Multi-Objective Optimization for High-Level Synthesis

Marcela Zuluaga[1], Andreas Krause[1], Guillaume Sergent[2], Markus Püschel[1]

[1] ETH Zurich, Department of Computer Science
[2] ENS Lyon
zuluaga@inf.ethz.ch, krausea@ethz.ch, guillaume.sergent@ens-lyon.fr, pueschel@inf.ethz.ch

1. Introduction

High-level synthesis tools offer great value to hardware designers, a simple and clean behavioral specification can quickly lead to the corresponding Register-Transfer Level (RTL) description. This automatic process is usually set to make implementation choices that optimize design goals, such as chip area, throughput, energy consumption, etc. However, as typically some of these goals conflict with each other, there is not a sequence of choices that optimizes at the same time every single metric. Thus, the degrees of freedom in high-level synthesis tools can generate a large number of designs, each with different characteristics, and from which the set of Pareto optimal ones constitute very valuable information to users. In order to confirm the optimality of a design, the entire design space needs to be evaluated, typically by running time-consuming synthesis processes.

One of our case studies is a design space created by an IP generator for sorting networks [7]. The user specifies the number of elements to sort and several tool configurations that affect the area and the throughput of the generated design. The plot in Fig. 1 shows the obtained area/throughput values after synthesizing each design that the tool can generate, for a module that sorts 256 elements. Area and throughput were found after running FPGA synthesis and place-and-route for each design; this exploration took several days. The Pareto-optimal designs are joined with a line and represent the only tradeoffs that are of interest to the users. The fundamental problem addressed in this work is how to predict the set of choices that lead to Pareto-optimal designs at low cost, this is by evaluating as few designs as possible.

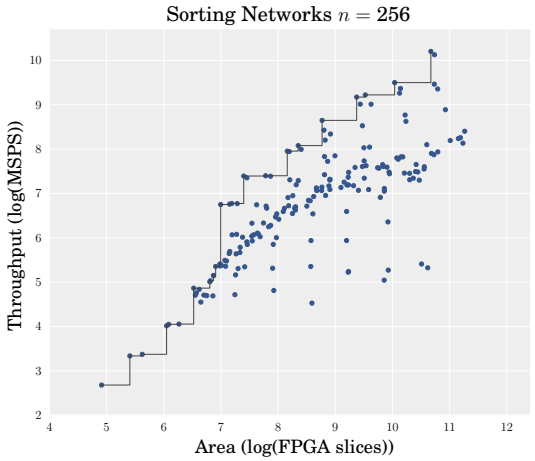

Figure 1. Evaluation of the design space generated for sorting networks that process 256 elements. Throughput is measured in mega samples per second (MSPS).

2. Pareto Active Learning

We propose the *Pareto Active Learning* (PAL) algorithm, that is specifically designed to efficiently address multi-objective optimization problems in which the objective functions are expensive to evaluate. PAL can be integrated with high-level synthesis tools in which design tradeoffs can be exposed through parameterization, to give the users a set of optimal solutions instead of hiding important tradeoffs or of asking them for parameters that do not affect the functionality of the resulting design.

PAL is parameterized by a user-defined variable ϵ to enable an intuitive mechanism to find the desired tradeoff between sampling cost and prediction accuracy. We now describe the main components of the algorithm.

(1) **Modeling.** PAL uses Gaussian process modeling to capture domain knowledge about the regularity in the design space. These models are used to predict the objective values of the designs that have not been evaluated yet, and to obtain the uncertainty associated with this prediction.

(2) **Sampling.** An iterative training process takes place until there is enough information to accurately predict the Pareto-optimal designs. On every iteration, a design is chosen to be evaluated with the goal of maximizing progress on accurately identifying Pareto-optimal designs. Predicted values and their corresponding uncertainties are used to guide this iterative sampling process.

(3) **Predictive classification.** A set of classification rules predicts whether a design is Pareto-optimal or not Pareto-optimal with high probability. This helps to evaluate progress and to determine when there is enough information to generate an accurate prediction.

(4) **Stopping criteria.** PAL terminates when all designs are classified as either Pareto-optimal or not Pareto-optimal. At this point, a prediction of the designs that are Pareto-optimal is generated. The user could make further evaluations within the optimal set if the tradeoff of interest was not evaluated during the training stage.

Fig. 2 shows an example of running PAL on a 2-objective maximization problem. At a given iteration the objective functions f_1 and f_2 are predicted for every configuration vector $\boldsymbol{x}$ in the design space that has not been evaluated yet. From this prediction we create the uncertainty region $R(\boldsymbol{x})$, which contains all the values that $(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}))$ can have with high probability. A predictive classification is made taking into account that a point $\boldsymbol{x}$ can fall anywhere within $R(\boldsymbol{x})$. As some points are not classified yet, the iteration continues with a new sample being evaluated. The next configuration $\boldsymbol{x}$ chosen for evaluation is the one with the uncertainty region with the largest diagonal, within the set of unclassified points plus the points classified as Pareto-optimal. This example considers $\epsilon = 0$. Greater values of ϵ relax the classification rules allowing the algorithm to stop earlier, trading off training cost for accuracy. Our approach uses an ϵ-accurate classification rule similar to the concept of ϵ-Pareto dominance [2].

PAL is an improvement over the GP-PUCB algorithm presented in [6]. This new approach has been devised to tackle multi-objective optimization as a general problem. Thus, it could be applied to a large range of complex processes with an arbitrary number of target objectives. Moreover, we provide theoretical bounds on PAL's sampling cost required to achieve a desired accuracy.

