Chapter 3
Optimization Algorithms for Design Space Exploration of Embedded Systems

Enrico Rigoni, Carlos Kavka, Alessandro Turco, Gianluca Palermo, Cristina Silvano, Vittorio Zaccaria, and Giovanni Mariani

Abstract This chapter is dedicated to the optimization algorithms developed in the MULTICUBE project and to their surrounding environment. Two software *design space exploration* (DSE) tools host the algorithms: Multicube Explorer and modeFRONTIER. The description of the proposed algorithms is the central part of the chapter. The focus will be on newly developed algorithms and on *ad-hoc* extensions of existing techniques in order to face with discrete and categorical design space parameters that are very common when working with embedded systems design. This chapter will also provide some fundamental guidelines to build a strategy for testing the performance and accuracy of such algorithms. The aim is mainly to build confidence in optimization techniques, rather than to simply compare one algorithm versus another one. The "no-free-lunch theorem for optimization" has to be taken into consideration and therefore the analysis will look forward to robustness and industrial reliability of the results.

3.1 Introduction

The optimization problems emerging in the design of embedded systems—and in particular those addressed within the MULTICUBE project— are multi-objective and characterized by the fact that all configuration parameters are discrete and possibly categorical.

Having more than one objective to be optimized (maximized or minimized) implies that the outcome of the optimization process is not a single solution but a set of solutions. This set of solutions, which is called the Pareto front, represents the trade-offs between the different (and possibly conflicting) objectives. A design is a Pareto design (or *point*) if it is not possible to improve one of its objective values without deteriorating at least another one, like in Fig. 3.1.

A. Turco (✉)
ESTECO, Trieste, Italy
e-mail: alessandro.turco@esteco.com

C. Silvano (eds.), *Multi-objective Design Space Exploration of Multiprocessor SoC Architectures*,
DOI 10.1007/978-1-4419-8837-9_3,

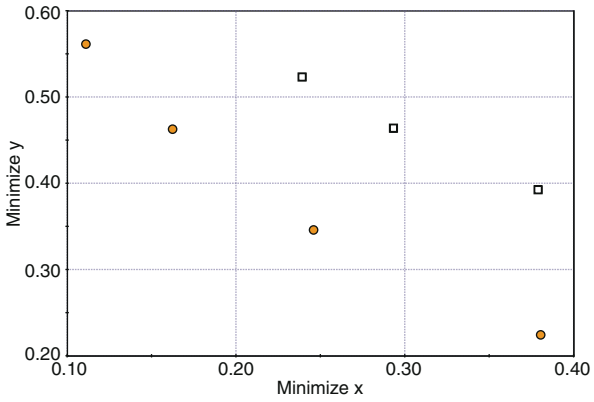

Fig. 3.1 An example of Pareto-dominance in a two-objective minimization problem. The circles represent non-dominated points, while squares are dominated

In problems with discrete categorical configuration parameters, the types of the input variables are discrete ranges and might also be unordered collections, meaning that optimization methods which assume an order relation cannot be used profitably. For example, the number of processors on the platform is a discrete ordered variable since it is an natural number. Instead, the type of *branch predictor* to be used (e.g., static, two-level adaptive, etc.) is a categorical variable since an ordering between the different instances cannot be defined.

The evaluation of the objective values for the designs selected for exploration is usually performed through a simulator during the optimization phase. Simulators accuracy depends on their level of abstraction which is inversely proportional to their computational complexity. A manual procedure may include additional delays to the already long simulation time.

Two examples of automatic optimization frameworks will be considered in this chapter. The first framework is the open source Design Space Exploration (DSE) Multicube Explorer. It has been initially conceived for the kind of problems discussed above. Throughout this chapter, a description of its optimization algorithms introduced within the framework will be provided.

The second tool is modeFRONTIER, a commercial software which has been widely used worldwide for more than ten years in different domains like aerospace, appliances, pharmaceutics, civil engineering, manufacturing, marine multi-body design, crash, structural, vibro-acoustics and turbo-machinery. All these domains define multi-objective optimization problems, but in continuous or mixed (continuous and discrete) domains, not in complete discrete and possibly categorical domains like the SoC design problems. Due to this reason, an initial re-target process to add support for categorical variables to modeFRONTIER has been carried out and the actual release of the software contains this work as well as the algorithms developed within the project.

The automatic design space exploration performed by one of these two tools is governed by an optimization algorithm. The algorithm is responsible for choosing the new configurations which have to be simulated and for analyzing the results obtained. The optimization phase can be preceded by a Design Of Experiments (DOE) study and it can be followed by some Post Processing analysis.

The optimization algorithms implement the mathematical strategies, or *heuristics*, which are designed in order to obtain a good approximation of the actual Pareto frontier. Real-world optimization problem are solved through rigorously proven converging methodologies only in an extremely few instances, since the high number of input parameters and the low smoothness of objective functions involved limit the possible usage of classical algorithms. Therefore a wide catalogue of heuristics have been designed trying to achieve a good balance between exploration of the design space and exploitation of the information carried by the best solutions found.

The Design of Experiments (DOE) usually precedes the optimization stage. The aim of a DOE is to test specific configurations regardless the objectives of the optimization run but rather considering their pattern in the input parameters space. It provides an *a priori* exploration and analysis which is of primary importance when a statistical analysis has to be performed: for example, a reduced factorial DOE can be the basis for a *principal components analysis*, since it avoids correlations among input parameters and therefore it highlights input-output relationships. Moreover, almost all optimization algorithms require a starting population of designs to be considered first and the DOE can provide it, eventually generating random input values if no other preference has emerged yet.

The Post Processing analysis could represent the starting point for a new attempt of optimization, but at the same time it conveys a deep insight into the problem structure. Starting from a correlation matrix, for example, it is possible to recognize if some objectives are conflicting or if they are correlated and there is no need to involve all of them in the optimization process.

A comprehensive list of publication on this topic is out of the aim of this introduction. The description of the algorithms listed in Sect. 3.3 will include the references necessary to understand them. The theoretical structure of multi-objective optimization as well as classical methods are deeply investigated in the book by Miettinen [10]. The paper by Erbas et al. [5] describes in details a MPSoC design problem solved with genetic algorithms (GA) and it introduces important concepts like evaluation metrics and repair mechanisms.

The innovation of the algorithms developed within the MULTICUBE project cannot be correctly evaluated without considering the whole picture: it is mathematically proven that it is not possible to rank optimization algorithms on the basis of their performances over all possible problems. On the contrary, it is possible to specialize an optimization strategy in order to solve "better" (later in this chapter the concept of quality for a multi-objective solution set will be addressed more precisely) a defined class of problems. The work done by MULTICUBE partners results in a very satisfactory trade-off between applicability and accuracy which is the true achievement of the project. Not only algorithms contribute to this result and this is the reason why this chapter will introduce also some feature of the software framework containing them and also an anticipation of the validation procedure necessary for an industrial knowledge transfer.

This chapter is organized as follows: Sect. 4.2 presents the problem description and the software framework while Sect. 4.3 introduces the design space exploration algorithms used throughout the project. Section 4.4 presents a detailed descriptions of validation strategy while Sect. 4.5 summarizes the main content of this chapter.

