An Efficient Design Space Exploration Methodology for Multiprocessor SoC Architectures based on Response Surface Methods

Gianluca Palermo, Cristina Silvano, Vittorio Zaccaria
Politecnico di Milano
Dipartimento di Elettronica e Informazione
Via Ponzio 34/5, 20133 Milano - Italy
E-mail: {gpalermo, silvano, zaccaria}@elet.polimi.it

Abstract—Multi-Processor System-on-Chip (MPSoC) architectures are currently designed by using a platform-based approach. In this approach, a wide range of platform parameters must be tuned to find the best trade-offs in terms of the selected figures of merit (such as energy, delay and area). This optimization phase is called Design Space Exploration (DSE) and it generally consists of a Multi-Objective Optimization (MOO) problem. The design space for an MPSoC architecture is too large to be evaluated comprehensively. So far, several heuristic techniques have been proposed to address the MOO problem for MPSoC, but they are characterized by low efficiency to identify the Pareto front. In this paper, an efficient DSE methodology is proposed leveraging traditional Design of Experiments (DoE) and Response Surface Modeling (RSM) techniques. In particular, the DoE phase generates an initial plan of experiments used to create a coarse view of the target design space; a set of RSM techniques are then used to refine the exploration. This process is iteratively repeated until the target criterion (e.g. number of simulations) is satisfied. A set of experimental results are reported to trade-off accuracy and efficiency of the proposed techniques with actual workloads1.

I. INTRODUCTION

In the recent years, Multi-Processor Systems-on-Chip (MPSoC) and Chip-Multi-Processors (CMPs) have become the de facto standard for embedded and general-purpose architectures. The platform-based design methodology [1] represents the winning paradigm to design optimized architectures and meeting time-to-market constraints. In this context, parametric System on-Chip (SoC) simulation models are built and evaluated to accurately tune the architecture to meet the target application requirements in terms of performance, battery lifetime and area. This tuning phase is called Design Space Exploration (DSE) and it generally consists of a multi-objective optimization problem. The problem is generally solved by exploring a large design space consisting of several parameters at system and micro-architectural levels. So far, several heuristic techniques have been proposed to address this problem, but they are characterized by low efficiency to identify the Pareto front. Evolutionary or sensitivity based algorithms are among the most notable, state-of-the art techniques.

In this paper, we propose an iterative design space exploration methodology exploiting traditional Design of Experiments (DoE) and Response Surface Modeling (RSM) techniques. First, the DoE phase generates an initial plan of experiments used to create a coarse view of the target design space; then a set of RSM techniques are used to refine the exploration. This process is iteratively repeated until a target criterion (e.g. number of simulations) is satisfied.

The proposed methodology is highly flexible because, in principle, any combination of DoE and RSM techniques can be used. However, from the analysis of the experimental results carried out in the paper with actual workloads, two guiding strategies (low-end and high-end) have been identified. From one side, the low-end strategy combines a DoE generated randomly to a linear regression model to obtain a less accurate but very efficient exploration. From the other side, the high-end strategy combines more sophisticated DoE and RSM techniques to obtain accuracy/efficiency trade-offs.

To the best of our knowledge, although there have already been some applications of DoEs and RSM techniques to the field of performance analysis and optimization, the work proposed in this paper represents the first in-depth, comprehensive application of DoE and RSM techniques to the field of multi-objective design space exploration for on-chip multiprocessors.

The paper is organized as follows. Section II discusses the state of the art related to design space exploration while Section III introduces a formalization of the problem of the design space exploration. Section IV introduces the design space exploration methodology proposed in this paper, while Section V reports the experimental results derived from the application of the proposed methodology on a general-purpose multiprocessor platform.

II. STATE OF THE ART

Several methods have been recently proposed in literature to reduce the design space exploration complexity by using traditional statistic techniques and advanced exploration algorithms. The proposed techniques can be partitioned mainly in two categories: heuristics for architectural exploration and methods for the system performance estimation and optimization.

