Evaluation of Design Space Exploration Strategies*

Francisco Moya[†], José M. Moya[‡], Juan C. López[†]

[†]Universidad de Castilla-La Mancha
Departamento de Informática
ESI. P° Universidad 4, 13071 Ciudad Read, Spain
{fmoya,lopez}@inf-cr.uclm.es

[‡]Universidad Politécnica de Madrid
Departamento de Ingeniería Electrónica
ETSIT, Ciudad Universitaria, 28040 Madrid, Spain
josem@die.upm.es

Abstract

The design space exploration (DSE) subsystem of a design automation tool is responsible for early identification of interesting zones of the design space.

Evaluation of the exploration strategy used in a tool is extremely difficult because there is not enough information about either the design space or the actual cost function. This paper describes a software environment for quantitative evaluation of isolated exploration algorithms based on a completely simulated framework. A simple model of the design space allows easy representation of interesting properties of exploration algorithms.

1. Introduction

Design space exploration (DSE) is a major component of many design automation tools. Exploration techniques are aimed at effectively pruning the design space and identifying the most potentially interesting zones to be developed afterwards.

A number of methodologies [1, 5, 6, 13] have been proposed for effective design space exploration in system synthesis at RT and behavioural levels. Tools for these higher abstraction levels, unlike physical design tools, require heuristics to guide the exploration towards a specific direction. Some researchers refer to exact design space exploration schemes but they rely on simplified formulations of the problem. For example, Chaudhuri et al. [6] provide an ILP formulation for simultaneous hardware scheduling and clock length assignment, but they do not consider loops or branches in the specification. This is equivalent to heuristically prune the design space in advance.

As a consequence of using heuristics, a certain amount of uncertainty is introduced in the exploration, which is closely related to the accuracy of the estimation. Reported results for behavioural level estimations [5, 14] show errors below 25% for small HLS benchmarks [7]. These errors may slightly affect the quality of the final solution but final synthesis stages may easily refine the solution identified by the estimations, therefore, a near-optimal result is very likely.

System level design automation tools add a much higher degree of uncertainty in predictive heuristics. This is due to unavailability of enough detailed information on the hardware and software structure during the early phases of the design process. A major contribution to this uncertainty is the cost associated to communication among the components of the design, which may easily surpass the cost of the functional components. Besides that, accurate estimation is even more important when the design process is started at a higher level of abstraction. Decisions made during initial phases of system level synthesis determine up to 80% of the final cost [13].

Some researchers (e.g. [14]) advocate more accurate models of design components. This is not always possible at the system level since there is not enough information available during early design stages. Accurate models require at the very least enough structural information to be able to predict communication cost.

The *Design Space Exploration* (DSE) subsystem of a design automation tool must be designed from the ground up to cope with uncertainties. Although traditional combinatorial optimization algorithms may be used for DSE, they will not perform as expected due to inaccurate estimations.

This paper describes a framework for quantitative evaluation of exploration algorithms. Properties such as design space coverage or the ability to recover from bad decisions may easily be analyzed in a problem-independent fashion. The paper is organized as follows. Section 2 describes the main components of a DSE module. Section 3 describes our environment focused on evaluation of one of those com-

*This work was done while the authors were with Universidad Politécnica de Madrid, and supported by the Spanish Research Foundation (CICYT) through the project HADA TIC97-0928.

ponents: exploration algorithms. We provide examples for two frequently used algorithms: *tabu search* and *simulated annealing*. Finally, section 4 presents a simple exploration strategy to increase effectiveness of traditional optimization algorithms when they are used for system-level exploration.

2. DSE subsystem

Identification of interesting areas of the design space relying on inaccurate heuristics is a completely different goal from that of simple combinatorial optimization. We identify three major areas of interest in a DSE subsystem:

- Design space representation.
- Metrics and estimations.
- Exploration algorithms.

Most of the research efforts on design space exploration for system level design automation has been oriented towards formal design space representation [10, 3]. They use hierarchical decomposition of the design space to allow a systematic exploration based on relevant figures of merit. The structure added to the design space specifies a partial order for the exploration process that matches the sequence of levels in the hierarchy. In some way design space representation constitutes an *a-priori* heuristic design-independent exploration that simplifies later decisions.

