Discrete Particle Swarm Optimization for Multi-objective Design Space Exploration

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

Abstract

Platform-based design represents the most widely used approach to design System-On-Chip (SOC) applications. In this context, the Design Space Exploration (DSE) phase consists of optimally configure a parameterized SOC platform in terms of system-level requirements depending on the target application. In this paper, we introduce the Discrete Particle Swarm Optimization methodology (DPSO) for supporting the DSE of an hardware platform. The proposed technique aims at efficiently profiling the target application and deriving an approximated Pareto set of system configurations with respect to the selected figures of merit. Once the approximated Pareto set has been built, the designer can quickly select the best system configuration satisfying the constraints. Experimental results show that the proposed DPSO technique can speed up the design space exploration time up to 5X with an accuracy of up to 70% with respect to a full search exploration for the selected benchmarks[1].

1 Introduction

Currently, *platform-based design* represents the de-facto approach to the design of modern System-On-Chip (SOC) applications [9]. In this approach, a parameterized embedded SOC architecture is optimally tuned to find the best trade-offs in terms of the selected figures of merit (e.g. energy, delay or area) for the given class of applications. This tuning process is called the *Design Space Exploration (DSE)* task. The overall goal of the DSE task consists of optimally configure the parameterized SOC platform in terms of system-level requirements depending on the given application.

In this paper, we propose a Discrete Particle Swarm Optimization (DPSO) heuristic for supporting the DSE phase. In particular, the approach mainly represents an extension and adaptation of existing techniques based on Particle Swarm Optimization [8] to the problem of the efficient architectural exploration of an hardware platform. To the best of our knowledge, the approach proposed in this paper represents the first attempt for applying a methodology based on Particle Swarm to the domain of Design Space Exploration of embedded applications.

The paper is organized as follows. A review of the most significant works appeared in literature concerning the DSE problem is reported in Section 2. The proposed DPSO is presented in Section 3, while Section 4 discusses some experimental results carried out to evaluate the efficiency with respect to accuracy. Finally, some concluding remarks have been reported in Section 5.

2 Background

The most trivial approach to determine the Pareto-optimal configurations in a large design space consists of the analysis of all feasible configurations. However, when the design space is too large, heuristic methods must be adopted to find acceptable near-optimal solutions.

Platune [7] is an optimization framework that introduces the concept of parameter independence to derive approximate Pareto curves without performing the exhaustive search over the whole design space. More recently in [12] and [1] Platune has been extended by applying genetic algorithms to optimize dependent parameters, resorting to the default Platune policy when independent parameters are specified by the user. The work in [3] proposes to use domain knowledge derived from the platform architecture to design design space exploration as a decision problem. Each action in the decision-theoretic framework corresponds to a change in the platform parameters. Exploration is modeled as a Markov Decision Process (MDP), and the solution to such MDP corresponds to the sequence of transformations to be applied to the platform to maximize a certain value function. Finding a solution to the problem requires to simulate the system only in particular cases of uncertainty, massively reducing the simulation time needed to perform the exploration of a system, while maintaining near-optimality of the results.

The paper in [4] introduces an interface specification language (PISA) that allows to separate the problem-specific part of an optimizer from the problem-independent part. This approach makes it possible to specify and implement representation-independent selection modules, which form the essence of modern multi-objective optimization

[1]This work was supported in part by the EC under grant for FP7-216693-MULTICUBE Project.

algorithms. Recently, the authors of [14] have extended the concept of design space exploration to include multi-dimensional Pareto sets of mappings per application.

In our previos works [11, 10], Pareto Simulated Annealing and the Random Search Pareto have been proposed to evaluate energy-delay tradeoffs for a set of multimedia kernels.

3 Proposed Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a heuristic search methodology that tries to mimic the movements of a flock of birds aiming at finding food [8]. PSO is based on a population of particles flying through an hyper-dimensional search space. Each particle possesses a position and a velocity; both variables are changed to emulate the social-psychological tendency to mimic the success of other individuals in the population (also called *swarm*). More formally, the position for particle i is changed by adding the velocity vector to the current position:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t)$$

while the velocity vector is updated with the following rule:

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

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 $[0, 1]$, $\vec{x}_{pbest_i}$ is the best position of particle i with respect to the minimization problem, $\vec{x}_{gbest}$ is the global best found so far. As can be noted, the formulation of the problem leads to solutions which try to 'follow' the leader's x_{gbest} position as well as attracting solutions versus the personal best solution of the particle $\vec{x}_{pbest_i}$.