Among the most recent heuristics for power/performance architectural exploration we can find [2]–[4]. In [2], the
authors compare the Pareto Simulated Annealing, the Pareto Reactive Taboo Search and Random Search exploration to identify energy-performance trade-offs for a parametric superscalar architecture running a set of multimedia kernels. In [3], a combined Genetic-Fuzzy system approach is proposed. The technique is applied to a highly parametrized SoC platform based on a VLIW processor in order to optimize both power dissipation and execution time. The technique is based on a Strength Pareto Evolutionary Algorithm coupled with fuzzy system rules in order to speedup the evaluation of the system configurations. In [4], domain knowledge about the platform architecture has been used in the kernel of a design space exploration engine. The exploration problem is converted to a Markov Decision Process (MDP) problem whose solution corresponds to the sequence of optimal transformations to be applied to the platform. The requirement of domain knowledge is the main difference with respect to the previous proposals in [2], [3].

State-of-the-art, system performance optimization has been presented in [5]–[8]. A common trend among those methods is the combined use of response surface modeling and design of experiments methodologies.

In [5], a Radial Basis Function has been used to estimate the performance of a superscalar architecture; the approach is critically combined with an initial training sample set that is representative of the whole design space, in order to obtain a good estimation accuracy. The authors propose to use a variant of the Latin Hypercube method in order to derive an optimal, initial set. In [6], [7] linear regression has been used for the performance prediction and assessment. The authors analyze the main effects and the interaction effects among the processor architectural parameters. In both cases, random sampling has been used to derive an initial set of points to train the linear model. A different approach is proposed in [8], where the authors tackle performance prediction by using an Artificial Neural Network paradigm to estimate the system performance of a Chip-Multiprocessor.

In [9], [10] a Plackett-Burman design of experiments is applied to the system architecture to identify the key input parameters. The exploration is then reduced by exploiting this information. The approach shown in [9] is directed towards the optimization of an FPGA, soft-core-based design, while in [10] the target problem is more oriented to a single, superscalar processor micro-architecture.

The present work represents a step forward towards the utilization of design of experiments and response surface methodologies to an efficient performance and power optimization of multi-processor architectures.

III. DESIGN SPACE EXPLORATION

The IP reuse and platform-reconfigurability approaches are converging into a new design paradigm [1], which is strongly influencing today’s automatic system synthesis. In this context, a virtual microprocessor-based architecture can be easily extended and customized for a target application, enabling a quick, low-risk deployment. More specifically, pre-verified components belonging to a specific library are instantiated and sized to meet specific constraints on the target application domain. However, the space of configurations (or "design space") can be very large.

The optimization problem involves the minimization (maximization) of multiple objectives (such as latency, energy, area, etc.) making the definition of optimality not unique. In fact, a system which is the best from the performance point of view, can be the worst in terms of power consumption and vice-versa. To address the DSE problem, we present the theory of Multi-Objective Optimization (MOO) [11].

A discrete multi-objective design space optimization problem includes a set of n parameters, called design variables, a set of m objective functions, and a set of k constraints.

The optimization goal consists of minimizing (or maximizing) a target function, i.e.:  
\[
\min_{x \in \mathcal{X}} \mathbf{g}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))
\]

subject to the constraints:  
\[
\mathbf{c}(\mathbf{x}) = (c_1(\mathbf{x}), c_2(\mathbf{x}), \ldots, c_k(\mathbf{x})) \leq 0
\]

where:  
\[
\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathcal{X}
\]
\[
\mathbf{y} = (y_1, y_2, \ldots, y_m) \in \mathcal{Y}
\]

where \( \mathbf{x} \) is the design vector, \( \mathbf{y} \) is the objective vector, \( \mathcal{X} \) is a subset of \( \mathbb{N}^n \) (denoted as design space) and \( \mathcal{Y} \) is the projection of the design space, by the objective functions \( \mathbf{f} \), into a space of dimension \( m \), \( \mathbb{R}^m \), called objective space.