3.2 Problem Description and Software Framework

To define an optimization problem it is necessary to identify the input parameters of the problem, which constitute the Design Space, and the output parameters, often called *metrics*. Among the output parameters some objectives must be selected in order to build the Objective Space. Other outputs can be considered as constraints, but also input parameters can be combined in order to obtain the desired constraints. A typical example is the couple of variables processor type-cache size: some processors may support only certain values of cache size and a constraint must be added to the problem to enforce this requirement.

The output values associated to a given configuration of input parameters are obtained through a simulator. The choice of which designs have to be explored is responsibility of the optimization algorithm and of the DOE designer for what concerns the starting points. The whole structure comprising inputs, outputs, simulator, algorithm is called workflow.

One of the first achievements of the MULTICUBE project is the definition of a common framework for generating a complete workflow using a standard XML format. Both DSE tools, Multicube Explorer and modeFRONTIER, can accept the same configuration file and they are able to run optimization starting from the information listed in it. Specific tags have been created in order to specify input and output values, a precise syntax is used to define constraints and the path to the executable simulator is included.

The communication between the optimization algorithm and the simulator is performed through XML file as well. The DSE tool is also responsible for launching parallel instances of the simulator, if the computational resources available allow them.

From the strictly mathematical point of view, the problems addressed can be characterized as follows. The vector of input variables $\mathbf{x} = (x_1, \ldots, x_N)$ can take values into different sets depending on the nature of its components. An integer variable is usually comprised between a lower and an upper bound, $x_i^L \leq x_i \leq x_i^U$, where $i \in [1, N]$. However it is possible that only some integer may be of interest, for example only the powers of 2. This kind of variables are similar to categorical variables, but they maintain the notion of order. On the contrary, another way of calling a categorical variable is *discrete unordered* variable since this is their main feature. The list of admissible values is called *catalogue*. The optimization problem is then

$$\begin{cases} \min & \mathbf{f}(x_1, \ldots, x_N), \\ \text{such that} & \mathbf{g}(x_1, \ldots, x_N) \geq 0, \\ & \mathbf{h}(x_1, \ldots, x_N) = 0. \end{cases} \tag{3.1}$$

The vector functions $\mathbf{f}$, $\mathbf{g}$ and $\mathbf{h}$ can have arbitrary dimensions. It is not necessary that all the functions involved in the problem are minimization targets. For example if a function $\phi(\mathbf{x})$ has to be maximized and it should occupy the i-th component of $\mathbf{f}$, a simple change of sign can solve the problem defining $\mathbf{f}_i := -\phi$. The same procedure can be applied to "less or equal to" constraints.

A point is said to be *feasible*, if it satisfies all the constraints addressed in $\mathbf{g}$ and $\mathbf{h}$. If the number of objectives is M, then given two feasible points $\mathbf{x}$ and $\mathbf{y}$, the point $\mathbf{x}$ is said to *dominate* $\mathbf{y}$ if $\mathbf{f}_i(\mathbf{x}) \leq \mathbf{f}_i(\mathbf{y})$ for $i = 1, \ldots, M$ and there exists at least one index $j \in [1, M]$ such that $\mathbf{f}_i(\mathbf{x}) < \mathbf{f}_i(\mathbf{y})$. Dominance induces a partial ordering onto the design and the objective space.

A vector of input parameters $\bar{\mathbf{x}}$ is said to be a *Pareto design* if there is not any other point dominating it. The corresponding point in the objective space $\mathbf{f}(\bar{\mathbf{x}})$ is said to be a *Pareto point*. The set containing all the Pareto points is the *Pareto front* and it is the solution of problem 3.1.

3.3 Algorithms

This section presents a brief description of the multi-objective optimization algorithms that have been tested within the project and implemented in the Design Space Exploration tools, highlighting the algorithm characteristics that are related to the specific properties of the optimization of an Embedded System. The algorithms presented in this section can be divided in three groups:

- Standard algorithms. This first group includes the algorithms that are well known in the multi-objective optimization field and have been implemented in the Design Space Exploration tools by following the models proposed by their original designers with none or minimal enhancements. The algorithms NSGA–II and MOGA–II belong to this group.
- Enhanced algorithms. This group includes all algorithms that are based on a previously defined algorithm but include noticeable enhancements that make them adequate for the specific problems addressed. The algorithms Enhanced-MOSA, Enhanced-ES and Enhanced-MOPSO belong to this group.
- New algorithms. This group includes all algorithms that have been specifically defined in the MULTICUBE project for multi-objective optimization in the context of System-on-Chip (SoC) design optimization. The algorithms MFGA and APRS belong to this group.

3.3.1 Standard Algorithms

The algorithms described in this section have been employed successfully in multi-objective optimization for years. They were not precisely designed for categorical variables, but they can treat them as simply discrete ones without excessively deteriorating their performances.

3.3.1.1 NSGA–II

The Non-dominated Sorting Genetic Algorithm II (NSGA–II) was developed by Prof. Deb et al. [3] at Kanpur Genetic Algorithms Laboratory (KanGAL). NSGA–II

is a fast and elitist multi-objective evolutionary algorithm which shares the basic structure with all genetic population-based algorithms.

The basic mechanism can be summarized as follows: starting from a parent population, some individuals are selected for generating a child population. The algorithm applies operators that work on the input variables of the selected individuals trying to improve their outputs: mutation and crossover are the standard choices for NSGA–II. This procedure is iterated for the requested number of generations.

This algorithm has some peculiar and powerful characteristics:

- It includes a fast non-dominated sorting procedure. Sorting the individuals of a given population according to the level of non-domination is a complex task, which makes in general non-dominated sorting algorithms computationally expensive for large population sizes. The adopted solution in NSGA–II performs a clever sorting strategy.
- The multi-objective search includes elitism. NSGA–II implements the multi-objective search using an elitism-preserving approach, which is introduced storing all non-dominated solutions discovered so far, beginning from the initial population. Elitism enhances the convergence properties towards the true Pareto-optimal set.
- A parameter-less diversity preservation mechanism is adopted. Diversity and spread of solutions is guaranteed without the use of extra parameters (like sharing parameters for example). NSGA–II adopts a suitable parameter-less niching approach called crowding distance, which estimates the density of solutions in the objective space, and a crowded comparison operator, which guides the selection process towards a uniformly spread Pareto frontier.
- The constraint handling method does not make use of penalty parameters. The algorithm implements a modified definition of dominance in order to solve constrained multi-objective problems efficiently: usual dominance is the criterion to sort feasible points. A feasible point will be always preferred to an unfeasible one. Unfeasible points are sorted looking at the (normalized, possibly) constraint violations sum.

The NSGA–II procedure is described graphically in Fig. 3.2. The individuals of the parent population P_t of size N and the new population Q_t of the same size (created by applying the variation operators crossover and mutation, to individuals in P_t selected by binary tournament) are grouped together. The combined population R_t is then sorted based on its non-domination level obtaining sets of non-dominated solutions (F_1, F_2, ...). The new population P_{t+1} is created by selecting the best non-dominated sets that can be completely inserted into the new population (with a combined size smaller or equal than N) plus members from the last set (which cannot be fully accommodated) selected using the crowded comparison operator.

