# Leveraging Bayesian Optimization to Speed Up Automatic Precision Tuning

Van-Phu Ha, Olivier Sentieys

# Leveraging Bayesian Optimization to Speed Up Automatic Precision Tuning

Van-Phu Ha
*Univ Rennes, Inria, IRISA*
Rennes, France
van-phu.ha@inria.fr

Olivier Sentieys
*Univ Rennes, Inria, IRISA*
Rennes, France
olivier.sentieys@inria.fr

***Abstract*—Using just the right amount of numerical precision is an important aspect for guaranteeing performance and energy efficiency requirements. Word-Length Optimization (WLO) is the automatic process for tuning the precision, i.e., bit-width, of variables and operations represented using fixed-point arithmetic. However, state-of-the-art precision tuning approaches do not scale well in large applications where many variables are involved. In this paper, we propose a hybrid algorithm combining Bayesian optimization (BO) and a fast local search to speed up the WLO procedure. Through experiments, we first show some evidence on how this combination can improve exploration time. Then, we propose an algorithm to automatically determine a reasonable transition point between the two algorithms. By statistically analyzing the convergence of the probabilistic models constructed during BO, we derive a stopping condition that determines when to switch to the local search phase. Experimental results indicate that our algorithm can reduce exploration time by up to 50%-80% for large benchmarks.**

## I. Introduction

The rapid development in scientific and technological innovations during the last decade opens a new era for intelligent and sophisticated systems. The demand for integrating many large applications in a limited silicon area poses new challenges for energy-efficient computing. Recently, Approximate Computing (AC) is considered as a good solution to address energy efficiency issues. The primary objective of approximation techniques is to trade quality of service for cost saving. One of the popular AC techniques is to use Fixed-Point arithmetic for low-precision computation in Digital Signal Processing or Machine Learning systems. This technique always requires a floating-point to fixed-point conversion that optimizes the fixed-point word-lengths for a good compromise between cost and quality requirement. This procedure, called Word-Length Optimization (WLO), accounts for 25-50% of design time [1] and is still considered as a problem of interest to reduce time-to-market.

Approaches to address WLO can be classified in two groups: analytical and simulation-based approaches. Analytical approaches relax the WLO problem for convexity and then apply some convex optimization algorithms to directly obtain the optimal solution [2], [3]. Despite handling WLO quickly, these approaches require the accuracy to be modeled as convex functions, which cannot be analytically constructed in general. Simulation-based approaches solve WLO by iterative search using simulations [4]–[8]. They are thus generic with all systems and quality metrics. However, these approaches do not scale well in large applications where many variables are involved since the number of simulations increases dramatically with the number of variables. In addition, most of these iterative searches are based on local search that moves with a short distance in the discrete domain. Hence, convergence speed is slow if the initial point is far from a local minima.

Bayesian Optimization (BO) is a popular approach for tuning hyper-parameters in machine learning algorithms [9]. This approach aims to optimize problems where the mathematical expression of the target function is unknown and without derivatives. BO constructs a probabilistic model based on past samples to suggest new points [10]. Thanks to this model, at a certain state, BO can ignore neighboring points if they produce a low probability of being good solutions to search more quickly. This feature might overcome weakness of the simulation-based approaches. However, there is no study yet indicating the performance of Bayesian optimization for WLO.

In this paper, we propose a Hybrid algorithm combining Bayesian optimization and a local search to improve the scalability of simulation-based approaches. We first show how this combination can lead to large improvements in exploration time. Then, we design a reasonable transition point between the two approaches to leverage the efficiency of the combination, which is also the core of our approach. Using design points sampled by BO, we derive a statistical metric to evaluate the convergence of the models during BO. Experimental results show that our Hybrid algorithm outperforms latest simulation-based approaches, reducing exploration time by up to 50%-80%, while leading to similar cost solutions.

The rest of the paper is organized as follows. We introduce necessary background and discuss related work in Section II. A motivation for our work is presented in Section III. Then, we describe our proposed Hybrid approach in Section IV. We show the performance of our approach in a comparison with different benchmarks and latest approaches in Section V followed by a conclusion in Section VI.

## II. Background and Related Work

In this section, we introduce the WLO problem and discuss earlier work before presenting Bayesian optimization.

### A. Word-Length Optimization and Classical Approaches

A number represented in fixed-point arithmetic contains integer and fractional word-lengths (WLs), represented on $I$ and $F$ bits, respectively. The integer WL covers the dynamic range whereas the fractional WL controls the precision. In this work, we focus on the WLO for fractional WL, which is the time consuming part of the exploration. Let the vector $\boldsymbol{W} = [W_0, W_1, \ldots, W_{N-1}]$ denote a word-length configuration with $N$ effective variables to be explored for fixed-point conversion.