The set of all the admissible solutions of an optimization problem is called feasible region. Formally, the feasible set \( \Omega \) is defined as the set of design vectors \( \mathbf{x} \) satisfying the constraints \( \mathbf{c}(\mathbf{x}) \):

\[
\Omega = \{ \mathbf{x} \in \mathcal{X} | \mathbf{c}(\mathbf{x}) \leq 0 \}
\]

The image of the feasible set \( \Omega \) in the objective space is denoted as \( \Psi = \mathbf{f}(\Omega) \).

In single-objective optimization, the feasible set is totally ordered according to the objective function \( f \). When several objectives are involved, the situation changes and the feasible set is partially ordered. To compare two solutions \( \mathbf{f} \) and \( \mathbf{g} \) we present the concept of dominance. If \( \mathbf{f} \) is no worse for all objectives than \( \mathbf{g} \) and better than for at least one objective, we say that \( \mathbf{f} \) dominates \( \mathbf{g} \) or \( \mathbf{f} < \mathbf{g} \).

Formally, \( \mathbf{f} \succeq \mathbf{g} \) iff:

\[
f_i \leq g_i, \forall i = 1, \ldots, m \text{ and } f_i < g_i \text{ for at least one } i
\]

Dominance in the objective space can be extended to the parameter space that is \( a \prec b \) iff \( \mathbf{f}(a) \prec \mathbf{f}(b) \).

We define the set \( \mathcal{X}_p \) of solutions not dominated by any other solution as the Pareto-optimal set while the corresponding set \( \mathcal{Y}_p = \mathbf{f}(\mathcal{X}_p) \) is called the Pareto-optimal front. The Pareto-optimal set is the actual solution of the multi-objective optimization problem.
A. Assessing the goodness of an approximate Pareto front

When proposing heuristic strategies for finding Pareto points, the issue of comparing Pareto fronts performance arises. In literature, many performance metrics have been used for evaluating the quality of the Pareto fronts with respect to reference Pareto fronts [12]–[14]. We present here the Average Distance from Reference Set and the Coverage function.

1) Average Distance from Reference Set (ADRS): The ADRS metric is used to measure the distance between a reference Pareto-optimal front R and another approximated front A:

\[
ADRS(R, A) = \frac{1}{|R|} \sum_{\vec{x} \in R} \left( \min_{\vec{a} \in A} d(\vec{x}, \vec{a}) \right)
\]

where

\[
d(\vec{x}, \vec{a}) = \max_{j=1,\ldots,m} \left\{ 0, \frac{f_j(\vec{a}) - f_j(\vec{x})}{f_j(\vec{x})} \right\}
\]

and \( m \) is the number of objective functions. ADRS is usually a normalized function which is measured in terms of percentage. The higher the ADRS, the worst is the approximated Pareto front.

2) Coverage: The coverage metric is used to measure the percentage of an approximated Pareto front \( A \) which is covered by the reference Pareto front \( R \) [15].

\[
C(R, A) = \frac{|\{\vec{a} \in A : \exists \vec{x} \in R : \vec{x} < \vec{a}\}|}{|A|}
\]

The value \( C(R, A) = 1 \) means that all the design vectors in \( A \) are dominated by the design vectors of \( R \). As opposite, \( C(R, A) = 0 \) represents the situation when none of the points in \( A \) are dominated by the set \( R \).

IV. A DESIGN SPACE EXPLORATION METHODOLOGY

In this paper, regression and interpolation methods are used to build our response surface models. Regression is a method that models the relationship between a dependent response function and some independent variables in order to describe, with the lowest error possible, both the known and unknown data. Interpolation is the process of assigning values to unknown points by using values from a small set of known points. Differently from regression, interpolation does not produce any error on the known data.

In the following, we will indicate as actual measurement a measurement performed by experimenting with the actual simulation model of the system, while with the term estimated measurement we will indicate a measurement based on a response surface model.

