking one of the specified technology (see Table 3.4).

This matrix will be our full chromosome for the NSGA-II algorithm.

We implemented our design space exploration following the steps of the NSGA-II which can be summarized by the diagram shown in Fig. 3.6.

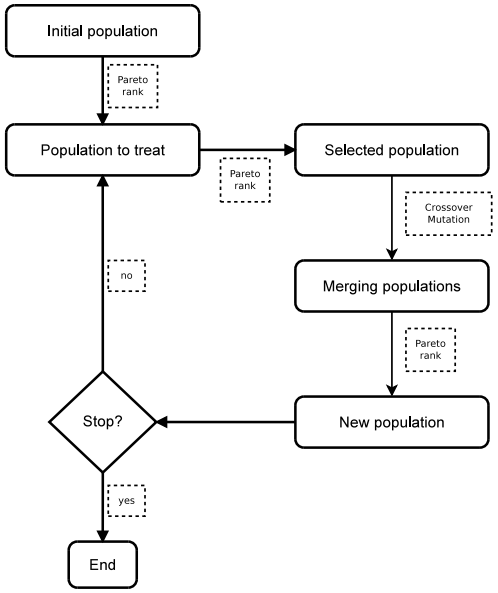


Figure 3.6: General NSGA-II steps

Initialization (the initial population)

We will work with a minimum size of population, namely 50, which is a common value in GAs [47]. The initial population will be a set of at least 50 solutions with the best Pareto ranks from a randomly-generated set of 10 000 solutions. The generated set places the blocks randomly (using a uniform distribution) and does not allow overlapping between the blocks. We could of course use a greedy algorithm as well as a more advanced method such as GRASP (Greedy Randomized Adaptive Search Procedure) [48]. This can be done as future work for comparison purposes.

Of course, having at least 50 Pareto solutions does not always happen. Actually, the selection is based on the Pareto rank so it does not include only the Pareto solutions (rank 1), but also the solutions with lower ranks until there are enough solutions.

Selection/crossover

For the selection step, two solutions will be allowed to make a crossover depending on a roulette wheel where the probability is proportional to the normalized Euclidean distance between the solutions ordered by their Pareto rank in the objective space. The normalization is done as follows :

$$\frac{g_i(a_j)}{\max\limits_{a_j \in A} g_i(a_j)} \tag{3.10}$$

where A is the set of alternatives with $a_j \in A$ and $g_i(a_j)$ is the evaluation of the alternative a_j on the criterion i.

The probability for two solutions to do a crossover will vary linearly with the Euclidean distance between them, as shown in Fig. 3.7. If two solutions are close to each other, they will have more chance to reproduce than if they are distant. This is to ensure the intensification properties of our algorithm. Therefore, we will have to specify a lower bound ($P_{c,min}$) and an upper bound ($P_{c,max}$) for the crossover probability. $P_{c,min}$ is set for the solutions which are the furthest to each other while $P_{c,max}$ is set for those which are the closest. In between, the probability will vary linearly inside these bounds.

These values will be fixed as $[P_{c,min} = 0.6; P_{c,max} = 1.0]$ since these seem to be common values [47].

Crossover

Let us now see how does the crossover occur. First, let us remark that it does not have limitations for the exploration process since the information contained in the matrix spans the whole circuit.

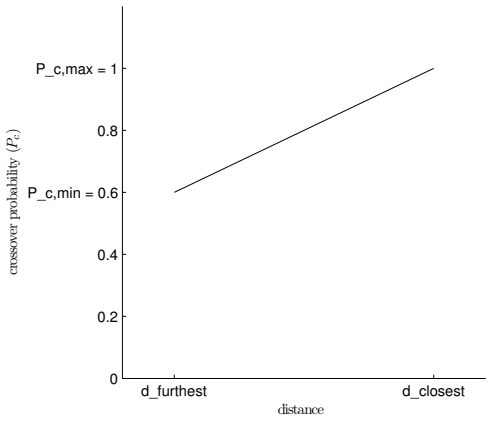


Figure 3.7: Evolution of the crossover probability as a function of the distance between two solutions

Second, we have to analyze how the chromosome is coded in order to see how we will apply the crossover step. For instance, let us choose the Layer (L) column as indicator for the crossover. If we order the matrix in Table 3.5 following this column, we will have the Table 3.6 and the Table 3.7 for another solution that we will use for the crossover.

Now, without loss of generality, let us suppose that the crossover happens (randomly) on line 7. One of the child will be the Table 3.8 and we see that the original scenario is not preserved since the first column (in bold) contains the same ID several times.

We observe that the only possible indicator for the crossover step is the ID column. Indeed if we order the two parents following the ID column, we have the Tables 3.9 and 3.10. If we still consider that the crossover occurs on line 7, we can have the child shown in Table 3.11. We see that there is no inconsistency since the scenario is still respected.

Mutation

A mutation cannot happen anywhere in the matrix. Indeed, if we take the conclusion about the choice of the crossover row indicator, all the elements except the ID column can mutate.