The main objective of WLO is to determine a good-enough word-length configuration that minimizes a cost function under a quality constraint:

$$\min \boldsymbol{C}(\boldsymbol{W}) \quad \text{Subject to} \quad \boldsymbol{\lambda}(\boldsymbol{W}) \geq \boldsymbol{\lambda}_{obj} \tag{1}$$

where $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are functions that express cost and quality, respectively, and the quality target is given as $\boldsymbol{\lambda}_{obj}$. How the functions $\boldsymbol{C}$ and $\boldsymbol{\lambda}$ are realized varies across work, ranging from analytical models to simulation-based approaches.

Some approaches [2], [3], [11] construct analytical models for noise power through mathematical expression to avoid costly simulations during the exploration process. These analytical approaches take advantage of a property of errors, under the hypothesis of linear and time-invariant (LTI) systems, that their propagation do not interfere with each other. Hence, the error introduced at a noise source may be propagated through the system independently and aggregated afterwards. Thus, these approaches cannot be directly extended to handle general non-LTI programs. Moreover, complex quality metrics, such as Structural Similarity (SSIM), which are not directly related to noise power, are hard to model analytically.

Many simulation-based approaches [4]–[8] were proposed based on iterative search using heuristics to address WLO. These approaches use variants of gradient descent algorithms that evaluate neighboring solutions which differ by one or few bits from the current solution at each iteration. Since the number of neighboring solutions increases with the number of variables, the number of solutions that must be evaluated at each iteration quickly increases with the complexity of the application. Additionally, due to short movements at each iteration (one or few bits), these approaches require many iterations to converge if the starting point is far from a local minima. These are the main drawbacks causing a huge number of simulations especially in large applications and hence leading to an extremely long exploration time. Min+1, Min+$b$ [4], [6] and Max$-1$ [4] are typical approaches of this group. The Min+1 (resp. +$b$) algorithm begins with a design where each variable is assigned by its minimum WL (MWL), i.e., the WL satisfying the quality target when other variables are set to the highest precision. At each iteration, the algorithm moves towards neighboring candidates by increasing by 1 (resp. $b$) bit(s) the best variable among the candidates until obtaining a solution satisfying the quality constraint. By contrast, Max$-1$ initializes variables with the highest possible WL and decreases variables by 1 bit until to reach the final solution. Afterwards, many variants of these two procedures were proposed. Heuristic approaches [12] or Hybrid combinations of Min+$b$ and Max$-1$ outperform the original algorithms [13]. Likewise, the Greedy Randomised Adaptive Search Procedure (GRASP) [8] combines local search and stochastic optimisation. GRASP is an iterative two-phase procedure. In the construction phase, the search algorithm (similar to Min+1) randomly selects one of the best neighboring candidates during gradient descent. Then, a Tabu search is applied to refine the solution found by the first phase. Tabu allows movements in both directions (+1/$-1$) and uses a Tabu list to skip some explored variables. These two phases are iterated and the randomization of the construction phase avoids to stay in local minima.

Some noise budgeting techniques [14], [15] decompose large applications into smaller kernels to break down the exponential complexity of WLO. However, these methods still face the limitation of the classical above-mentioned approaches, which are used for each kernel to solve local WLO problems.

### *B. Bayesian Optimization*

Bayesian Optimization (BO) is a machine-learning-based optimization method [9] aiming to optimize functions which usually have no mathematical expression and/or derivatives. Despite being widely applied in many real world problems, there is no study of BO for WLO. Generally, BO has two key elements: i) a probabilistic surrogate model for modeling the unknown objective function based on already observed samples and ii) an acquisition function that optimizes over the surrogate model to suggest next samples. One main difference in BO methods is the process of selecting the surrogate models. While Gaussian Processes (GP) are often used for continuous-domain moderate-size problems, tree-based models like Random Forest and Tree-structured Parzen Estimator (TPE) facilitate discrete-domain large-size problems [16]. TPE works by identifying points that could have been drawn, and that appear promising on the basis of the evaluation of a loss function at other points. This loss function is very important to guide the optimizer, as detailed in Section IV-A.

Unlike gradient-based algorithms, BO inspects past iterations to construct the interplay among variables to evaluate a loss function via a probabilistic model. Based on the knowledge from this model, BO is able to evaluate and ignore neighboring designs which have low probability of being good solutions to search more globally. As a result, it can speedup the search and better avoid local minima if current state is on a plateau or a bad local minima. Thus, BO is a promising candidate to overcome limitation of classical simulation-based WLO approaches. However, the computational complexity of BO is quadratic $O(i^2)$, whereas most heuristic approaches have a linear complexity $O(i)$, $i$ being the number of iterations. Thus, BO still faces scalability issues, especially for large applications in which BO requires more evaluations to obtain a good solution. Based on their features and properties, we propose a new method that combines Bayesian optimization and local search with a reasonable transition condition to improve the exploration efficiency. TPE is selected for BO because it is suited to discrete problems as WLO, and Tabu, a state-of-the-art WLO method, is used for local search.