So far, several approaches have been proposed for extending the formulation of the PSO technique to the multi-objective domain [13, 2]. Here we propose a technique 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 $\{f_1 \dots f_m\}$, each sub-swarm i solves the following problem:

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

where $p_{i,j}$ is a set of randomly chosen exponents. It can be shown that solutions to Problem 1 lie on the Pareto surface of the original problem. The approach presented in [2] uses, instead, a linear combination of cost functions $\{f_1 \dots f_m\}$ which, however, can be heavily biased on highly valued cost-functions disregarding low-valued ones.

The essential nature of the solution space of the problem we want to solve is discrete, while the approaches presented so far deal with a continuous search space.

We propose a Discrete Particle Swarm paradigm 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.

In our algorithm, the position of the particle is still updated with the traditional rule:

$$\vec{x}_i(t) = \vec{x}_i(t-1) + \vec{v}_i(t) \tag{2}$$

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

$$v_{i,k}(t) = \begin{cases} \text{sign}(x_{gbest,k} - x_{i,k}(t-1)) & \text{if}(\text{rand}() < p) \\ \text{randint}(-1, 1) & \text{otherwise} \end{cases} \tag{3}$$

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. The first rule, when applied, attracts the particle versus the leader of the swarm ($gbest$). The second rule forces the particle to follow a random walk. This ensures us to jump out from local minima in the objective function shown in Equation 1. As can be noted, there is no cognitive learning factor but only a social learning factor. We will address the introduction of cognitive factors in the future research.

4 Experimental Results

In this section, we present the experimental results obtained by implementing the proposed methodology in the *System Tuning Shell* optimization framework.

For the experimental results, we used the Wattch [5] virtual superscalar architecture as a simulation model of the target system. The architecture model has been plugged into the optimization framework by creating a suitable wrapper or *Driver*. The Wattch Architecture Driver and the DPSO exploration algorithm are independent to each other, so that one Architecture Driver can be substituted by another without changing the exploration algorithm. Similarly, given one new exploration algorithm, the user can describe it independently of the target architecture and easily plugging it in the framework.

The exploration of the Wattch architecture has been focused on those design parameters significantly impacting the performance and the energy consumption at the system-level. Each instance of the virtual architecture has been described in terms of the following parameters:

- Procesor issue width sizes (2, 4, 8).
- Number of integer ALUs (1, 2) and multipliers (1, 2).
- Number of floating point ALUs (1, 2) and number of multipliers (1, 2).
- Size of the I/D L1 caches (2K, 4K, 8K, 16K Byte)
- Size of the unified L2 cache (16K, 32K, 64K, 128K Byte).
- Block size of the I/D L1 caches (16, 32 Byte)
- Block size of unified L2 cache (32, 64 Byte).
- Associativity of the I/D L1 caches (1-way, 2-ways for the I-cache, 2-ways, 4-ways for the D-cache).

- Associativity of the unified L2 cache (4-ways , 8-ways).

Globally, the architectural design space is composed of 196608 points to be evaluated.

The benchmark suite used is composed of the following programs:

- *FIR*: A finite impulse response filter;
- *Gamma*: Numerical algorithm implementing the gamma function;
- *Gauss*: Gaussian elimination algorithm used to determine the solutions of a system of linear equations;
- *Quarcube*: Quadratic and cubic equations solving routine;
- *DCT*: Numerical algorithm implementing Discrete Cosine Transform.

The STShell framework has been applied to each of the above benchmarks to derive the Energy [nJ] and Delay[cycles] cost functions.

To measure the distance between the Pareto-optimal front and the approximated front we introduce the *Average Distance from Reference Set* [6] ($ADRS$). Let us assume $A \subseteq \Omega$ is a set of design vectors, while $p(A)$ is the relative non-dominated set, the function $ADRS(.,.)$ measures the distance between the $p(A)$ set and the Pareto-optimal set $X_p = p(\Omega)$ as follows:

$$ADRS(X_p, p(A)) := \frac{1}{|X_p|} \sum_{\vec{x_p} \in X_p} \left(\min_{\vec{a} \in p(A)} \{d(\vec{x_p}, \vec{a})\} \right) \tag{4}$$

where:

$$d(\vec{x_p}, \vec{a}) = \max_{j=1,\ldots,m} \left\{ 0, \frac{f_j(\vec{a}) - f_j(\vec{x_p})}{f_j(\vec{x_p})} \right\} \tag{5}$$

and m is the number of objective functions.