The mutation used is a random uniform distribution $U([a, b])$, where $[a, b]$ is the interval of values allowed for the mutation. For the discrete values, we use equidis-

Table 3.5: Output matrix template

ID	L	X	Y	S	AR	LX	LY	T
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
9	2	9	5	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
11	2	8.6	0.4	6.62	1	2.5729	2.5729	90
12	3	8.3	7.4	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 3.6: First parent, ordered by L column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
9	2	9	5	6.74	1	2.5962	2.5962	90
11	2	8.6	0.4	6.62	1	2.5729	2.5729	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
12	3	8.3	7.4	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 3.7: Second parent, ordered by L column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
4	1	1.6	8.5	18.6	1	4.3128	4.3128	90
5	1	1.2	1.3	18.6	1	4.3128	4.3128	90
6	1	0.6	4.7	18.6	1	4.3128	4.3128	90
3	2	5.9	4	18.6	1	4.3128	4.3128	90
9	2	5.4	8	6.74	1	2.5962	2.5962	90
10	2	8.5	8.1	6.62	1	2.5729	2.5729	90
11	2	2.8	4.6	6.62	1	2.5729	2.5729	90
1	3	7	6.3	18.6	1	4.3128	4.3128	90
2	3	7.4	9.8	18.6	1	4.3128	4.3128	90
7	3	5.6	5.5	0.54	1	0.7348	0.7348	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 3.8: Possible child, ordered by L column

ID	L	X	Y	S	AR	LX	LY	T
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
9	2	9	5	6.74	1	2.5962	2.5962	90
1	3	7	6.3	18.6	1	4.3128	4.3128	90
2	3	7.4	9.8	18.6	1	4.3128	4.3128	90
7	3	5.6	5.5	0.54	1	0.7348	0.7348	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 3.9: First parent, ordered by ID column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	1	7.5	11	6.74	1	2.5962	2.5962	90
9	2	9	5	6.74	1	2.5962	2.5962	90
10	1	4.5	8	6.62	1	2.5729	2.5729	90
11	2	8.6	0.4	6.62	1	2.5729	2.5729	90
12	3	8.3	7.4	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 3.10: Second parent, ordered by ID column; the line specifies the crossover cut

ID	L	X	Y	S	AR	LX	LY	T
1	3	7	6.3	18.6	1	4.3128	4.3128	90
2	3	7.4	9.8	18.6	1	4.3128	4.3128	90
3	2	5.9	4	18.6	1	4.3128	4.3128	90
4	1	1.6	8.5	18.6	1	4.3128	4.3128	90
5	1	1.2	1.3	18.6	1	4.3128	4.3128	90
6	1	0.6	4.7	18.6	1	4.3128	4.3128	90
7	3	5.6	5.5	0.54	1	0.7348	0.73485	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
9	2	5.4	8	6.74	1	2.5962	2.5962	90
10	2	8.5	8.1	6.62	1	2.5729	2.5729	90
11	2	2.8	4.6	6.62	1	2.5729	2.5729	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

Table 3.11: Possible child, ordered by ID column

ID	L	X	Y	S	AR	LX	LY	T
1	2	4.5	6	18.6	1	4.3128	4.3128	90
2	2	4	0.4	18.6	1	4.3128	4.3128	90
3	3	3.1	6.9	18.6	1	4.3128	4.3128	90
4	3	8.4	10.1	18.6	1	4.3128	4.3128	90
5	3	6.6	2.2	18.6	1	4.3128	4.3128	90
6	1	9	5.7	18.6	1	4.3128	4.3128	90
7	1	10	3.5	0.54	1	0.7348	0.7348	90
8	3	2.8	5.5	6.74	1	2.5962	2.5962	90
9	2	5.4	8	6.74	1	2.5962	2.5962	90
10	2	8.5	8.1	6.62	1	2.5729	2.5729	90
11	2	2.8	4.6	6.62	1	2.5729	2.5729	90
12	3	5.7	7.5	0.66	1	0.8124	0.8124	90

ID: component identification number; L: assigned layer;
(X, Y): geometrical coordinate; S: size (mm^2); AR: aspect ratio;
(LX, LY): length in X and Y axis; T: assigned technology for the layer

tributed probabilities. The mutation probability of a child will be set as $P_m = 0.3$. Empirical observations have shown that smaller mutation probability can easily lead to a local optimum. This can be explained by the fact that we choose that only one single element of a line can mutate instead of the whole line. If a child is forced to mutate, then one randomly-chosen value of the whole matrix will mutate within the range of values it is allowed to take.

A Gaussian mutation is also a common operator but it has not been chosen since it will produce a solution which is not far from the original one. This is not really interesting to have similar solutions when exploring the design space for integrated circuits. Of course, a large standard deviation value can be chosen but this will be likely to produce solution which are out of the feasible bounds.