## III. Motivations

In this section, we compare the performance of TPE with Tabu combined with Min+1 gradient descent for WLO. Then, we indicate benefits of combining TPE and Tabu via empirical evidence, which motivates the proposed Hybrid approach presented in Section IV. We used an Infinite Impulse Response (IIR) filter with Peak Signal to Noise Ratio (PSNR) equal to 40 dB as a quality constraint. Note that the trends in this section are consistent with the other benchmarks used in Section V-B.

### *A. Performance Analysis: TPE vs. Tabu*

Figure 1 shows the search process of Tabu and TPE during WLO of the IIR. Tabu searches in a narrow range constituted by near neighboring solutions which are some bits different to each other. Thus, it moves slowly towards optimal region. At

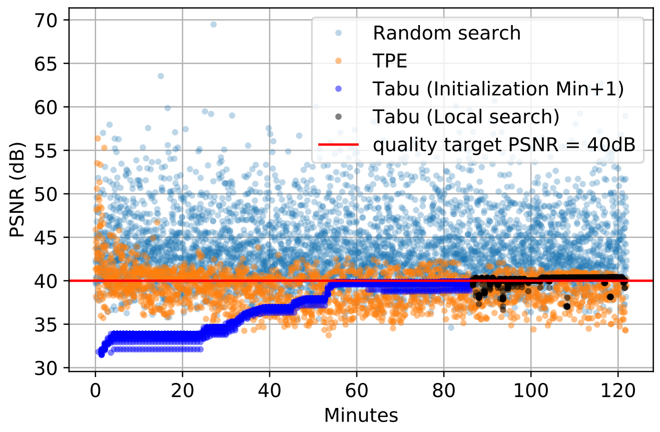

Fig. 1: Search process of Tabu and TPE. Random search is used as a reference. Points correspond to quality of different designs evaluated in the search process (for Tabu, all neighboring solutions in an iteration are plotted at a corresponding time). Tabu has two procedures: i) **Initialization** uses Min+1 to search in infeasible region until a solution satisfying the quality target is found and ii) **Local search**, a combination of Min+1 and Max-1, optimizes locally around the quality target.

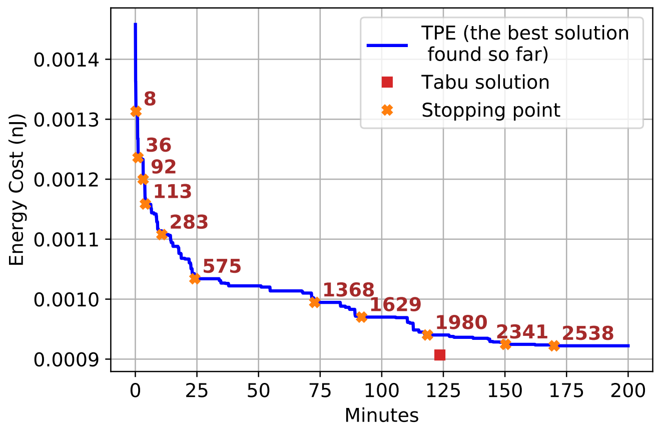

Fig. 2: Manually selected stopping points on the best solution obtained so far by TPE. The curve was constructed by best designs, satisfying the quality target with minimum cost, at a certain time. The numbers indicate index of the selected points.

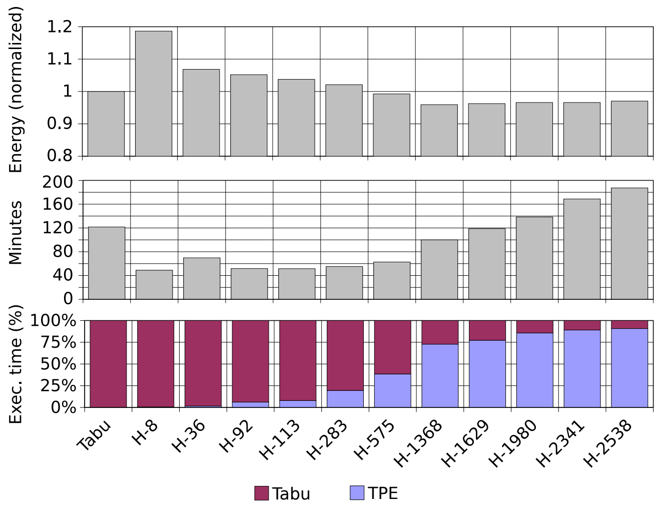

Fig. 3: A comparison in energy cost (top), total exploration time (middle) and separated exploration time (bottom) of Tabu and different combinations between TPE and Tabu given the selected transition points. The x-axis separates the data for different cases; H-X represents for the different combinations where X is the index of the stopping point of TPE.

some points in the initialization procedure, the search takes a significant period to surpass the plateau region which has no quality improvement. As a result, local search must wait for a long time and is only started when a feasible solution is found in the initialization procedure. Meanwhile, TPE first searches randomly in the solution space (very first solutions scattered in around 35-55 dB). Afterwards, it explores solutions using probabilistic models in a wide range and quickly finds feasible solutions around the quality target. Note that by using models, TPE can stick around the quality target to enhance the search efficiency instead of whole solution space as random search.