Another major research topic is the set of metrics and estimations used for assessment of candidate solutions. Traditional estimations for hardware-only systems involve structural information [1, 14], while estimations of software-only systems are usually based on statistical data collected from behavioural specifications ([4], part 4A). System-level design automation tools may either combine the two approaches [10] or apply only behavioural metrics [5]. In both cases the high abstraction of the system-level specifications introduce a large degree of uncertainty.

The last main component of the DSE subsystem is the exploration strategy, the algorithm or set of algorithms responsible for the actual exploration. RT-level and logic-level tools usually apply conventional combinatorial optimization techniques. For higher level environments previous research on early exploration considered implicit exploration strategies induced by the set of estimations and the hierarchical design space. We will show that algorithms implementing explicit strategies allow easier evaluation and comparisons among different exploration techniques, and they also allow finer control over the global behaviour of the synthesis tool.

A complete design space exploration methodology is proposed in [11]. The authors rely on simple linear estimators and traditional multi-objective optimization, with focus on linear programming and gradient methods. Linear programming methods do not scale well for a high number of decision variables. Besides that, a hierarchical decomposition of the design space as proposed in [10] (specially adequate for IP-based design) cannot guarantee the required orthogonality among decision variables. Finally, discontinuities in the cost function may affect the efficacy of gradient methods.

By contrast with linear programming formulations, heuristic strategies allow arbitrary non-linear estimations, arbitrary design space decomposition and better execution time for huge design spaces. We will focus on this kind of strategies in this article.

3. DSE analysis environment

Evaluation of a particular exploration strategy implemented in a CAD tool is extremely difficult since the analytical expression of the cost function is unknown. There is no way to infer whether the explored points of the design space are representative of the whole design space, or the exploration was actually reduced to a local search in a particular sub-optimal zone of the design space. We overcome those limitations with a simulated environment which provides both a completely known search space and the analytical expression of the cost function. This environment tries to model only those characteristics of the design space affecting the exploration process.

In order to be able to analyze a set of explicit exploration strategies, these strategies should be implemented using an abstract interface to interact with the rest of the DSE subsystem. The evaluation environment supplies a simulated replacement for those additional DSE components (design space structure and estimations).

For our tests we used a two-dimensional design space that simplifies data representation. Each point in the region $[-1, 1] \times [-1, 1]$ of the real plane corresponds to the particular design being considered. It might be argued that this is an extremely simplistic model for the design space. We will show that even such a simple model is useful to analyze some properties of the exploration algorithms. Ability to deal with complex design spaces is usually related to the formal design space representation, not to the exploration strategy.

3.1. Cost function

The cost function is explicitly specified by the user. For our examples we chose a derived form of a common test function for combinatorial optimization algorithms [2]:

$$f(x, y) = 0.5 + \frac{\sin^2(100\sqrt{x^2 + y^2}) - 0.5}{(1 + 10(x^2 + y^2))^2}$$

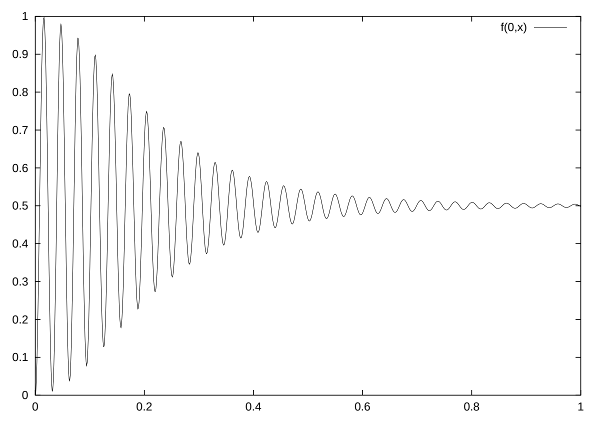

Figure 1. Cross section of selected cost function $f(x, y)$ for $y = 0$.

in the domain $[-1, 1] \times [-1, 1]$. The global minimum is at the origin but it is surrounded by many local optima located on concentric circles (see fig. 1).