Consistency test

Of course, infeasible solutions may appear after the crossover/mutation step, since these operations are made with randomness. In order to verify that, we perform a test on each new solution to check if there is overlapping between the blocks. Currently, the solutions which are infeasible will be discarded. Of course, it is possible to apply some repair mechanism but it is to be investigated as future work even if we already produce feasible solutions.

Termination

Three stop conditions have been implemented and are based on what is commonly used:

— Maximum number of iterations, set to 100.
— Maximum elapsed time, set to 30 minutes.
— Maximum number of iterations with an unchanged population, set to 10.

The maximum elapsed time has been chosen arbitrarily for quick testing purposes. Having a simulation time of a few hours would not be a problem either. Indeed, in practice, the optimization of one single architecture can take from several hours to several days with the current design tools.

3.5 Results and their use for a designer

The optimization was done for three criteria (so that we can visualize the design space) and the main results are given in Fig. 3.8 (3D plot) and Fig. 3.9 (interconnection length-cost projection). Two conclusions can be drawn from that figure:

— The [10; 20] range values for the IL criteria: a small enhancement of the IL value leads to a large increase of the cost so the interest for a design with more than 4 tiers seems low.
— The [260; 280] range values for the cost criteria: a small increase of the price can give a large enhancement of the performance. A designer might consider accepting a slightly higher price for a sensitively better performance, knowing that this information can be quantified with an accurate model. Indeed, with the estimate model that we propose, a small 10% increase of the cost can decrease the IL by 60%.

These results from did not take into account the degree of freedom of aspect ratio. If we go further by releasing a degree of freedom and allowing varying aspect ratios, we can have the Pareto front shown in Fig. 3.10. This figure shows the Pareto front from Fig. 3.9 (without aspect ratio, symbol: $\cdot$) alongside with a new Pareto front (with aspect ratio, symbol: +).

As expected, the Pareto front given when considering varying aspect ratios is globally better. Furthermore, by comparing the two graphs, we can see an interesting area where the two frontiers begin to merge at the cost value 350. This means that, in that area, it is not necessary to take the aspect ratio into account. Once again, these kind of information can be important in the design of an IC and yet they would not be available with the current design flows since only a small number of possibilities are explored. Indeed, due to the sequential nature of the current design flows, such degrees of freedom are not even tried since they dramatically increase the duration of each optimization loop.

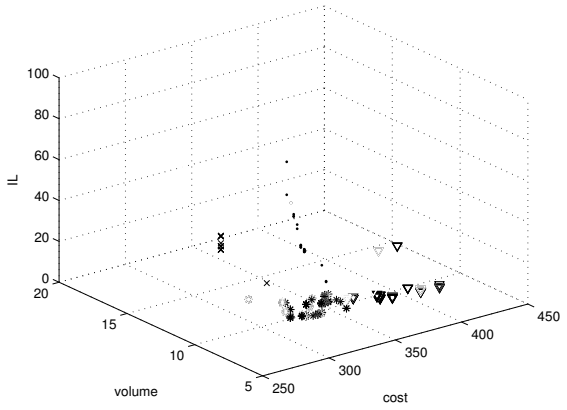


Figure 3.8: 3D view (interconnection length-volume-cost) of the Pareto frontier

· : 1 tier; × : 2 tiers; + : 3 tiers; * : 4 tiers; ▽ : 5 tiers

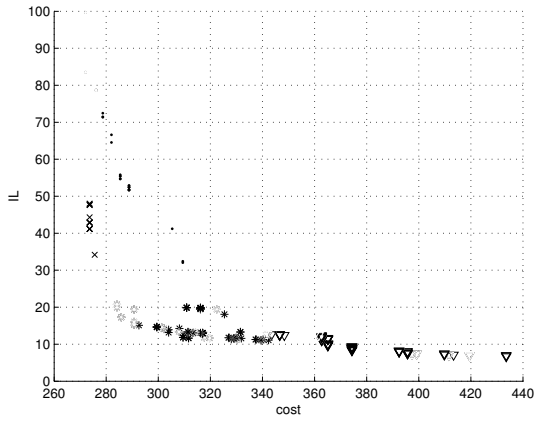


Figure 3.9: IL-cost projection view of the Pareto frontier

· : 1 tier; × : 2 tiers; + : 3 tiers; * : 4 tiers; ▽ : 5 tiers

3.6 Conclusion

The results have thus shown interesting analyses that can be relevant for a designer. First, using a multi-objective optimization methodology does not only consider all the criteria at the same time but also proceed to an extensive design space

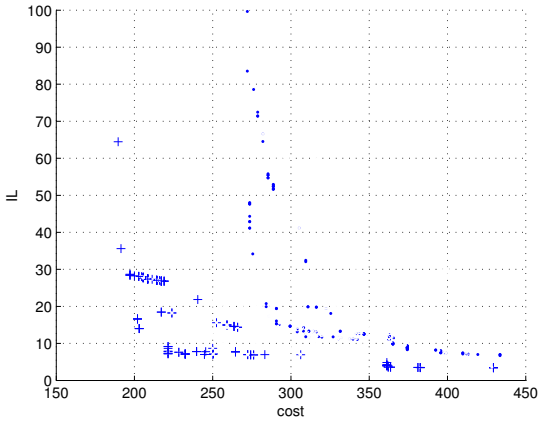


Figure 3.10: IL-cost projection view of the Pareto frontier (with and without aspect ratio)