For cost comparison (see Section V-A for more details on cost), Figure 2 shows the best solution obtained so far by TPE as a function of exploration time and the final solution obtained by Tabu. TPE converges quickly in the beginning. However, the cost is improving slowly for latest iterations, taking up a large portion of the total execution time. With this behavior, we emphasize the interest of stopping TPE in pruning phase to switch to a more efficient local search, such as Tabu.

## *B. Initial Combinations of TPE and Tabu*

From the performance analysis of TPE and Tabu, we empirically perform initial combinations of the two algorithms. TPE is stopped at a certain moment returning a temporary configuration, defined as a *transition point*, which is then served as starting point for Tabu. Among the designs explored by TPE, we use only the best solution obtained so far. These transition points (orange crosses in Figure 2) are selected if they have a significant improvement in terms of cost compared to the previous adjacent selected one. This ensures a high chance of having different WL configurations.

In Figure 3, we compare the performance of Tabu with TPE+Tabu for different transition points. The cost of the final solutions depends on the moment when TPE is stopped. Stopping TPE too early (H-8 to H-283) does not lead to good solutions compared to Tabu alone, even though their convergence speed is much faster. Stopping TPE after a certain amount of time (H-578, H-1368) can lead to an equivalent or even better solution cost, still in a shorter time. The benefit of the superior convergence speed is gradually decreased if TPE is stopped too late (H-1629 to H-2538). It is important to notice that, except for the early ones, the choice of the exact transition point does not impact much the cost of the final solution.

From this analysis, we highlight that TPE can quickly prune the design space and find a solution from which Tabu will be able to fine-tune the cost, resulting in a significant reduction in exploration time. This reduction depends on choosing the right transition point, which is not an obvious task especially in an automated way, and is the aim of the next section.

## IV. Proposed Hybrid Approach

In this section, we construct the loss function used in our Bayesian optimization and define a method to automatically find the transition point between TPE and Tabu.

## A. Loss Function

Derived from the original WLO problem of Eq. 1, we construct the loss function using the Lagrangian form as

$$f(\boldsymbol{W}) = \boldsymbol{C}(\boldsymbol{W}) - \alpha(\boldsymbol{\lambda}(\boldsymbol{W}) - \boldsymbol{\lambda_{obj}}), \tag{2}$$

with a positive and big enough $\alpha$. TPE will then follow $f(\boldsymbol{W})$ to find solutions that minimize the original problem.

Based on many WLO experiments, we experienced that the cost and quality functions tend to be proportional to $\boldsymbol{W}$. A high quality solution mostly corresponds to a high cost. Thus, the best solutions to satisfy the constraint with a small cost are likely to be around the quality target $\boldsymbol{\lambda_{obj}}$. Therefore, for faster convergence, we force the loss function to cover only a narrow range $[q_l, q_h]$ around $\boldsymbol{\lambda_{obj}}$. The loss function is then defined as

$$f(\boldsymbol{W}) = \begin{cases} \boldsymbol{C}(\boldsymbol{W}) - \alpha(\boldsymbol{\lambda}(\boldsymbol{W}) - \boldsymbol{\lambda_{obj}}) & \text{if } \boldsymbol{\lambda}(\boldsymbol{W}) \in [q_l, q_h] \\ +\infty & \text{otherwise} \end{cases} \tag{3}$$

Moreover, to penalize solutions below $\boldsymbol{\lambda_{obj}}$ with regards to solutions above the target, we choose $\alpha = 0.5$ for solutions below $\boldsymbol{\lambda_{obj}}$, and $\alpha = 0$ for other solutions in the range $[q_l, q_h]$.