NSGA–II allows both continuous (real-coded) and discrete (binary-coded) design variables. Specific mutation and crossover operators are applied to each kind of variables. Categorical (non-ordered discrete) parameters are treated as simple discrete ones. This drawback can be compensated by increasing the exploration capabilities of the algorithm allowing a larger mutation probability. NSGA–II tests a

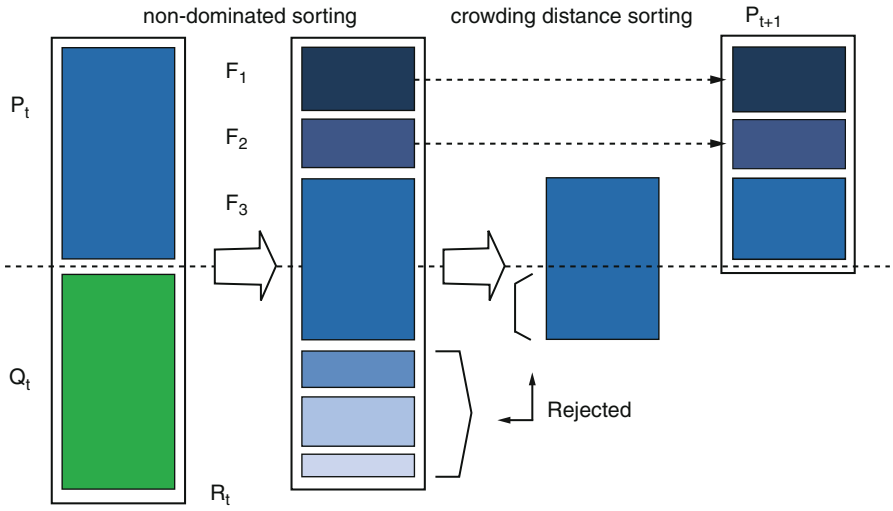

Fig. 3.2 NSGA–II sorting procedure

wider range of candidate solutions (hence the fictitious locality is damped) and its efficient elitism and selection routines drive the optimization towards the Pareto front. This algorithm has been implemented in Multicube Explorer while it was provided by modeFRONTIER from early releases.

3.3.1.2 MOGA–II

MOGA–II (Multi-Objective Genetic Algorithm, with elitism) was originally formulated by Poloni [11] and it reached the actual implementation with the introduction of elitism. This second version is equipped also with an optional improved crossover operator (directional crossover), but since it is not suited for discrete problems, its description is omitted.

This algorithm accepts only discrete variables (possible continuous variables have to be discretized with the desired accuracy), which are encoded as in classical genetic algorithms. Three operators govern the reproduction phase: one-point crossover, mutation and selection. The probabilities under which one of them is chosen are user-defined parameters.

Elitism guarantees that the best points remain in the parent population and hence hopefully their children will exhibit a similar behavior. MOGA–II achieves this result keeping a record of all non-dominated points found up to the current generation. The new population is created extracting randomly the requested number of new parents from the union (without repetitions) of the elite set and the set of newly generated children. This procedure gives to the algorithm a good balance between exploration and exploitation phase. This balance is fundamental for the performance of a multi-objective global search: the search space has to be sufficiently explored, but at the

same time the number of evaluated points must be kept low and the algorithm has to converge rapidly to the Pareto set.

Elitism in MOGA–II works in this direction. The elite set usually contains a few points in early stages of the optimization. Moreover, these points belong to the last generation with a high probability. Hence only a fraction of them will enter the next parents population promoting exploration. As long as new generations are created, the elite set grows and the probability of finding out a new elite point decreases. Therefore in the updated population there will be many points coming from the elite set exploiting their features.

In order to save simulation time, steady evolution was preferred against the classical generational evolution in MOGA–II. In almost any industrial application, the computational time spent in evaluating a point is much larger than the time employed by the optimization algorithm to prepare and request a new evaluation. A generational algorithm would keep a significant part of computational resources idle (in a cluster or grid systems), since before every generation is created, the algorithm needs to get the results of the evaluation of all the individuals of the previous generation.

Within a steady evolution, the child point replaces its parent immediately. Every time an evaluation ends, a new one is requested choosing randomly a parent from the actual population and applying the chosen operators. The elitism procedure is scheduled with the same frequency as in the case of a generational evolution, but it can be performed while some points are still being evaluated, using the information stored so far. This introduces a little delay in the propagation of the information, but it prevents delays in the computational grid or cluster system. The negative effects of this issue increase with the dimension of the population, but decrease as the number of requested generations increases. This algorithm is proprietary of ESTECO and it was provided by modeFRONTIER from early releases.

3.3.2 Enhanced Algorithms

Literature reports a continuous improvement of available methods since multi-objective optimization is a research field constantly pressed by new applications. New applications require new and better answers. In this section we considered as *enhanced* algorithms the implementations of well known algorithms which have been rewritten within the MULTICUBE project in order to better adapt to the SoC design problem. Indeed, specific operators have been designed for treating categorical variables and a careful attention has been addressed to the problem of optimizing also the computing resources needed for design evaluation.

3.3.2.1 Enhanced-MOSA

The Simulated Annealing (SA) method for optimization was introduced by Kirkpatrick [6], on the basis of a thermo-dynamical analogy.

The evolution of such a system is controlled by an external parameter called temperature. A related energy can be assigned to every possible configuration of the

system. When an initial configuration is perturbed, the difference in energy between the two states is responsible for the evolution of the system: if the new state is favorable, i.e. if it decreases the energy, then the new configuration is accepted. If this is not the case, the new state is accepted or rejected according to a probability distribution derived by Boltzmann. This distribution is a function of the temperature and when the temperature is high, the probability of accepting an unfavorable state is larger (see Metropolis et al. [9]).

The energy for the MOSA algorithm is (a suitable function of) the non-dominated ranking already described for NSGA–II. Hence a new point is always accepted if it is non-dominated by its parent. It will be also accepted in the opposite situation depending on a temperature-based probability distribution.

Temperature is simply a parameter, initially user-defined, which evolves during the optimization. MOSA starts with a hot phase accepting many points in order to explore the design space. Afterwards, a cold phase, during which only the best points survive, represents the exploitation part of the algorithm.

The creation of a child configuration from a parent one in the original formulation of the algorithm is a directional perturbation. A random direction *versor* represents the direction of the perturbation, while its length is predicted by a schedule similar to the temperature one: starting from a specified upper bound, the value decreases during the hot phase and it reaches the imposed lower bound in the cold phase. If the perturbation vector brings the point out of variables space boundaries, a bouncing routine will maintain the feasibility of the samples. This procedure also helps in differentiating and enhancing the exploration and the exploitation capabilities of the algorithm.

The concept of direction has no meaning working with categorical variables. The enhanced version of MOSA takes in account this problem keeping at the same time the idea of a tunable perturbation. Every categorical variable at each iteration has a probability to change its value depending on a law similar to the one applied to the temperature and perturbation length. If the value has to be changed, a new value is chosen randomly from the available list.

A lifespan counter is introduced in order to compensate for the uncontrolled randomness in the search for the best values for categorical variables. Especially during the hot phase there could be sequences of parents and children moving towards dominated regions of the objective space because of the Metropolis acceptance criterion. If the number of subsequent unwanted increasing in energy exceeds a threshold, Enhanced-MOSA replaces the child with its better-fitting parent.