· : Pareto front without aspect ratio; + : Pareto front with aspect ratio

exploration which is rarely done with current tools. Second, the qualitative results shown here can give relevant information to the designer and they can be quantified with a more accurate model. Third, the flexibility of MOO allows to easily consider new degrees of freedom without having to change the paradigm. Finally, this methodology and the associated algorithms has shown positive indicators of convergence and robustness as it will be shown in the next chapter.

4

Robustness of the methodology

In Chapter 3, we have shown how a 3D-SIC can be modelled in order to apply a NSGA-II metaheuristic. The obtained results indicate that multi-objective optimization can give qualitative and quantitative information to a designer that would not be available with current design tools.

In this chapter we will take a deeper look at the results of the multi-objective optimization. We will analyze the properties of the design space in order to have the view over the convergence and the robustness of our methodology. We will use classical performance indicators such as the contribution indicator, the spread indicator, the binary ϵ-indicator, the unary hypervolume indicator and the density of the Pareto-front which are presented in [11, 49]. These indicators can be grouped in 3 categories defined in [11]:

1. The convergence-based indicators:

   "The convergence metrics evaluate the effectiveness of the solutions in terms of the closeness to the optimal Pareto front."

2. The diversity-based indicators:

   "Diversity indicators measure the uniformity of distribution of the obtained solutions in terms of dispersion and extension. In general, the diversity is researched in the objective space."

3. The hybrid indicators: that combine both convergence and diversity measures.

4.1 Contribution indicator

The contribution is a convergence-based binary indicator. The contribution of an approximation PO_1 relatively to another approximation PO_2 is the ratio of non-dominated solutions produced by PO_1 in PO^*, which is the set of Pareto solutions of $PO_1 \cup PO_2$:

$$Cont(PO_1/PO_2) = \frac{\frac{\|PO\|}{2} + \|W_1\| + \|N_1\|}{\|PO^*\|} \tag{4.1}$$

where PO is the set of solutions in $PO_1 \cap PO_2$, W_1 the set of solutions in PO_1 that dominate some solutions of PO_2 and N_1 the set of non-comparable solutions of PO_1. This value has to be greater than 0.5 to indicate that PO_1 is better than PO_2 in terms of convergence to the Pareto front.

The Table 4.1 and the Fig. 4.1 show the evolution of the averaged contribution indicator over the iterations for the 5 experiments. We see that for the first iterations, $Cont(PO_i/PO_{i-1})$ is greater than 0.5, which means that the algorithm does indeed improve the solutions, then for the last iterations, the indicators are lower than 0.5 which means that there is a convergence.

Iteration	$Cont(PO_i/PO_{i-1})$
1	0.7626
2	0.8510
3	0.8917
4	0.8788
5	0.8295
...	...
38	0.4522
39	0.3870
40	0.2369

Table 4.1: Evolution of the contribution indicator

4.2 Spread indicator

The spread indicator I_s combines the distribution and cardinality to measure the dispersion of the approximated set A:

$$I_s = \frac{\sum_{u \in A} |\{u' \in A : \|F(u) - F(u')\| > \sigma\}|}{|A| - 1} \tag{4.2}$$