Let us define a design space \( S \) as a set of tuples \( s_i \). Each tuple \( s_i = (a_i, \mu_i) \) is composed of an instance of the architecture parameters \( a_i \) and the associated system metrics \( \mu_i \). Metrics \( \mu_i \) can be either derived from actual measurements \( \mu_i \) or estimated measurements \( \hat{\mu}_i \). The symbol \( S \) indicates a design space composed of tuples corresponding to actual measurements while \( S \) represents a design space of tuples corresponding to estimated measurements. We use the symbol \( P \) to represent a reduced design space corresponding to the Pareto front \( P = Pareto(S) \).

The proposed DSE methodology consists of the following steps:

1) Apply a DoE technique to pick up the set of configurations \( S_0 \) generating the plan of experiments to be run. This step provides an initial coarse view at iteration 0 of the target design space.

2) Run the experiments to obtain the actual measurements corresponding to the set of initial configurations \( S_0 \).

3) Compute the Pareto front associated with the initial design space \( S_0 \): \( PO = Pareto(S_0) \).

4) Apply a response surface technique to the Pareto front \( PO \). The response surface model generates the design space \( S_1 \) composed of a set of estimated measurements\(^2\).

5) Compute the Pareto front \( P_1 = Pareto(S_1) \).

6) Run the experiments to derive the actual measurements on the architectural configurations contained in \( P_1 \). The result is the design space \( P_1 \).

7) If \( P_1 \) covers \( P_0 \) (\( C(P_1, P_0) \)) by a percentage greater than 0 and the stopping criterion is not met, restart from step 4, where now \( P_0 \leftarrow P_1 \). The stopping criterion is the maximum number of actual measurements to be done.

The described strategy is not combined with a specific DoE but any DoE plan can be used, as it will be shown in Section IV-A. Concerning the response surface model, we target both interpolation and regression.

The proposed method assumes that the chosen RSM technique can infer the trends of the Pareto configurations in the entire design space. We try to avoid the use of the entire configuration space \( S_0 \) to infer \( S_1 \) in order to skip overfitting problems which would deteriorate the estimate \( S_1 \). The

\(^2\)Potentially, \( S_1 \) could be as large as the entire design space, however a sampling technique could be applied if it is not practically feasible to manage such a large space.
procedure strategy_interpolation (SO, maxp)
points = 0
PO = filter_pareto(SO)
hPl = train_interpolation_on(hPl)
P1 = perform_actual_measurements(hPl)
points = points + size(P1)
cov = coverage(P1, PO)
while (cov < 0.9 && points < maxp)
    PO = P1
    hPl = train_interpolation_on(hPl)
hPl = filter_pareto(hPl)
P1 = perform_actual_measurements(hPl)
points = points + size(P1)
cov = coverage(P1, PO)
done
return PO
done

procedure strategy_regression (SO, maxp)
points = 0
PO = filter_pareto(SO)
hPl = train_regression_on(hPl)
P1 = perform_actual_measurements(hPl)
points = points + size(P1)
cov = coverage(P1, PO)
while (cov < 0.9 && points < maxp)
    PO = P1
    hPl = train_regression_on(hPl)
hPl = filter_pareto(hPl)
P1 = perform_actual_measurements(hPl)
points = points + size(P1)
cov = coverage(P1, PO)
done
return PO
done

Fig. 1. Pseudo-code of the interpolation-based strategy.

Fig. 2. Pseudo-code of the regression-based strategy.

Shepard interpolation RSM, described later, fits well in this category, being possible to use it even in the case when few points are known in the original design space.

We will show later in Section IV-B1 that a highly parametric linear regression surface is needed to obtain a feasible coefficient of determination ($R^2 \geq 0.9$) on the training set. Unfortunately, this kind of regression surface needs much more training configurations than those available in the typical Pareto fronts associated with our problem. Thus we need to modify the kernel of the proposed method to include regression methods. In step 4 of the above methodology we use a sub-sampling of the configuration space $S_0$ (instead of $P_0$) to train the initial configuration space $S_1$. As before, at steps 5 and 6, each configuration of the Pareto front $P_1$ is then actually measured and the resulting $P_2$ is compared with the original $P_0$. To compute the next $S_0$, we merge the newly created Pareto front $P_1$ to the original $S_0$ configuration space. The remaining part of the algorithm is the same as described above. Figure 1 shows the pseudo-code of the original, interpolation-based technique while Figure 2 shows the regression based technique.