A steady state evolution is a second enhancement of the algorithm. The procedure is very similar to the evolution implemented in MOGA–II with the obvious change of the updating schedule: there is no elite set to be updated, but instead Enhanced-MOSA changes the value of the temperature, the perturbation length and the probability of replacement for categorical variables.

The standard implementation of MOSA is available in modeFRONTIER from early releases and it has also been implemented in Multicube Explorer. The described enhancements were developed for the MULTICUBE project and were implemented in modeFRONTIER.

3.3.2.2 Enhanced-ES

Evolution Strategies (ES) is a multi-objective optimizer that is able to reproduce different evolutionary algorithms. These algorithms share the selection operator and the operators schedule, while they differ in the ratio between parents and children points and in the definition of the set of points among which the new parents are selected.

The ES approach was first used at the Technical University of Berlin. During the search for the optimal shapes of bodies in a flow, the classical attempts with the coordinate and the well-known gradient-based strategies were unsuccessful. So, the idea was conceived of proceeding strategically. Rechenberg and Schwefel [14] proposed the idea of trying random changes in the parameters defining the shape, following the example of natural mutations.

Usually, there is a huge difference between mathematical optimization and optimization in the real-world applications. Thus, ES were invented to solve technical optimization problems where no analytical objective functions are usually available.

The general Evolutionary Strategy scheme is the following:

1. Initial population creation;
2. Individuals evaluation;
3. Selection of the best individual(s);
4. Recombination;
5. Mutation;
6. Individuals evaluation;
7. Return to step 3 until the required number of generation is achieved

Selection of the best results may be done only on the set of children or on the combined set of parents and children. The first option, which is represented usually with the notation (λ, μ)-ES, can "forget" some good results when all the children are worse than their parents. The second option, which is represented with the notation $(\lambda + \mu)$-ES, applies a kind of elitist strategy for the selection.

The best solutions may be identified in different ways: the implementation of ES provided in modeFRONTIER is capable of approximating the Pareto Set in multi-objective optimization using the Non-dominated/Crowding distance sorting technique as done in NSGA–II.

The main source of variation is a mutation operator based on a normal distribution. The standard deviation of this distribution changes during the generations in an adaptive manner. Each input variable has its own deviation with an initial and a minimal value that can be arbitrarily tuned.

A completely different operator has been introduced for categorical variables in the context of MULTICUBE. If such a variable is selected for mutation, its value is changed following an uniform distribution (i.e. the choice is completely random), since locality has no meaning. An adaptive strategy is instead performed over the probability of mutating each variable.

A discrete recombination operator is a second source of variability. It resembles the classical crossover operator, where information coming from two different parents

is exchanged producing a child point. The value of each variable has the same probability of coming from both parents.

The standard implementation of ES is available in modeFRONTIER from early releases and it has also been implemented in Multicube Explorer. The described enhancements were developed for the MULTICUBE project and were implemented in modeFRONTIER.

3.3.2.3 Enhanced-MOPSO

Particle Swarm Optimization (PSO) is an optimization methodology that mimics the movements of a flock of birds finding food [7]. PSO is based on a population of particles moving through an hyper-dimensional search space. Each particle possesses a *position* and a *direction*; both variables are changed to emulate a well known social-psychological phenomenon: mimic the success of other individuals in the population (also called *swarm*).

More formally, the position $\mathbf{x}$ for a single particle i is updated by means of a velocity vector $vecv$ by means of the following equation:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \delta_i(t) \tag{3.2}$$

while the direction vector is updated with the following equation:

$$\begin{aligned}\delta_i(t) = W\delta_i(t-1) + C_1 r_1(\mathbf{x}_{pbest_i} - \mathbf{x}_i(t-1)) \\ + C_2 r_2(\mathbf{x}_{gbest} - \mathbf{x}_i(t-1))\end{aligned}$$

where W is called the inertia weight, C_1 is the cognitive learning factor, C_2 is the social learning factor, r_1, r_2 are random numbers in the range [0, 1], $\mathbf{x}_{pbest_i}$ is the best position of particle i with respect to the minimization problem, $\mathbf{x}_{gbest}$ is the global best found up to time t. The formulation of the problem leads to solutions which try to 'follow' the leader's $\mathbf{x}_{gbest}$ position as well as attracting solutions versus the *personal* best solution of the particle $\mathbf{x}_{pbest_i}$.

Dealing with Multi-objective problems. So far, several approaches have been proposed for extending the formulation of the PSO technique to the multi-objective domain [2, 13]. The Enhanced-MOPSO technique is based on an "aggregating" approach where the swarm is equally partitioned in n subswarms, each of which uses a different cost-function which is the product of the objectives combined with a set of exponents randomly chosen.

In other words, given the original set of objectives $\{f_1 \ldots f_m\}$, each sub-swarm i solves the following problem:

$$\min_{\mathbf{x}\in X} \prod_{j=1\ldots m} f_j^{p_{i,j}}(\mathbf{x}) \tag{3.3}$$

where $p_{i,j}$ is a set of randomly chosen exponents. It can be shown that solutions to Problem 3.3 lie on the Pareto surface of the original problem. This approach

is different with respect to [2] because the latter uses a linear combination of cost functions $\{f_1 \dots f_m\}$. Linear combination can be heavily biased on highly valued cost-functions disregarding low-valued ones.

Dealing with the discrete design space. The essential nature of the solution space of the problems faced within the MULTICUBE Project is discrete, while the approaches presented so far deal with a continuous search space. Several proposals have been made so far in the literature to extend classical PSO to the discrete domain. One method for addressing the discrete design space exploration problem is applying particle swarm optimization to binary-valued solution elements [8]. In this case, while the velocity term is still real-valued, the position term is actually chosen between 0 and 1 by means of a sigmoidal function. Another method leverages the construction of a probability distribution for each value of the position vector [12]. The probability distribution is derived from the current value of the position vector which, in turn, depends on the velocity vector. The probability distribution is then transformed into a single integer number during fitness evaluation of each particle.

Enhanced-MOPSO is based on the concepts of *random walk* theory. A random walk is a path with the following properties:

- It has a starting point.
- The distance from one point to the next is constant
- The direction from one point to the next is picked up at random.

The position of the particle is still updated with the traditional rule:

$$\mathbf{x}_i(t) = \mathbf{x}_i(t-1) + \delta_i(t) \tag{3.4}$$

while each component k of the direction vector is updated with the following rule:

$$\delta_{i,k}(t) = \begin{cases} \text{sign}(x_{gbest,k} - x_{i,k}(t-1)) & \text{if(rand()} < p) \\ \text{randint}(-1, 1) & \text{otherwise} \end{cases} \tag{3.5}$$

where $p \in [0, 1]$ is a parameter of the algorithm.

As can be noted, the direction of the particle is updated following two rules: rule 1 attracts the particle versus the leader of the swarm ($gbest$), rule 2 forces the particle to follow a random walk. This ensures us to jump out from local minima in the objective function shown in Eq. 3.3.

3.3.3 New Algorithms

The algorithms listed below can be considered as completely new proposal in the scientific literature [17].