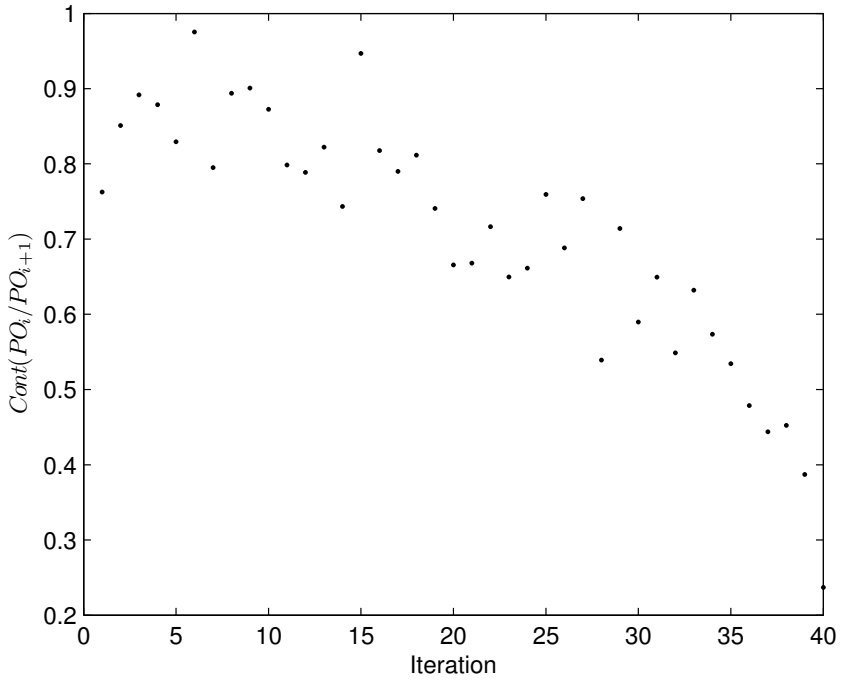


Figure 4.1: Evolution of the contribution indicator

where $F(u)$ is a fitness function and $\sigma > 0$ a neighborhood parameter. The closer is the measure to 1, the better is the spread of the approximated set A.

The Fig. 4.2 shows the results of the spread indicator I_s function of the neighborhood indicator σ (all the 5 experiments share the same graph shape). We see that the Pareto front is well spread ($I_s \geq 0.9$) for $\sigma < 0.35$ in average for the 5 runs (normalized values), so we can consider that the algorithm produces a well-spread approximation of the Pareto front.

4.3 Binary ϵ-indicator

The binary ϵ-indicator is a convergence-based indicator. It will give the quality of a solution front in comparison with a another set, with regards to all objectives. Let us consider a minimization problem with n positive objectives. An objective vector $f^1 = (z_1^1, z_2^1, \ldots, z_n^1)$ is said to ϵ-dominate another objective vector $f^2 = (z_1^2, z_2^2, \ldots, z_n^2)$ if $\forall 1 \leq i \leq n : \ z_i^1 \leq \epsilon \cdot z_i^2$, for a given $\epsilon > 0$. A binary ϵ-indicator $I_\epsilon(A, B)$ gives the factor ϵ such that for any solution in B there is at least one solution in A that is not worse by a factor of ϵ in all objectives. $I_\epsilon(A, B)$ can be calculated as

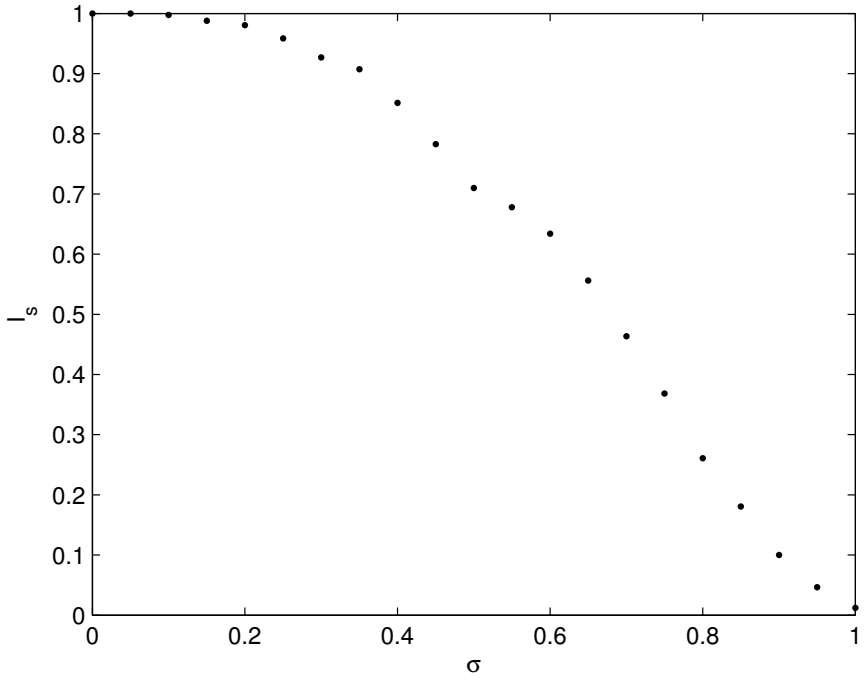


Figure 4.2: Spread indicator I_s function of the neighborhood parameter σ

follows [49]:

$$I_\epsilon(A,B) = \max_{z^2 \in B} \min_{z^1 \in A} \max_{1 \leq i \leq n} \frac{z_i^1}{z_i^2} \tag{4.3}$$

The reference set R computed from all the experiments will serve to show the evolution of the ϵ-indicator over time. This evolution is shown in Table 4.2 (for averaged values after each 10 iterations). We can see that in the first iterations, $I_\epsilon(A,R) > 1$ and $I_\epsilon(R,A) \approx 1$ which means that the front is improved while in the last iterations, $I_\epsilon(A,R) > 1$ and $I_\epsilon(R,A) > 1$ which shows convergence.

A comparison of the binary ϵ-indicators between each experiment is also given, in Table 4.3. We can see that $I_\epsilon(A,B) > 1$ and $I_\epsilon(B,A) > 1$ which indicates that neither A weakly dominates B nor B weakly dominates S. This means that the generated front is consistent from one experiment to another.