We introduced huge discontinuities near the local optima to model actual discontinuities in the cost function of system-level tools. A simple modification to $f(x, y)$ is enough for our experiments:

$$g(x,y) = \begin{cases} 0.5 + \frac{\sin^2(100\sqrt{x^2+y^2})-0.5}{(1+10(x^2+y^2))^2} & \text{if } |x| \geq |y| \\ 1 & \text{otherwise} \end{cases}$$

Presence of this kind of discontinuities in system-level tools is easily shown with an example. Consider partitioning a design into software for some micro-controller and an FPGA implementing critical functionality. Design automation tools should maximize usage of the FPGA to avoid wasting the resources of the FPGA. On the other side if the tool allocates too much functionality on the FPGA we'll need to use two FPGAs (or a more expensive one) roughly doubling the cost of the system. A small variation in the design produces a huge increment in the total cost. Besides that, design automation tools are explicitly looking for these discontinuity points (maximizing usage of allocated resources). Our experiments show that convergence of some combinatorial optimization algorithms is heavily affected by these discontinuities (fig. 2), therefore it should also be considered in the model for exploration strategies.

The plots of figure 2 were generated from data produced by a single run of two frequently used combinatorial optimization algorithms: *simulated annealing*[12] and *tabu search*[8]. We use the same procedure for design space quantization and encoding described in [2] for both algorithms. Our implementation of simulated annealing follows the cooling scheme proposed in [9]. The implemented tabu search strategy limits the size of the tabu list to a fixed quantity (as proposed in [8]) but we also introduce a small quantity of random noise (0.1% of the full scale) to avoid falling into loops of already visited local minima. We will discuss this in the following subsection.

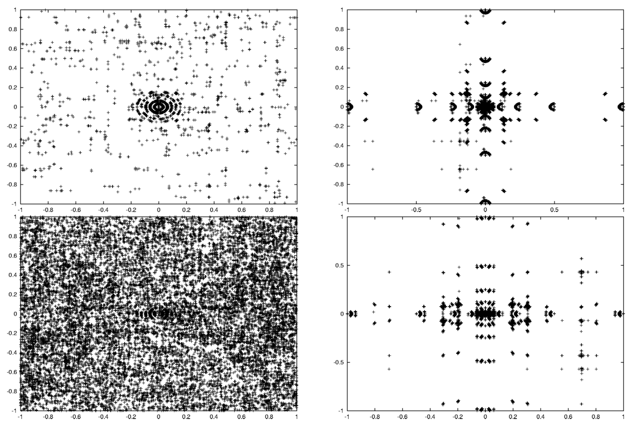

Figure 2. Explored design space for two optimization algorithms with cost function $g(x, y)$. The left side of the figure corresponds to *simulated annealing*, and the right side to *tabu search*. The upper plots represent the exploration for a continuous cost function $f(x, y)$, while the lower plots show the impact of discontinuities in the cost function. Tabu search is almost unaffected.

It is also possible to generate a more meaningful color plot. The color scale is used to encode the normalized frequency of visited regions (each point and its neighborhood) across multiple runs of the exploration algorithms. For our examples the results are very similar to what is already represented in figure 2. Color images for the examples in this paper may be found along the source code (see section 6).

The cooling scheme we used for simulated annealing is considered to be very robust. This claim is supported by the fact that the algorithm was almost always able to find the global minimum at the origin. On the other side, for a DSE subsystem, it is more important the identification of interesting zones than actually finding the global optimum (uncertainty makes impossible to ensure a global optimum). Figure 2 shows that simulated annealing performs almost a random search across the entire design space in presence of huge discontinuities.

3.2. Uncertainties

A fundamental difference between traditional combinatorial optimization algorithms and system-level synthesis is that the cost function is unknown during the early phases

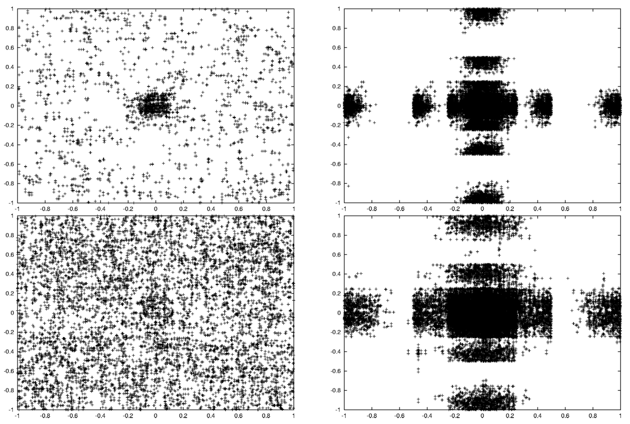

Figure 3. Explored design space for two optimization algorithms with cost function $f(x, y)$ (no discontinuities). The left side of the figure corresponds to *simulated annealing*, **and the right side to** *tabu search*. **The upper plots represent the exploration for 5% error in cost function evaluation, while the lower plots show the impact of 10% error in the cost function.**