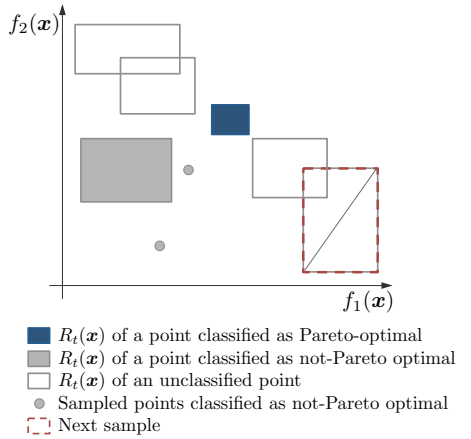

Figure 2. An example iteration of PAL in a 2-objective maximization scenario.

3. Related Work

Evolutionary algorithms. Evolutionary algorithms are often used to solve multi-objective optimization problems. Most of these approaches do not use models to generate predictions on the objective functions. Therefore, typically they require a large number of iterations in order to obtain an accurate prediction of the Pareto-optimal set. A small subset of evolutionary algorithms uses modeling techniques to speedup the search process. A well known algorithm in this category is ParEGO [3]. We compare our results against this approach.

Scalarization. Another approach is to combine the objective functions to create one or several single-objective optimization problems. A drawback of this solution is that it is hard to cover all possible tradeoffs.

Bayesian optimization. Much work has been done on single-objective optimization based on Bayasian modeling. The GP-UCB algorithm, proposed in [4], uses Gaussian process to model the objective function and to guide sampling. The authors also provide convergence guarantees. We base our analysis on their results to obtain similar guarantees in a multi-objective scenario.

4. Experimental Results

PAL was evaluated on two applications from the domain of hardware design [1, 7], in which it is very expensive to run low-level synthesis to obtain the exact cost and performance of a single design. We compare the performance of PAL against ParEGO. In both data sets PAL outperforms ParEGO, requiring about 33% less evaluations. Our metric to compare actual versus estimated Pareto front is the percentage error $\mathcal{E}_{t,i}$, which expresses the percentage error on the Pareto front prediction along the axis defined by the objective function i after evaluating t designs from the design space. This metric is based on the logarithmic hypervolume [5], which proportionally penalizes mispredictions across the objective space. Fig. 3 shows the results obtained with the sorting network data set in Fig. 1. These plots show, for different values of ϵ, the percentage error obtained for each of the objective functions versus the number of evaluations performed when the algorithm terminated.

5. Conclusion

This work proposes the PAL algorithm to solve an optimization problem that is not uncommon in high-level synthesis processes. A particularity of hardware design is that the cost-performance tradeoff needs to be adjusted to every application requirements,

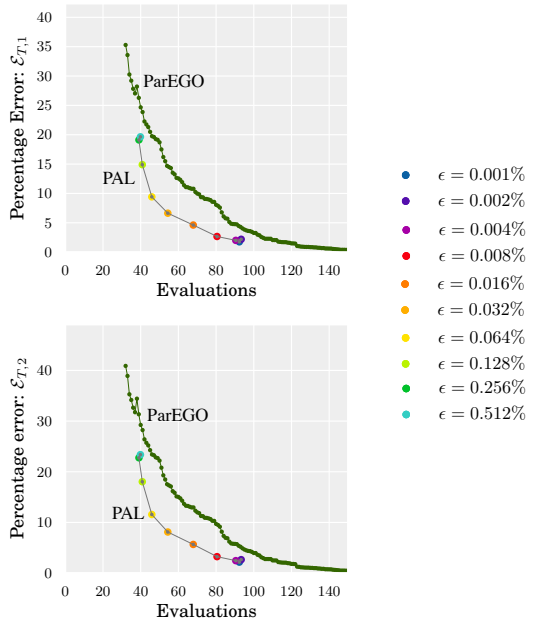

Figure 3. An example iteration of PAL in a 2-objective maximization scenario.

thus it is important to provide the users with a wide range of optimal tradeoffs. We make use of state-of-the-art machine-learning theory to efficiently solve this problem, thus enabling the construction of more advanced high-level synthesis tools. Finally, we carried out an extensive empirical evaluation, where we demonstrate PAL's effectiveness on several real-world multi-objective optimization problems. Our results show that sampling a small fraction of the design space can indeed yield predictions of the Pareto front that are sufficiently accurate to be of use in a wide range of applications.

References

[1] O. Almer, N. Topham, and B. Franke. A Learning-Based Approach to the Automated Design of MPSoC Networks. *Architecture of Computing Systems (ARCS)*, pages 243–258, 2011.

[2] S. Helbig and D. Pateva. On several concepts for ϵ-efficiency. *Operations-Research-Spektrum*, 16:179–186, 1994.

[3] J. Knowles. ParEGO: a Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50 – 66, 2006.

[4] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proc. of International Conference on Machine Learning (ICML)*, 2010.

[5] E. Zitzler, D. Brockhoff, and L. Thiele. The Hypervolume Indicator Revisited: on the Design of Pareto-compliant Indicators via Weighted Integration. In *Proc. of the 4th International Conference on Evolutionary Multi-criterion Optimization (EMO)*, pages 862–876, 2007.

[6] M. Zuluaga, A. Krause, M. P.A., and M. Püschel. "Smart" Design Space Sampling to Predict Pareto-optimal Solutions. In *Languages, Compilers, Tools and Theory for Embedded Systems (LCTES)*, pages 119–128, 2012.

[7] M. Zuluaga, P. Milder, and M. Püschel. Computer Generation of Streaming Sorting Networks. In *Proceedings of the 45th Annual ACM/IEEE Conference on Design Automation (DAC)*, 2012.