## B. Transition Point for Hybrid Approach

In Bayesian optimization relying on TPE [10], hyper-parameters –WL configurations $\boldsymbol{W}_i$ in our case– are chosen uniformly over the search space and evaluated by the loss function $f(\boldsymbol{W})$. Then, obtained samples $\{\boldsymbol{W}_i, f(\boldsymbol{W}_i)\}$ stored in an observation history $\boldsymbol{\mathcal{H}}$ are divided into two groups. The first group contains *good* samples where the loss $f_i$ is less than a threshold $\gamma^*$, whereas the second group consists of the remaining, considered as *bad* samples. TPE uses these two sample groups to model two likelihood probability density functions $l(\boldsymbol{W}) = p(\boldsymbol{W}|f(\boldsymbol{W}) < \gamma^*)$ and $g(\boldsymbol{W}) = p(\boldsymbol{W}|f(\boldsymbol{W}) \geq \gamma^*)$, respectively. Then, it decides which hyper-parameter to try in the next iteration by maximizing the ratio

$$\frac{l(\boldsymbol{W})}{g(\boldsymbol{W})} = \frac{p(\boldsymbol{W}|f(\boldsymbol{W}) < \gamma^*)}{p(\boldsymbol{W}|f(\boldsymbol{W}) \geq \gamma^*)}.$$

Algorithm 1 describes TPE algorithm with a stopping condition

**Algorithm 1** TPE with stopping condition

```
1:  $\boldsymbol{\mathcal{H}} \leftarrow \{\}$
2:  for $i \in [1, \dots, T]$ do
3:      $\boldsymbol{W}^* = argmax \frac{l(\boldsymbol{W})}{g(\boldsymbol{W})}$
4:      Evaluate $f(\boldsymbol{W}^*)$
5:      $\boldsymbol{\mathcal{H}} \leftarrow \boldsymbol{\mathcal{H}} \cup (\boldsymbol{W}^*, f(\boldsymbol{W}^*))$
6:      if stopping condition is satisfied then
7:          Return $\boldsymbol{W}_{stop}$
8:      end if
9:      Update $l(\boldsymbol{W})$ and $g(\boldsymbol{W})$ given $\boldsymbol{\mathcal{H}}$
10: end for
```

checked at each iteration. The algorithm stops if the stopping condition is satisfied and then returns a WL configuration $\boldsymbol{W}_{stop}$ which serves as the starting point of Tabu. How the condition is constructed is described as follows.

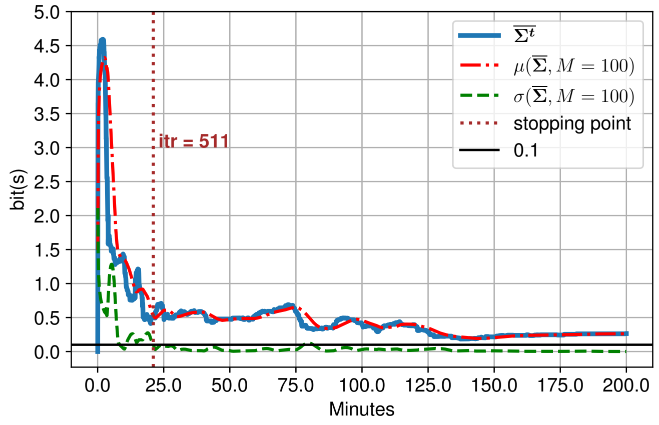

Fig. 4: Stopping point for IIR, quality target PSNR = 40dB.

At $t^{th}$ iteration, we consider the vector $\boldsymbol{S^t_L}$ comprising the $L$ top-performing solutions from the observation history $\boldsymbol{\mathcal{H}}$. Our main objective is to evaluate if the next iteration of TPE can lead to a new top-performing solution, that is significantly different in terms of WL configuration compared with the solutions in $\boldsymbol{S^t_L}$. Then, we decide to stop TPE if the solutions in $\boldsymbol{S^t_L}$ share similar WL configurations. Otherwise, we still wait for further exploration iterations. The similarity of WL configurations in $\boldsymbol{S^t_L}$ reflects that TPE is likely to converge to a local region formed by a number of similar WL solutions, which only differ by few bits. Thus, continuing to spend time with TPE is likely to be inefficient. Instead, a local search like Tabu is more reasonable to converge quickly in that local region. This is the key idea in our method for the combination. We use TPE for narrowing the solution space down to a good region, from which Tabu will continue the search for fine-tuning to provide the final solution following its greedy gradient.