In the following, we show the DoE techniques as well as the RSM methods that can be used in the proposed design space exploration methodology. In the experimental results, we show how the performance characteristics of each strategy can vary with respect to the selected DoE and RSM.

A. Design of Experiments

Although several design of experiments have been proposed in the literature so far, we present here the most traditional DoE which we will leverage in the construction of our efficient design space exploration methodology.

1) Random: In this case, design space configurations are picked up randomly by following a Probability Density Function (PDF). In our methodology, we will use a uniformly distributed PDF.

2) Full factorial: In statistics, a factorial experiment is an experiment whose design consists of two or more parameters, each with discrete possible values or "levels", and whose experimental units take on all possible combinations of these levels across all such parameters. Such an experiment allows studying the effects of each parameter on the response variable, as well as the effects of interactions between parameters on the response variable. In this paper, we consider a 2-level full factorial DoE, where the only levels considered are the minimum and maximum for each parameter. If, for example, we have only two tunable parameters i.e., cache size and associativity, where $size \in \{2K, 4K, 8K, 16K, 32K\}$ and $associativity \in \{2, 4, 8\}$, then a 2-level full factorial design will be composed by the following design vectors $(size, associativity) \in \{(2K, 2), (2K, 8), (32K, 2), (32K, 8)\}$.

If the number of experiments for a full factorial design is too high to be feasible, a fractional factorial design can be performed in which some of the possible combinations are omitted. An analysis of the use of fractional factorial design is however out of the scope of this paper and it will be investigated in our future work.

A factorial experiment is usually analyzed by means of ANOVA (Analysis of Variance) [16] techniques, and, as in our case, it is frequently used as an input to a first or second order linear response surface model. We will present regression techniques in the section devoted to the response surface methodologies.

3) Central composite design: A central composite design is an experimental design specifically targeted to the construction of response surfaces of the second order (quadratic) without requiring a three-level full or fractional factorial DoE. The design consists of three distinct sets of experimental runs:

- A 2-level full or fractional factorial design;
- A set of center points, i.e. experimental runs whose values of each parameter are the medians of the values used in the factorial portion;
- A set of axial points, i.e., experimental runs identical to the center points except for one parameter. In the general central composite design, this parameter will take
on values both below and above the median of the two levels. In this paper, we will use a face centered central composite design where the remaining parameter assumes the upper and lower bounds of the parameter range. Considering the previous example (cache size and associativity), a face centered central composite design will be composed of the following design vectors:

\[
\{ (2K, 2), (2K, 8), (32K, 2), (32K, 8), \\
(8K, 4), (2K, 4), (32K, 4), (8K, 2), (8K, 8) \}
\]

4) Box-Behnken: The Box-Behnken design is suitable for quadratic models where parameter combinations are at the center of the edges of the process space plus a design with all the parameters at the center. The primary advantage is that the parameter combinations avoid extreme values taken at the same time (in contrast with the central composite design). This may be suitable to avoid singular points in the generation of the response surface, which would deteriorate it. Considering the previous example (cache size and associativity), a Box-Behnken design is composed of the following design vectors:

\[
\{ (8K, 4), (2K, 4), (32K, 4), (8K, 2), (8K, 8) \}
\]

B. Response Surface Methods

We present here the RSMs used for the construction of our efficient design space exploration methodology.