of the exploration. The design space exploration subsystem must provide two kinds of heuristics: predictive heuristics for cost function approximation, and exploration heuristics to guide the exploration. Our simulated environment is only focused on evaluation of exploration heuristics, but uncertainties in the cost function must be introduced in the model in order to realistically predict exploration efficacy.

We consider two causes of uncertainty during cost function evaluation: design indetermination and cost indetermination. The lack of precise knowledge of the location of the design being evaluated in the whole design space is due to incompleteness of the design. An early design may actually be refined by later stages into hundreds of alternatives. For our examples we modeled the design uncertainty as a random noise added to the coordinates of the point being considered.

As shown in figure 3 introduction of noise in cost function evaluation (to both the coordinate and the cost function value) also have a major impact in the efficacy of the exploration. Adding random noise for at most 10% of the full scale our implementation of simulated annealing degenerates into almost a random search (even without discontinuities in the cost function). The percentage of tolerable noise for each algorithm is not meaningful by itself because it depends on the cost function considered. Anyway it is still useful in order to compare the robustness of a set of algorithms.

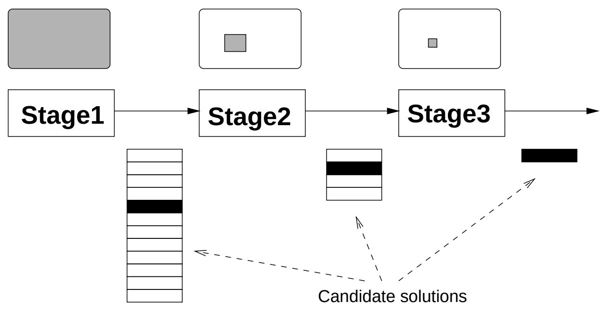

Figure 4. We model a CAD tool as a cascade of combinatorial optimization algorithms focused around the interesting points identified by previous stages.

3.3. Model for the tool

We modeled the complete design automation tool-set by means of a cascade of combinatorial optimization algorithms as shown in fig. 4. Each stage introduce much less uncertainty during the cost function evaluation than the previous stage. Also, each stage explores only a limited zone of the design space around the interesting points provided by the previous stage.

The way in which those stages are connected to each other depends entirely on the DSE strategy. In the following section we will describe a simple strategy that requires only small modifications of traditional combinatorial optimization algorithms.

4. A simple DSE strategy

Combinatorial optimization algorithms may easily identify interesting zones of the design space in each stage of the tool model. The best solution may be representative of an interesting zone but other good solutions should also be considered for further exploration. Instead of keeping only the best solution the algorithms must be modified to keep the best n solutions. This turned out to be a trivial change of less than ten lines of code. The number of solutions kept for further exploration (n) is related to the uncertainty in the current stage. For the examples shown in figure 5 we used three stages of simulated annealing with a list of 25, 5 and 1 best solutions respectively.

Another global exploration algorithm is responsible for guidance through the set of available solutions towards the global optimum. We used a simple backtracking algorithm.

Having a completely simulated environment allows us to know the actual value of the cost function at any moment. This is useful to represent the evolution of the exploration strategy as shown in figure 5.

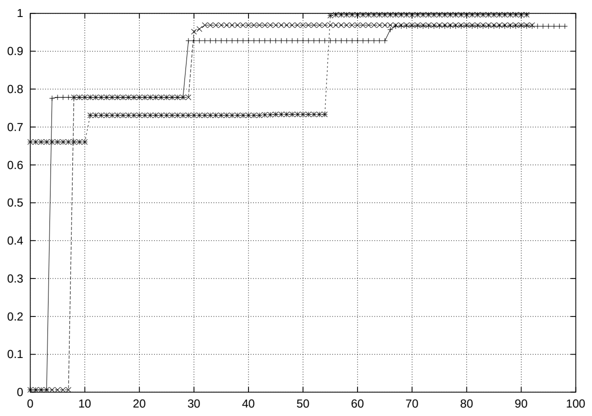

Figure 5. Evolution of the actual value of the *best-so-far* solution for several runs of the exploration strategy described in section 4. The global optimum value is 1.0.