3.3.3.1 MFGA

The acronym of this new algorithm stands for Magnifying Front Genetic Algorithm since its main purpose is to work on the local Pareto front in three directions: towards (approaching the true front), laterally (obtaining a wider front) and internally (enhancing the uniformity of the front samples).

With the introduction of elitism, genetic algorithms such as NSGA–II found a very good answer to the problem of converging faster than previous implementations. The question now is how to converge better, without slowing down.

Elitism is considered as the main cause of too concentrated Pareto fronts [1, 4]. If the optimization problem is difficult, only a few points will be non-dominated and will become a sort of basin of attraction. Indeed, elitist strategies will keep these points in the parent population and crowding distance or similar techniques are not useful to "dilute" them until a large number of Pareto points are found. However without elitism the request for quality cannot even be addressed, since the algorithm would converge too slowly. Literature reports two promising ideas in order to modify this operator without removing it.

Deb and Goel [4] proposed a controlled elitism approach. Their algorithm selects the new parent population accepting also dominated points with a preference for those points coming from less crowded regions. Computed points are ranked by domination and ordered by crowding distance. An exponentially decreasing number of points are selected from each rank starting from the top of the list. Figure 3.3 shows how a combined population R_t of size $2N$ (parents plus children) is reduced

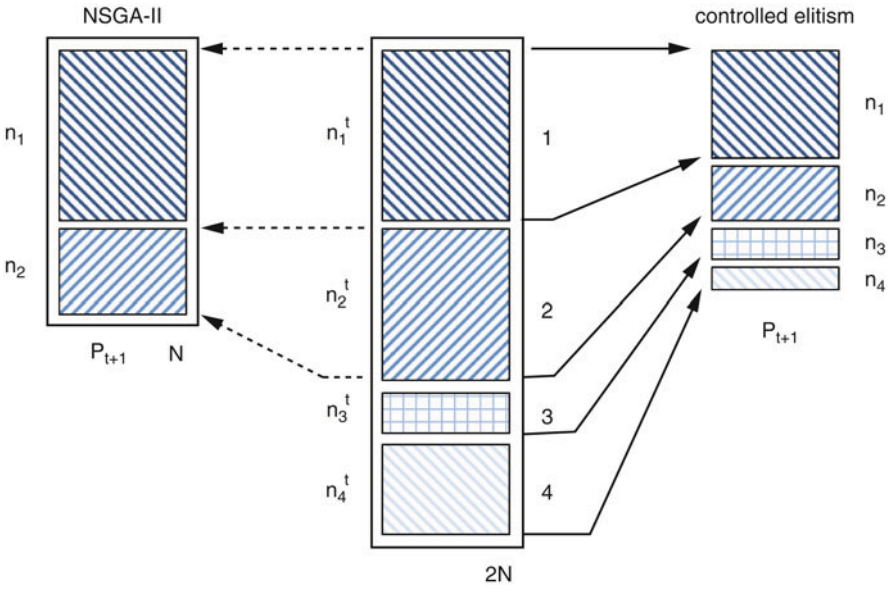

Fig. 3.3 Controlled Elitism sorting procedure vs NSGA–II

to a population P_{t+1} of size N by selecting n_i individuals from each non-dominated front of size n_i^t, using crowded tournament selection to reduce each front if necessary.

This choice helps in obtaining a more uniform front, filling possible gaps with points coming from higher ranks. It is remarkable that slight improvements are achieved also in convergence rate and in lateral spreading of the computed front, as reported in the cited paper.

Aittokoski and Miettinen [1] studied a different strategy called variable size population. Their idea is to transform all first-rank points into new parents regardless their number. The result is an algorithm that cannot perform worse than a classical elitist one considering convergence rate (since it does not waste any useful information) and it guarantee a better diversity maintenance.

MFGA is an algorithm that tries to *managing elitism* mixing the two cited ideas in an original scheme, including also a steady state evolution. The algorithm switches automatically between the two approaches depending on the dimension of the local Pareto front, allowing:

- wider exploration of the design space: new generations are created following the reduced elitism approach until the local Pareto front reaches one third of the population size (fixed by the DOE size).
- better exploitation of the obtained information: the parents update is done following the variable size scheme, only for local fronts that contain a number of points from one to two third of the population size.
- diversity preservation: the reduced elitism is reintroduced for larger fronts.

The steady state evolution implemented together with this mixed procedure is quite demanding from the computational point of view, since every time the evaluation of a new point is performed, the parent population is completely recomputed including the new achieved information. This choice is well suited for problems involving long simulation time.

Classical operators govern the parents-children recombination, but they are rebuilt trying to enlarge the kind of problems treatable using them [17]. Mutation and crossover operators act in MFGA variable-wise in order to treat easily mixed problems involving real, integer and categorical variables. MFGA was completely designed by ESTECO for the MULTICUBE project. Its inclusion in future commercial releases of modeFRONTIER is planned.

3.3.3.2 APRS

The acronym of this new algorithm stands for Adaptive-windows Pareto Random Search. It is an iterative optimization algorithm that tries to optimize locally the each Pareto solution found up to the previous iteration.

The main characteristics of the algorithm are represented by the three keywords of its name: *adaptive-windows*, *Pareto* and *random search*.

- *Adaptive-windows*: the APRS is an algorithm that has a dynamic windows size which is reduced with the time spent in the exploration and with the goodness of the point found in the current windows.

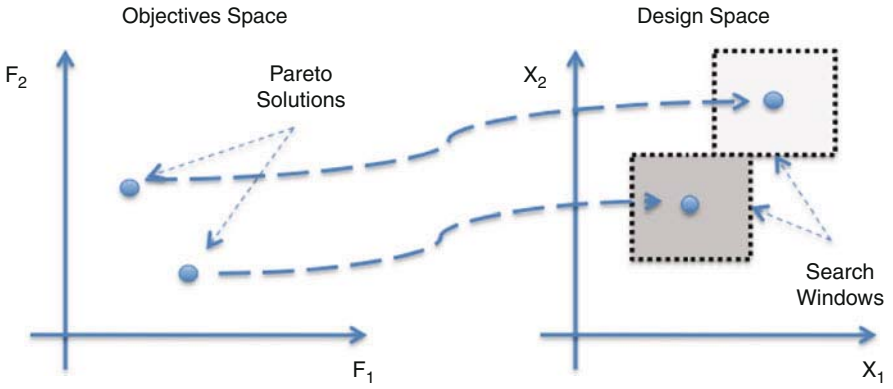

Fig. 3.4 Serach window idea in the APRS algorithm

- *Pareto*: the starting points of each iteration of the algorithm are the current Pareto solutions (found up to previous iteration) where the search windows are centered on.
- *Random search*: the new configurations to be evaluated are randomly selected within those windows.

The idea behind the algorithm is simple and it is based on iterative local search in the neighborhood of good points: the current Pareto set (see Fig. 3.4). Moreover, as for temperature in Simulated-Annealing strategies, the algorithm has a parameter represented by the search-window size that reduces the dynamism of the algorithm during the optimization life-time. In particular, the search-window centered on each Pareto point is reduced whenever no better solutions are found in the iteration.