1) Linear regression: Linear regression is a regression method that models a linear relationship between a dependent response function \( f \) and some independent variables \( x_i \), \( i = 1 \cdots p \) plus a random term \( \varepsilon \). In this work we apply regression by taking into account also the interaction between the parameters as well as quadratic behavior with respect to a single parameter. We thus consider the following general model:

\[
f(\vec{x}) = \alpha_0 + \sum_{k=1}^{p} \alpha_k x_k^2 + \sum_{i=1}^{p} \sum_{k=1, k \neq i}^{p} \beta_{i,k} x_i x_k + \sum_{k=1}^{p} \gamma_k x_k + \varepsilon
\]

Least squares analysis can be used to determine a suitable estimate for the parameters. Ordinary least squares determines the values of unknown quantities in a statistical model by minimizing the sum of the squared residuals (the difference between the predicted and observed values). A measure of the quality of fit associated with the resulting model is called coefficient of determination and defined as follows:

\[
R^2 = \frac{SSR}{SST}
\]

where

\[
SST = \sum_i (y_i - \bar{y})^2
\]

is the total sum of squares of observations \( y_i \) and average observations \( \bar{y} \), and

\[
SSR = \sum_i (f_i - \bar{y})^2
\]

is the regression sum of squares between the model estimates \( f_i \) and the average of observations.

As a rule of thumb, the higher \( R^2 \), the better the model fits the data. An \( R^2 \) equal to 1.0 indicates that the regression line perfectly fits the data.

To use the linear regression as RSM, a correct configuration of the model must be identified. First of all, a numerical value for each symbolic value of the parameters is chosen. This value will be used directly in the computation of the regression function replacing the symbolic value of the parameters. As an example, a 'plain' encoding for a cache size parameter could be \{"2K","4K","8K"\} \( \rightarrow \{1,2,3\} \). An 'exponential' coding would produce, for the same sequence of symbolic values \{"2K","4K","8K"\} \( \rightarrow \{2^1,2^2,2^3\} \) note that this encoding is proportional to the real size of the cache. Other type of encoding include clustering around the minimum and maximum value for using only two encoding values (namely \{-1,1\} or \{0,1\}) as an example, a "two-zero" encoding would encode the minimum and the maximum to \{0,1\} and all the remaining values to the nearest boundary value. For the previous cache example we have that \{"2K","4K","8K"\} \( \rightarrow \{0,0,1\} \).

Both simple and quadratic forms of the linear regression model should be analyzed including a test of the following configurations of the model:

- **Heavy**: All the parameters of the design space are included in the model.
- **Medium**: All the parameters except for the most irrelevant are used.
- **Light**: Only the most relevant parameters are used.

To select which parameters are relevant and which are not, a main effect analysis [16] on a random subset of the design space configurations can be performed.

By running the previous analysis on a subset of configurations of the entire design space, we found out that including interaction effects is a 'must' for our RSM design. Figure 3 shows, among the models including the interaction effects, the coefficient of determination of the best models we found. The quadratic, plain encoding and heavy parameter model is the one which presented the best coefficient of determination. We will use this model configuration for the experimental results. We then measured the effect of output transformations on the
accuracy of linear models. The preprocessing function transforms the response values before being fed to the linear model training in order to minimize the error. We considered a family of transformations as potential candidates \( \{y^1, y^{0.5}, \log(y)\} \) and \( y^{-1} \). For each transformation, we selected a set of random configurations as input to the linear regression model and computed the maximum normalized error on the training set. Figure 4 shows that the best behaving output transformation is the \( \log(y) \) function. We will stick with this preprocessing function for the remaining part of the simulation results.

2) Shepard’s based Interpolation: Interpolation is the process of assigning values to unknown points by using values from a small set of known points. In this paper we use the Shepard’s inverse distance weighting (IDW) technique as a method for multivariate interpolation. In this technique, the interpolated value of a response function \( f \) for an unknown design \( \vec{x} \) is computed by using \( N + 1 \) known design points \( \vec{x}_k \) as follows:

\[
f(\vec{x}) = \frac{\sum_{k=0}^{N} w_k(\vec{x}) f(\vec{x}_k)}{\sum_{k=0}^{N} w_k(\vec{x})} \tag{11}
\]

where:

\[
w_k(\vec{x}) = \frac{1}{\mu(\vec{x}, \vec{x}_k)^p}, \tag{12}
\]

is the IDW weighting function defined by Shepard, \( p \) is the power of the model and \( \mu \) is the distance between the known point \( \vec{x}_k \) and the unknown \( \vec{x} \). For computing the power coefficient \( p \), we selected a set of random configurations of size similar to those of a Pareto front (\( \leq 300 \)) as input to the interpolation model and computed the maximum normalized error on the the prediction of a small design space centered around the input Pareto front. Figure 5 shows that the best behaving output power value is 5. We will use this value for the remaining part of the simulation results.