Also, in Table 4.4 are given the ϵ-indicator between iterations of an experiment (the same observations apply for the other runs). We can see that in the first iterations, the front is always improved ($I_\epsilon(A_i, A_{i-1}) > 1$ and $I_\epsilon(A_{i-1}, A_i) \leq 1$) while in the last iterations, it begins to converge ($I_\epsilon(A_i, A_{i-1}) > 1$ and $I_\epsilon(A_{i-1}, A_i) > 1$).

Iteration	Averaged $I_\epsilon(A, R)$	Averaged $I_\epsilon(R, A)$
1	5.5255	1.0178
10	4.4307	1.0235
20	3.8102	1.1023
30	2.3614	1.1234
40	1.6569	1.2381

Table 4.2: Evolution of the binary ϵ-indicator (averaged values compared to the reference set R) over time

$I_\epsilon(A, B)$	Run 1	Run 2	Run 3	Run 4	Run 5
Run 1	1	1.7270	1.3594	1.8664	1.2542
Run 2	1.5713	1	1.4122	1.7791	1.3420
Run 3	1.4737	1.8638	1	1.9268	1.3069
Run 4	1.3436	1.4564	1.2843	1	1.2365
Run 5	1.4214	1.7650	1.3918	1.7545	1

Table 4.3: Comparison of the binary ϵ-indicators for each experiment

Iteration	$I_\epsilon(A_i, A_{i-1})$	$I_\epsilon(A_{i-1}, A_i)$
1	1.6674	1.1053
2	1.7223	1
3	2.4439	1
4	1.7477	1
5	2.0577	1
...	...	...
38	1.8788	1.4916
39	1.5344	1.8065
40	1.9862	1.6609

Table 4.4: Comparison of the binary ϵ-indicators between iterations of the same experiment

4.4 Unary hypervolume indicator

The hypervolume is an hybrid indicator. Since we already used a binary indicator (*epsilon*), we will use the hypervolume indicator I_H in its unary form. I_H, associated with an approximation set A is given by the volume of the space portion that is weakly dominated by the set A [11].

The evolution of the hypervolume (averaged values) is given in Table 4.5 and in Figure 4.3. We can see that the value is (linearly) increasing over time.

The result for each experiment is also given, in Table 4.6 and the used reference point is the worst point computed from all the sets. As we can see, the values are rather consistent from one run to another.

Iteration	Averaged hypervolume
1	0.0574
10	0.0701
20	0.0876
30	0.0931
40	0.1036

Table 4.5: Evolution of the unary hypervolume indicator (averaged values compared to the reference set R) over time

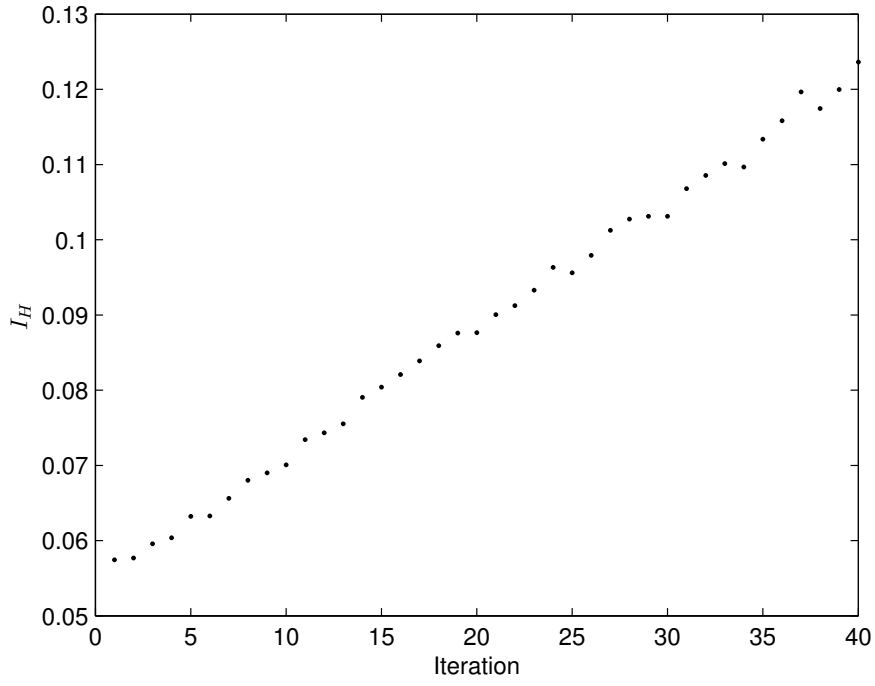


Figure 4.3: Evolution of the unary hypervolume indicator (averaged values compared to the reference set R over time

Run	I_H
1	0.1171
2	0.1105
3	0.1015
4	0.1185
5	0.0939

Table 4.6: Hypervolume for each experiment

4.5 Density of the Pareto front - gaps in the frontier

Another indicator of the Pareto front structure is its density. Here we will measure the density by finding gaps in the frontier. This will be done by counting the number of solutions in the neighborhood of another solution. Since the extreme distance between two solutions is 450.364 in average for the 5 runs (non-normalized), we consider that an acceptable neighborhood is twice the distance between two solutions if all the solutions were equidistant. We have thus a neighborhood of about 2. This test has shown that there was always at least one solution near another one, even for a neighborhood of 1, meaning that the algorithm can produce a sufficiently dense frontier.