More in detail, the general structure of the algorithm is the following:

1. Creation of an initial set of solutions selected within the entire DS by using a DoE
2. Initial solutions evaluation (S)
3. Identification of the Pareto solutions ($P' = ParetoFilter(S)$)
4. While the termination condition is not met
   (a) Use the Pareto set as initial set for each iteration $S < -P'$
   (b) For each point in the Pareto set ($p \in P'$)
      i) Randomly select a configuration within the reduced DS delimited by the search window centered on the pareto point ($W(p)$)
      ii) Evaluate the selected configuration s ($S = S + s$)
   (c) If the Pareto set is not changed from the previous iteration (i.e. if $P' == ParetoFilter(S)$), reduce the search window ($|W| = |W| * \alpha$)
   (d) Otherwise, Identification of the new Pareto set ($P' = ParetoFilter(S)$)

The algorithm exposes three parameters, the initial set $|S|$ size, the initial size of the windows $|W|$ and the windows reduction coefficient α. APRS was completely designed by POLIMI for the MULTICUBE project and has been implemented in Multicube Explorer.

3.4 Validation Strategies

The so called *no-free-lunch theorem for optimization* [18] is a rigorous mathematical theorem which states the impossibility of ranking algorithms on the basis of their performances: averaging on all possible problems, every algorithm will obtain results exactly equal to all the others. A direct corollary is that if one algorithm appears better than another one in solving a specific problem, there must exist another problem in which the original algorithm appears worse than the other.

This theorem implies that an optimization algorithm is as important as the validation strategy which may reveal its ability in solving a specific set of problems. This issue is not a mere academic question, but it has relevant effects on the industrial exploitation of the achievement obtained. It is of primarily importance to build confidence on the proposed algorithmic strategies and to prove their robustness. This is the main achievement of the researches carried on by the MULTICUBE project, whose partners agreed on the need of a validation step in order to transfer the (possibly academic) high quality knowledge to the industrial world.

This section focuses on the first two steps performed in this direction within the project. The first one consists in showing that all the algorithm described in Sect. 3.3 can solve a benchmark problem of SoC design in a satisfactory manner. The second step is to show the advantages of an automatic optimization process with respect to a traditional approach. The combination of these two results can guarantee the reliability of the proposed approach. At the same time, the validation process must be intended as an iterative process which follows but at the same time precede the development of new optimization strategies.

3.4.1 Algorithm Comparison

The problem selected as benchmark for the algorithms validation is based on the Low-Power Processor Use Case delivered by STM-C described in Chap. 8. In this paragraph, we compare the previously introduced algorithms to identify the most suitable to the architecture under consideration. The executable model for the design space exploration is the *sp2sim* simulator, which models the SP2 microprocessor design. The benchmark application selected is the *164.gzip* application, based on the popular gzip application.

The design space consists of 11 configuration parameters, 7 system metrics and 3 objectives. The configuration parameters are grouped in three categories: out-of-order execution parameters, cache system parameters and branch prediction parameters, as shown in Table 3.1. The system metrics are grouped in three categories: performance, power dissipation and area occupation metrics, as shown in Table 3.2. The three objectives to be minimized have been selected from each one of the metrics group: total_cycle, power_dissipation and area.

In order to compute optimization metrics that provide a reasonable measure of quality of the algorithms, it is necessary to compare the Pareto fronts obtained by

Table 3.1 Input parameters for the benchmark problem

Category	Parameter	Description	Values
Out of order execution	rob_depth	Reorder buffer depth	32, 48, 64, 80, 96, 112, 128
	mreg_cnt	Rename register number	16, 32, 48, 64
	iw_width	Instruction window width	4, 8, 16, 24, 32
Cache system	icache_size	Instruction cache size	16, 32, 64
	dcache_size	Data cache size	16, 32, 64
	scache_size	Secondary cache size	0, 256, 512, 1024
	lq_size	Load queue size	16, 24, 32
	sq_size	Store queue size	16, 24, 32
	mshr_size	Miss holding register size	4, 8
Branch prediction	bht_size	Branch history table size	512, 1024, 2048, 4096
	btb_size	Branch target buffer size	16, 32, 64, 128

Table 3.2 Output parameters for the benchmark problem

Category	Metric	Description
Performance	total_cycle	Total cycle number
	total_instr	Total instruction number
	IPC	Instruction per cycle
Power dissipation	total_energy	Total energy consumed
	total_dissipation	Average power dissipation
	peak_power_dissipation	Peak power dissipation
Area occupation	Area	Area occupied

the algorithms with a reference Pareto front, which should be the real Pareto front of the optimization problem, or at least a good approximation of it. The design space of the problem outlined above consist of 1,161,216 designs. Since the time required to evaluate all these designs is too large to be considered as an option, a statistical study of the configuration parameters was performed in order to try to identify parameters which could be fixed to a constant value to reduce the size of the design space without a significant reduction of the problem interest for SoC designers. A statistical study was performed with random exploration using Multicube Explorer, exploring a set of 5,000 randomly selected designs.

In order to reduce the size of the design space, the following parameters with low contribution where identified: rob depth, lq size, sq size and mshr size. Adequate constant values were selected for them, reducing the size of the design space to only 9,216 designs. The reduced problem was considered valid both from the point of view of SoC designers and from the point of view of the mathematical properties of the design space and its associated Pareto front. All designs in the reduced design space were evaluated by performing a full factorial multi-level exploration obtaining the real Pareto front in a few days of execution time. This Pareto front consists of 18 points. Figure 3.5 shows if and when the considered algorithms discover them.

The performance measures selected for the algorithm comparison concern both time and quality. Since by far the most time consuming component of the optimization procedure is the simulator execution, its number of evaluations has been selected as a fair measure of the required algorithm execution time. Concerning the quality of the

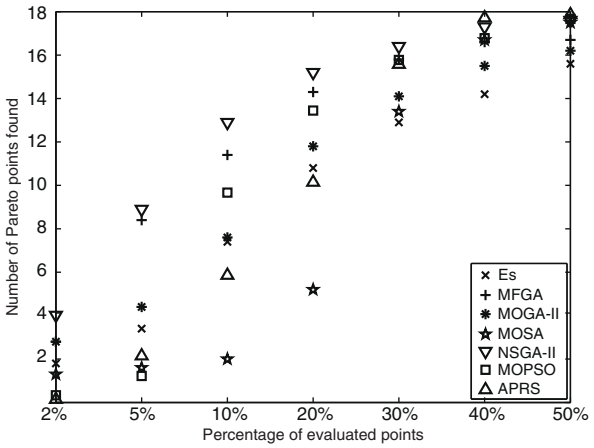

Fig. 3.5 Algorithm performance comparison on the reduced benchmark problem

solutions found by each algorithm, a set of four metrics has been selected considering the following criteria:

- Accuracy: measured as the distance between the obtained Pareto front and the reference (or real) front.
- Uniformity: measured as the distribution of the solution set in the trade-off curve.
- Extent: measured as the coverage of the objective space considering boundary points.

The D-metric, Δ-metric and ∇-metric have been selected from [5], while the ADRS (Average Distance from Reference Set) metric has been selected from [15]. Both D-metric and ADRS provide indication of accuracy, Δ-metric of uniformity and ∇-metric of extent.