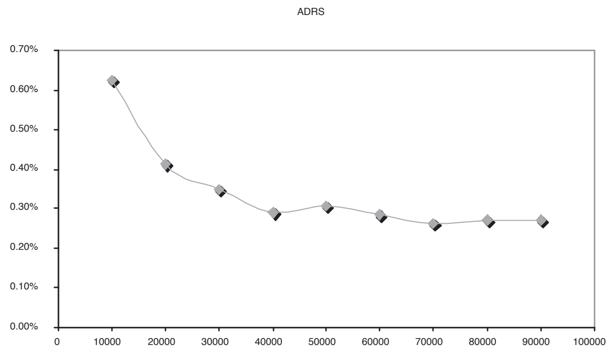

Figure 1. Behavior of the Average Distance From Reference Set metric averaged on the selected set of benchmarks by varying the number of points.

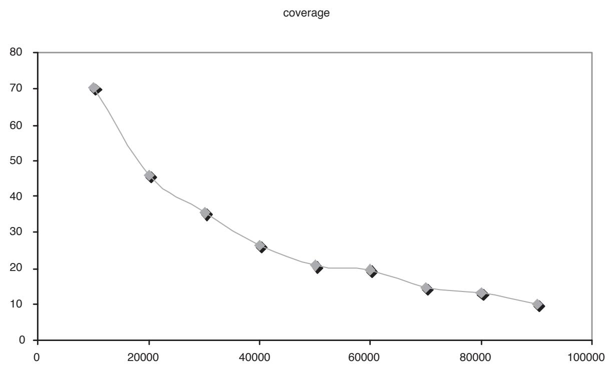

Figure 2. Behavior of the coverage function of the approximated DPSO Pareto set with respect to the real Pareto-front, averaged on the selected set of benchmarks.

Figure 1 shows the average behavior of the Average Distance From Reference Set metric averaged on the selected set of benchmarks by varying the number of points analyzed by our DPSO algorithm. To generate these data, we varied the number of iterations of the algorithm (also called the *generation* number) as well as the sub-swarm size and number. Also the probability p (Equation 3) has been varied by assuming the values $\{0.1, 0.5, 0.8\}$. As can be seen, the value of the Average Distance From Reference Set metric is very low ($< 0.70\%$) and reaches an asymptote at 0.30%. This represents a good indication that our algorithm, even with few iterations and swarm size, can generate a good approximation of the optimal Pareto front.

To further measure the performance of the DPSO algorithm, we use the *coverage* metric [15]. Let us consider A, $B \subseteq \Omega$ be two sets of design vectors and $p(A)$, $p(B)$ be the corresponding nondominated set, the coverage function $C(.,.)$ maps the ordered pairs $(p(A), p(B))$ to the [0,1] interval as follows:

$$C(p(A), p(B)) := \frac{|\{\vec{b} \in p(B); \exists \vec{a} \in p(A) \ : \ \vec{a} \prec \vec{b}\}|}{|p(B)|} \tag{6}$$

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

Figure 2 shows the average behavior of the coverage of the approximated DPSO Pareto set with respect to the real Pareto-front. The metric has been averaged across all the benchmarks by varying the number of points analyzed by our DPSO algorithm as in the previous metric. As can be noted, the coverage drops under 50% before reaching the 20.000 analyzed points (i.e. 10% of the total design space) while it goes under 30% for 40.000 points (20 % of the total design space). In other words, the accuracy of the algorithm in approximating the real Pareto front is more than 70 % when the percentage of the analyzed design space is only 20%.

Finally, we considered as further metric, the average euclidean distance between the approximated and the real Pareto front. The function $distance(.,.)$ measures the distance between the approximated Pareto set B set and the Pareto-optimal set A as follows:

$$\text{distance}(A, B)) := \text{median}_{\vec{a} \in A} \left(\min_{\vec{b} \in B} \{ \text{d}(\vec{b}, \vec{a}) \} \right) \quad (7)$$

where $\text{d}(a, b)$ is the euclidean distance, in the objective function space, between points $\vec{b}$ and $\vec{a}$. We use the *median* operator since it is more robust with respect to outliers.