4.6 Conclusion

In this chapter, we have shown that the used methodology has proved to be robust even if the problem contains criteria of heterogeneous nature. With the several indicators that we have analysed, we can conclude that the algorithm we used can show good properties of convergence, spread and density.

The methodology can thus be considered as robust even if the problem is not homogeneous (heterogeneous nature of the criteria) and this shows that a multi-criteria paradigm can be suitable for the design of 3D-SICs.

Also, analyses have been performed to determine the shape of the Pareto front. Globally, the Pareto front is not convex, as one may expect since the heterogeneous nature of the criteria. This is probably due to some correlation between the criteria but this is still to be investigated.

Bibliography

[1] S. Al-Sarawi, D. Abbott, and P. Franzon, "A review of 3-d packaging technology," *Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on*, vol. 21, no. 1, pp. 2 –14, feb 1998.

[2] S. Das, A. Fan, K.-N. Chen, *et al.*, "Technology, performance, and computer-aided design of three-dimensional integrated circuits," pp. 108–115, 2004.

[3] E. Beyne and B. Swinnen, "3D System Integration Technologies," *Integrated Circuit Design and Technology, 2007. ICICDT '07. IEEE International Conference on*, pp. 1–3, May 2007.

[4] K. Kumagai, C. Yang, H. Izumino, *et al.*, "System-in-silicon architecture and its application to H.264/AVC motion estimation for 1080HDTV," *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, pp. 1706–1715, Feb. 2006.

[5] T. Kgil, S. D'Souza, A. Saidi, *et al.*, "PicoServer: using 3D stacking technology to enable a compact energy efficient chip multiprocessor," *SIGPLAN Not.*, vol. 41, no. 11, pp. 117–128, 2006.

[6] A. W. Topol, J. D. C. La Tulipe, L. Shi, *et al.*, "Three-dimensional integrated circuits," *IBM J. Res. Dev.*, vol. 50, no. 4/5, pp. 491–506, 2006.

[7] S. Gupta, M. Hilbert, S. Hong, *et al.*, "Techniques for Producing 3D ICs with High-Density Interconnect," in *21st International VLSI Multilevel Interconnection Conference, Waikoloa Beach*, 2004.

[8] P. Pieters and E. Beyne, "3d wafer level packaging approach towards cost effective low loss high density 3d stacking," in *Electronic Packaging Technology, 2006. ICEPT '06. 7th International Conference on*, 26-29 2006, pp. 1 –4.

[9] R. Patti, "Three-dimensional integrated circuits and the future of system-on-chip designs," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1214–1224, June 2006.

[10] J. Brans and B. Mareschal, *PROMETHEE-GAIA. Une Méthodologie d'Aide à la Décision en Présence de Critères Multiples*. Paris, France: Ellipses, 2002.

[11] E.-G. Talbi, *Metaheuristics : from design to implementation*. John Wiley & Sons, 2009.

[12] G. B. Dantzig, *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*. New York: Wiley, 1951, ch. XXI.

[13] N. Karmarkar, "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, no. 4, pp. 373–395, Dec. 1984. [Online]. Available: http://dx.doi.org/10.1007/BF02579150

[14] P. Vincke, *Multicriteria Decision-Aid*. J. Wiley, New York, 1992.

[15] M. Ehrgott and X. Gandibleux, *Multiple Criteria Optimization. State of the art annotated bibliographic surveys*. Kluwer Academic, Dordrecht, 2002.

[16] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley, New York, 546 pp, 1986.

[17] J. Holland, "Adaptation in natural and artificial systems," 1975.

[18] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decision Sciences*, vol. 8, no. 1, pp. 156–166, 1977.

[19] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[20] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549, 1986.

[21] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech Concurrent Computation Program, Tech. Rep., 1989.

[22] M. Dorigo, "Optimization, Learning and Natural Algorithms (in Italian)," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[23] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, *Metaheuristics for Hard Optimization*. Springer, 2006.

[24] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, Apr. 2004. [Online]. Available: http://dx.doi.org/10.1007/s00158-003-0368-6

[25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 182–197, 2000.

[26] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, pp. 221–248, 1994.

[27] V. Belton and T. Stewart, *Muliple Criteria Decision Analysis: An Integrated Approach*. Dordrecht: Kluwer Academic, 2002.