A fair evaluation of non-deterministic algorithms requires several repeated runs without changing any parameter besides the random generator seed. Notwithstanding the relative small search space consisting of only 9,216 designs, very large variations can be observed in the algorithms behavior and a rigorous study needs to analyze also this aspect. It was agreed that 10 repetitions were a good trade off among statistical issues, purposes of the evaluation and significance of the problem. Preliminary tests were performed in order to estimate the best choices for the tunable parameters which then have been kept fixed.

Algorithms parameters are usually problem-dependent. Some of them depend also on the user expectations: the optimal choices (if any) for parameters controlling the ratio between exploration and exploitation phase (like temperature schedule in MOSA, for example) are strictly related to the number of evaluations the user can afford. It was decided to tune these parameters considering the largest target (i.e. 50% of the design space, as described below) and accepting possible worse results in the partial samples.

The evaluation process then proceeds checking at fixed numbers of explored points the quality of the non-dominated front found so far. The steps selected for the

evaluation are: 184 designs (corresponding to about 2% of the design space), 461 (5%), 922 (10%), 1,843 (20%), 2,765 (30%), 3,686 (40%) and 4,608 (50%). Only the requests of evaluation of new designs were counted, since sometimes the algorithms request the evaluation of an already evaluated design due to the inherent behavior of their random engines. In any practical application, the time needed to retrieve the stored information is incomparably smaller than the time that would be spent by a new evaluation. Besides the fact that in this experiment it is known in advance any value thanks to the previous full factorial exploration, the real optimization process was simulated by counting each design only once.

Some algorithms occasionally cannot generate new designs when working with discrete problems if some parameters are not set properly. The chosen benchmark problem has a small variable space and in the exploitation phase (where usually recombination is less effective than in the exploration one) the algorithms may get stuck in the endless repeated evaluation of the same designs. This behavior was observed and was overcame by increasing the explorative capabilities of the algorithms.

A last remark concerns the input variables nature. They are all discrete, but none of them is categorical. This choice allows to test fairly a wider range of algorithms, but on the other hand, the test cannot highlight completely the improvements gained with the enhancements described above.

With a small variance, all algorithms reach an ADRS value below 2% evaluating 30% of the design space (see Fig. 3.6). This result can be considered very promising. Variations in the slope of the lines for some algorithms are a consequence of possible different behaviors in successive phases of the optimization process. The most clear example is MOSA with its hot and cold phase. MOSA is tuned to reach the top of the exploitation phase at 50% of evaluations and therefore its results are the worst up to 20–30%, while at the end it is one of the most effective algorithms. APRS shows a similar behavior.

It is very difficult to analyze the uniformity and the extent of the partial front found by the algorithms during the optimization process. The true Pareto front is

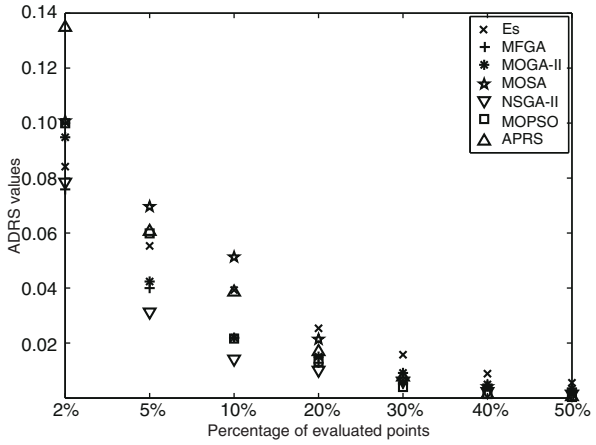

Fig. 3.6 Algorithm performance comparison on the reduced benchmark problem in terms of ADRS metric [15]

not uniform itself, since the problem is highly discrete both in the search space and in the objective space: notwithstanding the real values achievable by the objective functions, it is observed the formation of clusters of points with a cylindrical shape. Only some tips belong to the Pareto front and the distance between two nearby solution points is relatively large.

Extent metric gives some insight into the evolution of the non-dominated front. Some algorithms (MOPSO, MOSA, and APRS) span a wider range than others. This is due to a sharper division between exploration and exploitation phase. Indeed, the higher values of ∇-metric are achieved when the local front contains points which will be dominated by the following generations. These designs may span a wider area in the objective space resulting in a higher value of the extent metric.

The analysis of the metrics values obtained offer a deep insight into the algorithms structure in addition to the comparison information. Efficiency assessments can be drawn in terms of ADRS metric and of number of Pareto points found. Under this perspective, all algorithms behaved in a satisfactory manner on the proposed benchmark problem: as remarked before, starting from 30% of the design space exploration, all scored less than 0.02 in ADRS metric. The worst score in the achievement of Pareto points can be taken as an indicator of the reliability of the proposed algorithms: ES found 15.6 points which however corresponds to 86.6% coverage of the true Pareto front. Since in real-world problems the solution set is unknown, this percentage is clearly a good guarantee that the algorithms will reach at least a significant part of the Pareto front.

3.4.2 The Complete Optimization Problem

The problem proposed for algorithm comparison can be very useful for validating the whole optimization process as well. The procedure followed for obtaining the subspace of points for the comparison can be summarized as:

- simulate an initial set of 5,000 random points;
- perform statistical analysis on the sample in order to detect the most significant variables and the most appropriate values for the remaining ones;
- simulate all the configurations (full factorial exploration, 9,216 points) obtained varying the selected variables;
- extracting the non-dominated set.

The approximated Pareto set obtained counts 18 designs and it cost about 14,000 simulations. This procedure is similar to a manual optimization, which however in most of the cases would have produced only a reduced number of pseudo-optimal designs. The idea is to perform a first exploration in order to extract useful information about the problem. The second step is the analysis of the data obtained and their consequent exploitation through a second exploration phase, this time limited on a restricted and affordable search space.

The sequence exploration-exploitation-exploration is exactly the same hypothesized by the elitism operator of MFGA algorithm in Sect. 3.3. However that algorithm

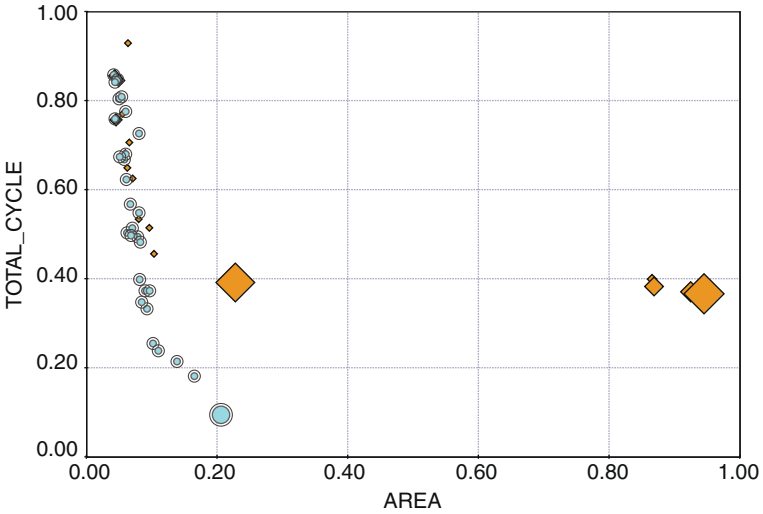

Fig. 3.7 Pareto fronts obtained with the reduced (*diamonds*) and the complete problem (*circles*). The Area and the Total Cycle objectives values are plotted on the horizontal and vertical axes, respectively. The Power Dissipation objective is represented by the size of the points: a larger point means an higher value. The highlighted points are non-dominated. The values of the metrics are normalized since the model is proprietary of STM

shows to be able of much faster convergence. An optimization run of the original problem, without the restriction introduced on the input variables, can support this observation. Indeed, MFGA after only 3,000 evaluation can produce a non-dominated set which outperform the one obtained by the procedure just described [16]. In Fig. 3.7 the two sets are plotted on a 2D plot where the third object is represented by the size of the points.