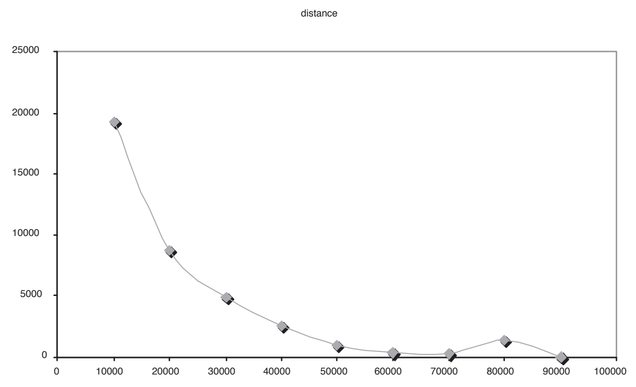

Figure 3. Behavior of the distance operator when the number of points analyzed increases, averaged on the selected set of benchmarks.

Figure 3 shows the behavior of the distance operator when the number of points analyzed increases. As can be noted, the function presents an exponential behavior which drops to negligible values after 50.000 analyzed points (25% of the search space).

5 Conclusions

In this paper we presented a Discrete Particle Swarm Optimization methodology (DPSO) for supporting the *design space exploration* task (DSE). By means of the proposed technique, we are able to efficiently profile the target applications and derive an approximated Pareto set of configurations with respect to the selected figures of merit. Experimental results have shown that the proposed DPSO technique is able to speed up the design space exploration time up to 5X with an accuracy of up to 70% with respect to a full search for the selected set of benchmarks.

References

[1] G. Ascia, V. Catania, and M. Palesi. A multi-objective genetic approach for system level exploration in parameterized system-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):635–645, 2005.

[2] U. Baumgartner, C. Magele, and W. Renhart. Pareto optimality and particle swarm optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, 2004.

[3] G. Beltrame, D. Bruschi, D. Sciuto, and C. Silvano. Decision-theoretic exploration of multiprocessor platforms. In *CODES+ISSS '06: Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pages 205–210, New York, NY, USA, 2006. ACM Press.

[4] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. Pisa - a platform and programming language independent interface for search algorithms, 2003.

[5] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings ISCA 2000: International Symposium on Computer Architecture*, pages 83–94, 2000.

[6] J. A. Czyak P. Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimisation. *Journal of Multi-Criteria Decision Analysis*, (7):34–47, April 1998.

[7] T. D. Givargis and F. Vahid. Platune: a tuning framework for system-on-a-chip platforms. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 21(11):1317–1327, November 2002.

[8] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[9] K. Keutzer, S. Malik, A. R. Newton, J. Rabaey, and A. Sangiovanni-Vincentelli. System level design: Orthogonolization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(12):1523–1543, December 2000.

[10] G. Palermo, C. Silvano, S. Valsecchi, and V. Zaccaria. A system-level methodology for fast multi-objective design space exploration. In *GLSVLSI '03: Proceedings of the 13th ACM Great Lakes symposium on VLSI*, pages 92–95, New York, NY, USA, 2003. ACM Press.

[11] G. Palermo, C. Silvano, and V. Zaccaria. A flexible framework for fast multi-objective design space exploration of embedded systems. In *Proceedings of International Workshop-Power And Timing Modeling, Optimization and Simulation, PATMOS03*, 10–12 2003.

[12] M. Palesi and T. Givargis. Multi-objective design space exploration using genetic algorithms. In *Proceedings of the Tenth International Symposium on Hardware/Software Codesign, 2002. CODES 2002*, May 6–8 2002.

[13] M. Reyes-Sierra and C. A. Coello. Multiple-objective particle swarm optimizers: A survey of the state of the art. *http://www.lania.mx/∼ccoello/EMOO/reyes06.pdf.gz*, 2006.

[14] C. Ykman-Couvreur, V. Nollet, T. Marescaux, E. Brockmeyer, F. Catthoor, and H. Corporaal. Pareto-based application specification for mp-soc customized run-time management. *Embedded Computer Systems: Architectures, Modeling and Simulation, 2006 International Conference on*, pages 78–84, 2006.

[15] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.