[28] J. Dyer, "MAUT – multiattribute utility theory," in *Multiple Criteria Decision Analysis: State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott, Eds. Boston, Dordrecht, London: Springer Verlag, 2005, pp. 265–285. [Online]. Available: http://www.springeronline.com/sgw/cda/frontpage/0,11855,5-165-22-34954528-0,00.html

[29] T. Saaty, "The Analytic Hierarchy and Analytic Network Processes for the Measurement of Intangible Criteria and for Decision-Making," in *Multiple Criteria Decision Analysis: State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott, Eds. Boston, Dordrecht, London: Springer Verlag, 2005, pp. 345–408. [Online]. Available: http://www.springeronline.com/sgw/cda/frontpage/0,11855,5-165-22-34954528-0,00.html

[30] R. Benayoun, J. de Montgolfier, J. Tergny, and O. Larichev, "Linear programming with multiple objective functions: Step method (STEM)," *Mathematical Programming*, vol. 1, no. 3, pp. 366–375, 1971.

[31] H. Nakayama and Y. Sawaragi, "Satisficing trade-off method for multiobjective programming," in *Interactive Decision Analysis*, ser. Lecture Notes in Economics and Mathematical Systems, M. Grauer and A. Wierzbicki, Eds. Springer Berlin Heidelberg, 1984, vol. 229, pp. 113–122. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-00184-4_13

[32] J. Brans and P. Vincke, "A preference ranking organization method," *Management Science*, vol. 31, no. 6, pp. 647–656, 1985.

[33] R. Benayoun, B. Roy, and B. Sussman, "ELECTRE: une méthode pour guider le choix en présence des points de vue multiples," SEMA-METRA International, Direction Scientifique, Tech. Rep., 1966, note de travail 49.

[34] J. Figueira, S. Greco, and M. Ehrgott, *Multiple Criteria Decision Analysis: State of the Art Surveys*. Boston, Dordrecht, London: Springer Verlag, 2005.

[35] P. Fishburn, *Utility Theory for Decision Making*. Wiley, New York, 1970.

[36] R. Keeney and H. Raiffa, *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York", 1976.

[37] B. Mareschal and J. Brans, "Geometrical representations for MCDA," *European Journal of Operational Research*, vol. 34, pp. 69–77, 1988.

[38] J. Brans, B. Mareschal, and P. Vincke, "Promethee: a new family of outranking methods in multicriteria analysis," in *Operational Research, IFORS 84*, J. Brans, Ed. North Holland, Amsterdam, 1984, pp. 477–490.

[39] M. Behzadian, R. Kazemzadeh, A. Albadvi, and M. Aghdasi, "PROMETHEE: A comprehensive literature review on methodologies and applications,"

European Journal of Operational Research, vol. 200, no. 1, pp. 198–215, 2010. [Online]. Available: http://EconPapers.repec.org/RePEc:eee:ejores:v:200:y:2010:i:1:p:198-215

[40] J. Figueira, V. Mousseau, and B. Roy, "ELECTRE methods," in *Multiple Criteria Decision Analysis: State of the Art Surveys*, J. Figueira, S. Greco, and M. Ehrgott, Eds. Boston, Dordrecht, London: Springer Verlag, 2005, pp. 133–162. [Online]. Available: http://www.springeronline.com/sgw/cda/frontpage/0,11855,5-165-22-34954528-0,00.html

[41] D. Milojevic, T. Carlson, K. Croes, R. Radojcic, D. F. Ragett, D. Seynhaeve, F. Angiolini, G. V. der Plas, and P. Marchal, "Automated pathfinding tool chain for 3D-stacked integrated circuits: Practical case study." in *3DIC*. IEEE, 2009, pp. 1–6.

[42] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *ICCAD '04: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 306–313.

[43] H. Yan, Q. Zhou, and X. Hong, "Thermal aware placement in 3D ICs using quadratic uniformity modeling approach," *Integr. VLSI J.*, vol. 42, no. 2, pp. 175–180, 2009.

[44] G. Pelosi, "The finite-element method, part i: R. l. courant [historical corner]," *Antennas and Propagation Magazine, IEEE*, vol. 49, no. 2, pp. 180 –182, april 2007.

[45] D. Milojevic, L. Montperrus, and D. Verkest, "Power dissipation of the network-on-chip in multi-processor system-on-chip dedicated for video coding applications," *Journal of Signal Processing Systems*, p. 15, June 2008.

[46] F.-J. Veredas, M. Scheppler, W. Moffat, and B. Mei, "Custom implementation of the coarse-grained reconfigurable adres architecture for multimedia purposes." in *FPL*, T. Rissa, S. J. E. Wilton, and P. H. W. Leong, Eds. IEEE, 2005, pp. 106–111.

[47] L. Davis, "Adapting operator probabilities in genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 61–69. [Online]. Available: http://dl.acm.org/citation.cfm?id=93126.93146

[48] J. Hart and A. Shogan, "Semi-greedy heuristics: An empirical study," *Operations Research Letters*, vol. 6, pp. 107–114, 1987.

[49] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, and V. da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117 – 132, april 2003.