Moreover, the analysis of the input variable values producing this new and enhanced solution set contributes to the insight in the problem structures more than the statistical calculation previously carried on. A better front is found violating the prescription proposed by this latter one.

Notwithstanding these results, the validation process cannot be considered as concluded. On the contrary, it must be a complementary tool to the research for new and better algorithms. The two processes should provide new problems each other. Once a detailed validation has stated the reliability of the optimization technique over one applicative field, new field could open and new algorithms become necessary. Following these ideas, an industrial validation step will be described in Chap. 8.

3.5 Conclusions

The results contained in the present chapter shows the expertise gained within the MULTICUBE project in handling optimization problems arising in MP-SoC architecture design. Two different but complementary meanings can be associated to

the word “handling” in this context. First, the proposed methodologies can solve the problems concretely and in a satisfactory manner: the Design Exploration tools employed within the project can support all the phases of the process, since they provide appropriate solutions to define correctly the problem, they contain algorithms able to optimize the selected metrics and they include many post-processing resources. The second meaning refers to the validation path which builds the necessary reliability to exploit the research results in an industrial context.

The definition of the problem is extremely flexible, but at the same time the fixed XML vocabulary is universal in the sense that all the components (tools and simulators) needed to work on the problem are able to *speak* the same language. Different simulators with different level of abstraction can be connected with the same optimization work-flow and the different optimization tools can work with all the simulators without additional modifications.

The algorithm presented are obviously the central part of the process. Although they follow different approaches, they all try to exploit the a priori knowledge of the problem structure in order to better investigate the unknown objective space shape. The presence of categorical variables is a first obstacle to overcome and indeed many of the proposed algorithms implement particular strategies for handling this kind of variables. Following this direction there is still room for improvements: is it possible, for example, to design a categorical crossover operator? It should be an operator which mixes information between the parent designs trying to conserve possible structures or good combinations among their categorical variables.

The computational cost of the simulations is another important element to examine. The steady-state evolution implemented by some of the algorithm is a first answer. However the final number of evaluations required to achieve an accurate and uniform sample of the Pareto front is the key issue. Since all the MULTICUBE algorithms seemed to perform equivalently well, the result obtained by MFGA on the complete benchmark problem can represent a guarantee that also other algorithms can save many simulations.

Other quality of the solution set have been considered besides accuracy. Uniformity and extent are considered as complementary objectives. This opens a complete new field of research: if the result of the optimization stage is a very detailed sample of a large Pareto set, which are the points on the front that should be selected for the prototyping stage? How is it possible to help the so called Decision Maker? This stage has been considered as a separate step for long time, however recent research results in optimization tries to combine the two steps. The objective is an algorithm which returns a user-defined number of points taken from the Pareto set selecting them for their diversity.

The validation strategy proposed has a twofold merit. On one hand, simply the fact that a validation strategy has been addressed is relevant from the applicative point of view, since this is the only way of building confidence on the proposed optimization strategy. On the other hand, the validation procedure described in Sect. 3.4 contains some elements that can constitute a paradigm for evaluating optimization algorithms. A first element is to define a large set of indicators for the quality of the solution sets: a single metric can hide more than what it shows, while a deep insight in the

problem and in the algorithms can be obtained combining the results of different measurements. Another important suggestion is to check the chosen metric values at previously defined fixed numbers of evaluations. Finally, the validation stage can be considered concluded only when the new optimization algorithms have been tested against other kind of approaches (classical algorithms, manual optimization protocols, etc).

The algorithms and the procedure described in this chapter prove the overall reliability of the Design Space exploration tools, modeFRONTIER and Multicube Explorer in handling and in solving optimization problems in the field of Embedded System Design. The peculiarities of such an environment have been sufficiently recognized and exploited in order to provide solutions in affordable computational time (considering also the high consuming simulators). The study has been enough deep to open new questions for improving the capabilities of the algorithms in this field as well as for opening new research directions.

References

1. Aittokoski, T., Miettinen, K.: Efficient evolutionary method to approximate the pareto optimal set in multiobjective optimization. In: Proc. International Conference on Engineering Optimization (EngOpt) (2008)
2. Baumgartner, U., Magele, C., Renhart, W.: Pareto optimality and particle swarm optimization. IEEE Transactions on Magnetics **40(2)**, 1172–1175 (2004)
3. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga–ii. IEEE Transactions on Evolutionary Computation **6**(2), 181–197 (2002)
4. Deb, K., Goel, T.: Controlled elitist non–dominated sorting genetic algorithm for better convergence (2001). KanGal Report 200004
5. Erbas, C., Cerav–Erbas, S., Pimentel, A.: Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. IEEE Transactions on Evolutionary Computation **10**(3), 358–374 (2006)
6. Gelatt Jr., C.D., Vecchi, M., Kirkpatrik, S.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
8. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: Proceedings of the Conference on Systems, Man and Cybernetics, pp. 4104–4109 (1997)
9. Metropolis, N., Rosenbluth, A., Teller, A., Teller, E.: Equation of state calculation by fast computing machines. J. Chem. Phys. **21**(1953), 1087–1092 (1953)
10. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publisher (1999)
11. Poloni, C., Pedirola, V.: Ga coupled with computationally expensive simulations: Tools to improve efficiency. In: Genetic Algorithms and Evolution Strategy in Engineering and Computer Science, chap. 13. John Wiley & Sons (1998)
12. Pugh, J., Martinoli, A.: Discrete multi-valued particle swarm optimization. In: Proceedings of IEEE Swarm Intelligence Symposium, pp. 103–110 (2006)
13. Reyes-Sierra, M., Coello, C.A.: Multiple-objective particle swarm optimizers: A survey of the state of the art. http://www.lania.mx/∼ccoello/EMOO/reyes06.pdf.gz (2006)
14. Schwefel, H.: Evolution and Optimum Seeking. Wiley & Sons (1995)

15. Silvano, C., Zaccaria, V., Palermo, G.: ReSPIR: A response surface-based pareto iterative refinement for application-specific design space exploration. IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems **28**(12) (2009)
16. Turco, A., Kavka, C., Bocchio, S.: Optimization of an embedded parallel system-on-chip platform using modeFRONTIER. In: Poster Session at DATE'10 Conference (2010)
17. Turco, A., Kavka, C.: MFGA: a genetic algorithm for complex real-world optimization problems. International Journal of Innovative Computing and Applications **3**(1), 31–41 (2011)
18. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)