Each solution in $\boldsymbol{S^t_L}$ is represented by a WL configuration with $N$ effective variables. Let the $L \times N$ matrix $\boldsymbol{W_t}$ contain WL configurations of the $L$ top performing solutions at iteration $t$. Elements $w^t_{i,j}$ of $\boldsymbol{W_t}$ represent the number of bits of variable $j \in [1, N]$ from solution $i \in [1, L]$. We use the standard deviation to evaluate the distribution of WL values for each variable (i.e., each column of $\boldsymbol{W_t}$) as

$$\boldsymbol{\Sigma_t} = [\sigma^t_0, \sigma^t_1, \dots, \sigma^t_{N-1}], \tag{4}$$

where $\sigma^t_j$ is the individual standard deviation of WL values of the $j^{th}$ variable and is calculated as

$$\sigma^t_j = \sqrt{\frac{1}{L-1}\sum_{i=0}^{L-1}(w^t_{i,j} - \overline{W^t_j})}, \qquad \overline{W^t_j} = \frac{1}{L}\sum_{i=0}^{L-1} w^t_{i,j}. \tag{5}$$

The smaller the standard deviation, the more similar the solutions. To evaluate the similarity on all WL configurations at a certain iteration, we aggregate standard deviations by the average of all $\sigma^t_j$ with $j \in [0, N-1]$ as

$$\overline{\boldsymbol{\Sigma^t}} = \frac{1}{N}\sum_{i=0}^{N-1} \sigma^t_i. \tag{6}$$

$\overline{\boldsymbol{\Sigma^t}}$ represents the average difference on the number of bits for the WL configurations of the L top-performing solutions. By evaluating the convergence of TPE through $\overline{\Sigma^t}$ at each iteration, we obtain the same behavior on all our benchmarks. In Figure 4, we show an illustration of this behavior for the IIR example. In the beginning, the curve of $\overline{\Sigma^t}$ varies dynamically with a high variance, around 4.5 bits. It constantly finds new good solutions which are relatively different to each other, resulting in dissimilar WL configurations in $S_L^t$ and then producing a high value of $\overline{\Sigma^t}$. Afterwards, it enters a more stable area with a value of $\overline{\Sigma^t}$ less than 1 bit. This indicates that TPE is likely to converge to a local region. So, we should stop TPE and switch to a greedy algorithm like Tabu to search locally around this region. Based on that behavior, we derive two indicators to detect the stable region to decide to stop TPE. At the $t^{th}$ iteration, we evaluate the stability of $\overline{\Sigma^t}$ in the $M$ latest samples via mean and standard deviation as

$$\mu(\overline{\Sigma}, M) = \frac{1}{M} \sum_{i=t-M-1}^{t} \overline{\Sigma^i} \tag{7}$$

$$\sigma(\overline{\Sigma}, M) = \sqrt{\frac{1}{M-1} \sum_{i=t-M-1}^{t} (\overline{\Sigma^i} - \mu(\overline{\Sigma}, M))} \tag{8}$$

$\mu(\overline{\Sigma}, M)$ indicates the average magnitude of $\overline{\Sigma}$. $\sigma(\overline{\Sigma}, M)$ is chosen with a small value to ensure the stability of $\overline{\Sigma}$. Then, two corresponding thresholds are chosen to decide the adequate stopping moment for TPE. In our evaluation, we monitor $L = 25$ top-performing solutions at each iteration and choose two conditions, $\mu(\overline{\Sigma}, M) \leq 1$ and $\sigma(\overline{\Sigma}, M) \leq 0.1$, with $M = 100$ to stop TPE. Figure 4 indicates with a dashed line the iteration where TPE is stopped.

## V. Evaluation

### A. Experiment setup

Experiments are performed on an Intel Xeon E5640 2.67GHz with 4GB memory running Linux. We used Adaptive TPE (ATPE), an extension on top of TPE implemented on Hyperopt [17], which provides a machine learning model to automatically tune the hyper-parameters, e.g., $\gamma *$, of TPE. Tabu search is described in [8]. We use energy as our cost model. The energy model counts the number of operations performed by each operator, and calculates the total cost based on the energy consumption of an operation empirically gathered from several ASIC synthesis/simulation for different WLs. An operator is characterized by the WLs of the operands, the WL of the result, and the arithmetic operation performed. Characterization is performed using Synopsys Design Compiler and Prime Time using a 28nm FDSOI technology.

We compare our Hybrid approach with reference algorithms (Max$-1$, Min$+1$, Tabu search and GRASP, as in Section II-A) in terms of solution cost and exploration time. GRASP is configured with $T_{RCL} = 3$ running in 10 iterations as in [8]. During the execution of Min$+1$ and Tabu, we experienced that the search process usually gets stuck in situations that keep increasing the number of bits and thus cost, without a significant quality improvement. Actually, in such situations there are more distant neighbors that gain a significant amount of quality. As a result, more quality can be achieved without increasing the number of bits too much, which reduces redundant cost. The bias versions of Min$+1$ and Tabu search add a bias to the selection criterion to reduce the priority of the candidates for which a significant number of bits has already been added, which leads to optimized versions of the original algorithms.