V. EXPERIMENTAL RESULTS

To validate the methodology, an experimental optimization campaign has been setup for a symmetric shared-memory multi-processor architecture. The simulation tool we used is Sesc [17], a fast simulator for chip-multiprocessor architectures with out-of-order processors that is able to provide energy and performance results for a given application.

The entire optimization process has been carried out over a set of four applications derived from the SPLASH-2 [18] benchmark suite \( (U = \{FFT, OCEAN, LU, RADIX\}) \) with 3 different data-sets for each application. We focused the analysis on the architectural parameters listed in Table I, where the minimum and maximum values have been reported. The resulting design space consists of \( 2^{17} \) configurations.

For each application, we gathered information about the system response in terms of average execution time and energy consumption. However, we formulated the multi-objective problem as a ‘robust’ problem consisting of finding a system configuration which minimizes the geometric average \(^3\) value of the system response in each application scenario \( \xi_k \in U \):

\[
\min_{\vec{x} \in \mathcal{X}} E[f(\vec{x}, \xi)] \tag{13}
\]

We formalize our multi objective problem as a minimization of the average execution time and mW/MIPS system response, over the set of twelve \( \xi_k \in U \) different scenarios:

\[
\min_{\vec{x} \in \mathcal{X}} \frac{\text{avg\_time\_ums}(\vec{x})}{\text{avg\_mW\_per\_MIPS}(\vec{x})} \tag{14}
\]

where:

\[
\text{avg\_time\_ums}(\vec{x}) = \prod_{\xi_k \in U} \text{time\_ums}(\vec{x}, \xi_k) \tag{15}
\]

\(^3\) A geometric mean tends to dampen the effect of very high or low values, which might bias the mean if a straight average (arithmetic mean) were calculated.

12

![Fig. 4. Maximum Normalized Error of the linear regression on the training set by varying number of input configurations.](image)

![Fig. 5. Maximum Normalized Error of the Shepard interpolation on the overall design space for 300 samples of training set, by varying the power coefficient.](image)

**Table I**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min.</th>
<th>Max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 instruction cache size</td>
<td>2K</td>
<td>16K</td>
</tr>
<tr>
<td>L1 data cache size</td>
<td>2K</td>
<td>16K</td>
</tr>
<tr>
<td>L2 shared cache size</td>
<td>32K</td>
<td>256K</td>
</tr>
<tr>
<td>L1 instruction cache assoc.</td>
<td>8</td>
<td>8w</td>
</tr>
<tr>
<td>L1 data cache assoc.</td>
<td>8</td>
<td>8w</td>
</tr>
<tr>
<td>Processor issue width</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>I/D/L2 block size</td>
<td>16</td>
<td>32</td>
</tr>
<tr>
<td># Processors</td>
<td>2</td>
<td>16</td>
</tr>
</tbody>
</table>

![Geometric mean](image)

![Shepard's based Interpolation](image)

![Presprocessing function](image)

![Figure 4](image)

![Figure 5](image)
A. Setup of the experimental campaign

Using the Shepard’s method and the linear regression-based Pareto front of the geometric averages which represent the actual objective functions of the problem. The complete set of implementations is shown in Table II.