Although the DSE strategy proposed in this section is highly effective there are a lot of possible enhancements:

- The number of best solutions kept for each stage may be controlled dynamically with criteria other than the cost function value. For example, we may want to preserve the variety in a set of candidate solutions by collapsing similar solutions into just one solution.

- Instead of simple backtracking a *branch-and-bound* approach may be used in order to backtrack as soon as a very bad solution is detected.

5. Conclusions and future work

The quality of the final design is closely related to the effectiveness of the design space exploration subsystem. Besides that, *time-to-market* is becoming an increasingly important factor in complex system design. Both efficacy and efficiency of an exploration strategy must be analyzed to evaluate the quality of a design. To our knowledge there is no previous work addressing the needs of exploration strategy analysis as another factor affecting the quality of the design process. We described a framework that complements the methodology described in [11] for easy evaluation of exploration strategies by providing a simulated model of the design automation tool and associated environment. We do not intend to offer a realistic model. Our simple models allow for tradeoff evaluation among several exploration strategies and will be extended when the need arises.

Although we have focused on problem-independent exploration algorithms problem-specific exploration strategies may also be mapped into a simplified models. We are developing visual representation techniques for these situations where a simple 2D design space may not be suitable.

Evaluation of DSE subsystems should also consider evaluation of structural decompositions of the design space. This could be achieved by means of specialized metrics.

6. Source code

The source code and data files mentioned in this article may be found in the URL `http://www.inf-cr.uclm.es/www/fmoya/DSE/`

References

[1] S. Bakshi and D. D. Gajski. Design Exploration for High-Performance Pipelines. In *Proc. IEEE/ACM Conf. on Computer-Aided Design*, 1994.

[2] R. Battiti and G. Tecchiolli. The Reactive Tabu Search. *ORSA Journal on Computing*, 6(2), 1994.

[3] O. Bentz, J. M. Rabaey, and D. Lidsky. A Dynamic Design Estimation and Exploration Environment. In *Proc. ACM/IEEE Design Automation Conf.*, 1997.

[4] B. W. Boehm. *Software Engineering Economics*. Prentice-Hall, 1981.

[5] A. Bogliolo, L. Benini, and G. D. Micheli. Adaptive Least Mean Square Behavioral Power Modeling. In *Proc. European Design and Test Conference*, 1997.

[6] S. Chaudhuri, S. A. Blythe, and R. A. Walker. A Solution Methodology for Exact Design Space Exploration in a Three-Dimensional Design Space. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 5(1), 1997.

[7] N. Dutt and C. Ramachandran. Benchmarks for the 1992 High Level Synthesis Workshop. Technical Report 92-107, Information and Computer Science Department, U.C. Irvine, 1992.

[8] F. Glover. Tabu search - part I. *ORSA Journal of Computing*, 1(3), 1989.

[9] M. D. Huang and A. Sangiovanni-Vincentelli. An Efficient General Cooling Schedule for Simulated Annealing. In *Proc. of ICCAD*, 1986.

[10] M. F. Jacome and J. C. López. Supporting Early System-Level Design Space Exploration in the Deep Submicron Era. In J. C. López, R. Hermida, and W. Geisselhardt, editors, *Advanced Techniques for Embedded Systems Design and Test*. Kluwer Academic Publishing, 1998.

[11] L. Jóźwiack and S. A. Ong. Quality-Driven Decision Making Methodology for System-Level Design. In *Proceedings of the 22nd EUROMICRO Conference*, 1996.

[12] S. Kirkpatrick, C. D. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220, 1983.

[13] R. H. Klenke, M. Meyassed, J. H. Aylor, B. W. Johnson, R. Rao, and A. Ghosh. An Integrated Design Environment for Performance and Dependability Analysis. In *Proc. ACM/IEEE Design Automation Conf.*, 1997.

[14] S. Y. Ohm, F. J. Kurdahi, N. Dutt, and M. Xu. A Comprehensive Estimation Technique for High-Level Synthesis. In *Proc. IEEE International Symposium on System Synthesis*, 1995.