We evaluate cost and exploration time on five applications. FIR and IIR filters are implemented with 5-stage cascaded structure of $33^{th}$ and $2^{nd}$ order filters, respectively. The number of effective variables to optimize is 17 for FIR and 33 for IIR. Block-Matching and 3D Filtering (BM3D) [18] is a state-of-the-art image processing noise reduction algorithm with 45 effective variables. ISP is an Image Signal Processor chain which applies a sequence of processing kernels on raw data from camera sensors to produce an enhanced color image. ISP comprises four kernels (Non-Local Means (NLM) denoising filter [19], Demosaicing, Gamma Correction, and Unsharp filtering) and has 74 effective variables. Stereo Matching (SM) is widely used in computer vision to extract a depth information of a scene from two images taken from two cameras. SM includes 85 effective variables to optimize.

We use PSNR as the quality metric for filters and Structural Similarity (SSIM) for image processing applications. Two quality constraints $\boldsymbol{\lambda_{obj}}$ are evaluated for PSNR (40 dB and 50 dB) and SSIM (0.9 and 0.99). We use the narrow range $[q_l, q_h]$ as $\pm 2$ dB around $\boldsymbol{\lambda_{obj}}$ for PSNR, $\pm 0.09$ for SSIM=0.9, and $\pm 0.009$ for SSIM=0.99.

### B. Performance Evaluation

Figure 5 presents the comparison between our Hybrid approach and the references in terms of solution cost (top) and exploration time (bottom) for our five benchmarks and two accuracy constraints. Max$-1$ converges fast but always leads to the worst solution cost. Conversely, GRASP obtains relatively competitive solutions but with a long exploration time. The runtime of GRASP is proportional to the number of iterations used to obtained different randomized starting points to avoid local minima. Most solutions found by GRASP are slightly better than those obtained by Min$+1$, Min$+1$[bias] and Tabu. However, GRASP does not outperform Tabu[bias] and is therefore not that efficient in spending longer time through randomization (see e.g., ISP-0.9 example where GRASP still gets stuck in bad local minima).

Thanks to the local bi-directional search, Tabu always find better solutions compared to Min$+1$, and sometimes the bias version is better, sometimes not. However, for a fair comparison, we always use the best reference as a competitor to our Hybrid approach. We also provide the solutions obtained by TPE only iterating for the same time as Tabu[bias]. Solutions obtained by TPE are relatively competitive to Tabu[bias]. For BM3D-0.99, SM-0.9, and SM-0.99, TPE found better solutions than Tabu, which indicates that TPE is able to avoid bad local minima better than Tabu.

The results show that our Hybrid approach always find competitive solutions in shorter time than other methods. For small problems solved in minutes such as FIR-40 and FIR-50, it is sufficient to use classical approaches because they are able to converge in short time. For larger problems solved in a few hours to a few days, our approach can reduce by 50% to 80% (66% on average) exploration time compared to the best reference with similar cost. Our approach even defeats TPE in most benchmarks with slightly better solutions found in a much shorter time (2x-5x faster than TPE, 3.7x on average). Apart

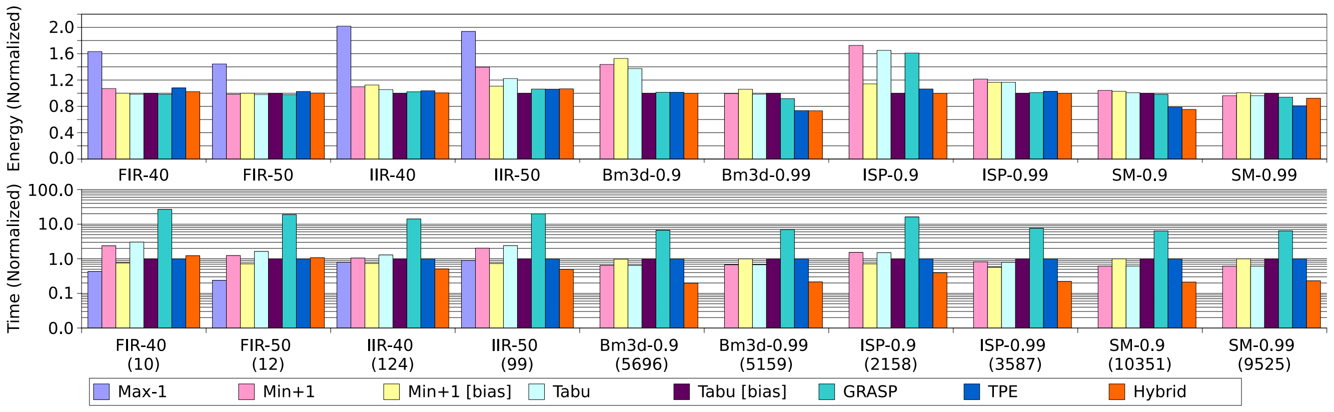

Fig. 5: Energy cost of solutions (top) and exploration time (bottom) normalized to Tabu[bias]. Numbers in parentheses are the execution time in minutes of Tabu[bias] used for the normalization. Results for Max$-1$ are removed for better readability when leading to bad solutions.