![Table II: Evaluated Strategies](image)

<table>
<thead>
<tr>
<th>Strategy implementation name</th>
<th>DoE</th>
<th>RSM</th>
</tr>
</thead>
<tbody>
<tr>
<td>full-factorial-plain</td>
<td>Full-factorial</td>
<td>-</td>
</tr>
<tr>
<td>full-factorial-shepard</td>
<td>Full-factorial</td>
<td>Shepard’s method</td>
</tr>
<tr>
<td>full-factorial-linear</td>
<td>Full-factorial</td>
<td>Linear regression</td>
</tr>
<tr>
<td>random-plain</td>
<td>Random</td>
<td>-</td>
</tr>
<tr>
<td>random-shepard</td>
<td>Random</td>
<td>Shepard’s method</td>
</tr>
<tr>
<td>random-linear</td>
<td>Random</td>
<td>Linear regression</td>
</tr>
<tr>
<td>central-composite-plain</td>
<td>Face centered CCD</td>
<td>-</td>
</tr>
<tr>
<td>central-composite-shepard</td>
<td>Face centered CCD</td>
<td>Shepard’s method</td>
</tr>
<tr>
<td>central-composite-linear</td>
<td>Face centered CCD</td>
<td>Linear regression</td>
</tr>
<tr>
<td>box-behnken-plain</td>
<td>Box-Behnken</td>
<td>-</td>
</tr>
<tr>
<td>box-behnken-shepard</td>
<td>Box-Behnken</td>
<td>Shepard’s method</td>
</tr>
<tr>
<td>box-behnken-linear</td>
<td>Box-Behnken</td>
<td>Linear regression</td>
</tr>
</tbody>
</table>

B. Evaluation of the proposed strategies

Both strategies have been applied to obtain an approximated strategy by using the results provided in the previous section. The considered strategies have been run until a maximum number of actual measurements associated with it, while 7 (b) plots the actual Pareto points miss-rate and the number of actual measurements associated with each strategy. The actual Pareto points miss-rate is the percentage of points of the actual Pareto front not identified by the specific strategy.

In order to give a fair comparison with other methods proposed in the literature we introduced the results of a Multi-Objective Simulated Annealing (MOSA) [19] applied to the same design space. The MOSA algorithm has been run several times with different configuration values (number of epochs, size of each epoch and temperature decrease coefficient); for a fixed number of actual measurements, the one with the best ADRS has been shown.

Considering the ADRS, actual Pareto miss rate and number of actual measurements objectives, the dominant solutions we can identify are the following: 'box-benken-linear', 'random-linear', 'full-factorial-shepard', 'mosa_25_10_05' and 'mosa_20_100_05'. The strategies proposed in this paper show a very small number of actual measurements, confirming their efficiency features. The 'mosa_25_10_05' also shows a low number of actual measurements, however it is characterized by an high value of Pareto miss-rate.

More in detail, Figure 7(a) suggests that strategies based on the 'linear regression' are characterized by a number of actual measurement less than 350, while 'shepard'-based strategies are between 300 and 600. Finally, simulated annealing solutions show a slight decreasing trend of ADRS at the expense of an increasing number of actual measurements.

Figure 7(b) shows that 'linear regression'-based approaches have a higher miss-rate but present the lowest number of actual measurements. A central role is played by the 'full-factorial-shepard' strategy, with a good trade-off between the two performance measures. Finally, the 'mosa_20_100_05' shows a very good miss-rate combined with a high number of actual measurements.

C. Final considerations

As a final statement, we can conclude that both regression and interpolation strategies are the same face of a flexible optimization flow which allows the designer to trade-off accuracy and efficiency, in terms number of actual measurements. We can suggest, among all the proposed strategies, two guiding strategies which could be adopted depending on the specific designer’s goal:

- **Low-end strategy.** The 'random-linear-regression' can be adopted when only an coarse grain approximation of the Pareto front is needed, with very few actual measurements.
The presented set of experimental results have shown how view of the target design space; then a set of response surface (DoE) paradigm and a set of Response Surface Modeling extraction techniques are used to refine the design space an initial plan of experiments which are used to create a coarse different response surface methodologies. This process is repeated iteratively until a target criterion, e.g. number of simulations, is satisfied. The proposed methodology is accurate and flexible in terms of multi-objective simulated annealing. The Design and Analysis of Computer Experiments. Springer-Verlag, 2003.