from IIR-50 and SM-0.99 where solutions obtained by our approach is 0.4% and 14.5% worse than TPE, respectively, the other cases record a 0.4%-6.3% solution improvement (3.3% on average) compared to TPE. It is clear that the speedup of our approach comes from the benefits of i) TPE to speedup the first phase and ii) our transition condition to stop TPE early. The solution cost improvement takes advantage of the good convergence of Tabu in local regions. In absolute execution time, our approach reduces exploration from 7.2 days to 1.5 days compared to TPE for SM-0.9, from 4 days to 18.8 hours compared to Tabu[bias] for BM3D-0.9, and from 2.5 days to 13.3 hours compared to Tabu[bias] for ISP-0.99.

## VI. Conclusion

In this paper, we propose a Hybrid method leveraging Bayesian Optimization and local search to improve the scalability issue that WLO faces for large applications. We first show how TPE is efficient at narrowing down the search on good regions thanks to its randomness and Tabu for fine-tuning because of its bi-directional local search. Then, the core of our approach is to derive a statistical metric to evaluate model convergence of TPE, to automatically find the right transition moment between TPE and Tabu, and to maintain the search efficiency. For large applications, our approach significantly outperforms state-of-the-art approaches, with a 50%-80% reduction in exploration time, while still providing similar or better cost solutions. The exploration of the parameters used in our methods is important but left as future work.

## Acknowledgment

This work was supported by the French National Research Agency under the ARTEFaCT (ANR-RF-2015-01) and AdequateDL (ANR-18-CE23-0012-01) projects, and by Inria. The authors also want to thank Tomofumi Yuki for his support and valuable feedback in applying TPE for WLO and designing experiments to construct the Hybrid approach.

## References

[1] M. Clark *et al.*, "Accelerating fixed-point design for mb-ofdm uwb systems," *CommsDesign, January*, vol. 4, 2005.
[2] S.-C. Chan and K. M. Tsui, "Wordlength optimization of linear time-invariant systems with multiple outputs using geometric programming," *IEEE Trans. on Circ. and Syst.*, vol. 54, no. 4, pp. 845–854, 2007.
[3] P. D. Fiore, "Efficient approximate wordlength optimization," *IEEE Trans. on Computers*, vol. 57, no. 11, pp. 1561–1570, 2008.
[4] M.-A. Cantin, Y. Savaria, D. Prodanos, and P. Lavoie, "An automatic word length determination method," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 5, pp. 53–56, 2001.
[5] G. A. Constantinides, P. Y. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Trans. on CAD of Int. Circ. and Syst.*, vol. 22, no. 10, pp. 1432–1442, 2003.
[6] K. Han, I. Eo, K. Kim, and H. Cho, "Numerical word-length optimization for cdma demodulator," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 4, pp. 290–293, 2001.
[7] D.-U. Lee, A. A. Gaffar, R. C. Cheung, O. Mencer, W. Luk, and G. A. Constantinides, "Accuracy-guaranteed bit-width optimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1990–2000, 2006.
[8] H.-N. Nguyen, D. Ménard, and O. Sentieys, "Novel algorithms for word-length optimization," in *19th European Signal Processing Conf.*, pp. 1944–1948, IEEE, 2011.
[9] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.
[10] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, pp. 2546–2554, 2011.
[11] K. N. Parashar, D. Menard, and O. Sentieys, "A polynomial time algorithm for solving the word-length optimization problem," in *IEEE/ACM Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 638–645, 2013.
[12] W. Sung and K.-I. Kum, "Simulation-based word-length optimization method for fixed-point digital signal processing systems," *IEEE Transactions on Signal Processing*, vol. 43, no. 12, pp. 3087–3090, 1995.
[13] M.-A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, vol. 2, pp. II–II, 2002.
[14] V.-P. Ha, T. Yuki, and O. Sentieys, "Towards generic and scalable word-length optimization," in *DATE 2020-23rd IEEE/ACM Design, Automation and Test in Europe*, pp. 1–6, IEEE, 2020.
[15] D. Novo, I. Tzimi, U. Ahmad, P. Ienne, and F. Catthoor, "Cracking the complexity of fixed-point refinement in complex wireless systems," in *IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 18–23, 2013.
[16] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," in *NIPS workshop on Bayesian Optimization in Theory and Practice*, vol. 10, p. 3, 2013.
[17] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: a python library for model selection and hyperparameter optimization," *Computational Science & Discovery*, vol. 8, no. 1, 2015.
[18] M. Lebrun, "An analysis and implementation of the bm3d image denoising method," *Image Processing On Line*, vol. 2, pp. 175–213, 2012.
[19] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *IEEE Confe. on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 60–65, 2005.