Roughness of Microarchitectural Design Topologies and its Implications for Optimization

Benjamin C. Lee and David Brooks
School of Engineering and Applied Sciences, Harvard University
{bclee, dbrooks}@eecs.harvard.edu

Abstract

Recent advances in statistical inference and machine learning close the divide between simulation and classical optimization, thereby enabling more rigorous and robust microarchitectural studies. To most effectively utilize these now computationally tractable techniques, we characterize design topology roughness and leverage this characterization to guide our usage of analysis and optimization methods. In particular, we compute roughness metrics that require high-order derivatives and multi-dimensional integrals of design metrics, such as performance and power. These roughness metrics exhibit noteworthy correlations (1) against regression model error, (2) against non-linearities and non-monotonicities of contour maps, and (3) against the effectiveness of optimization heuristics such as gradient ascent. Thus, this work quantifies the implications of design topology roughness for commonly used methods and practices in microarchitectural analysis.

1 Introduction

Microarchitectural optimization is becoming increasingly complex due to greater design diversity driven by two broad trends. First, we observe previously infeasible parts of the design space becoming viable for implementation as the industry moves into the multiprocessor domain. In particular, designers may consider smaller, simpler cores to achieve performance through thread-level parallelism across many of these cores. Secondly, we observe increasing metric diversity as different market segments each define their own acceptable compromises between latency, throughput, power, and temperature. Both trends lead to increasingly non-intuitive design spaces and interesting optima. Characterizing spaces and identifying optima will require advances in microarchitectural simulation and modeling methodology. Designers should leverage best known practices in classical analysis and optimization to further their understanding and supplement their intuition.

Recent advances in applying statistical inference to microarchitectural analysis have enabled fundamentally new capabilities and closed the divide between detailed simulation and classical analysis. More comprehensive characterization, visualization, and optimization of the design topology is now tractable given computationally efficient predictive models constructed from splines or neural networks [2, 9]. For example, pareto frontiers and contour maps for large design spaces are now possible. Iterative optimization heuristics, such as gradient ascent, are also more tractable as predictive models replace simulation within the iterative loop. Fundamentally, predictive models allow designers to leverage the wealth of literature and history in classical analysis and optimization.

As the microarchitectural design community adopts predictive modeling and transfers methods from other more established quantitative disciplines, the emphasis shifts from whether these methods are computationally possible to (1) which methods should be used and (2) how they can be used most effectively. For example, we may now inexpensively generate many contour maps but we lack a formal mechanism to identify particularly interesting contours. While we might implement gradient ascent to search the design space, we may have little guidance for tunable heuristic parameters (*e.g.*, number of trials). Furthermore, we may have even less intuition about the choice of heuristic (*e.g.*, should a stochastic variant have been used instead).

Intuitively, the roughness of a design topology will have direct implications for the effectiveness of various analysis and optimization techniques. Rough topologies increase the likelihood of optimization heuristics identifying local sub-optima. Trends toward greater design diversity imply greater roughness; optimization methods that are robust when applied to rough topologies will become increasingly important. To achieve this robustness, we propose linking topology roughness to usage patterns for any given method. This link will guide users to particularly interesting regions of the design space, to particular parameter values in a tunable heuristic, and to more robust methods that can handle greater non-monotonicity.

To establish this link, we compute roughness metrics for multi-dimensional design topologies and discuss the metrics' role in more effectively using common analysis and optimization methods. In particular, the following summarizes the contributions of this work:

1. **Roughness Metrics:** We compute high-dimensional topological roughness metrics, requiring numerical approximations for high-order derivatives and multi-dimensional integrals. These approximations are provided by computationally efficient predictive models. We examine the link between roughness and model error, demonstrating non-trivial roughness correlations of 0.38 and 0.45 against performance and power, respectively (Section 3).

2. **Roughness and Visualization:** We construct contour maps to visualize the design topology and survey the practical applications of these maps in bottleneck analysis and workload characterization. We examine the link between roughness and contour maps, ensuring contours with greater observed roughness produce larger metric values (Section 4).

3. **Roughness and Optimization:** We implement and evaluate gradient ascent, comparing its results against the true global optimum identified from exhaustively evaluating the predictive models. This comparison is a new capability made possible by efficient models for moderately sized design spaces. We examine the link between roughness and gradient ascent effectiveness, demonstrating non-trivial roughness correlations of 0.35 and 0.20 against heuristic deficiency and path length to optima, respectively (Section 5).

2 Methodology and Background

We apply a scalable and efficient simulation paradigm that defines a comprehensive design space, simulates sparse samples from the space, and constructs regression models for performance and power prediction [9, 10]. This computationally tractable approach reveals trends and trade-offs at high resolution for design evaluation and optimization.

2.1 Simulation Framework

We use Turandot, a generic and parameterized, out-of-order, superscalar processor simulator [12]. Turandot is enhanced with PowerTimer to obtain power estimates based on circuit-level power analyses and resource utilization statistics [1]. The modeled baseline architecture is similar to the POWER4/POWER5. The simulator has been validated against both a POWER4 RTL model and a hardware implementation. This simulator implements pipeline depth performance and power models based on prior work [19].

	Set	Parameters	Measure	Range	$\|S_i\|$
S_1	Depth	depth	FO4	9::3::36	10
S_2	Width	width	issue b/w	2,4,8	3
		L/S reorder queue	entries	15::15::45	
		store queue	entries	14::14::42	
S_3	Physical Registers	general purpose (GP)	count	40::10::130	10
		floating-point (FP)	count	40::8::112	
		special purpose (SP)	count	42::6::96	
S_4	Reservation Stations	branch	entries	6::1::15	10
		fixed-point/memory	entries	10::2::28	
		floating-point	entries	5::1::14	
S_5	I-L1 Cache	i-L1 cache size	KB	16::2x::256	5
S_6	D-L1 Cache	d-L1 sache size	KB	8::2x::128	5
S_7	L2 Cache	L2 cache size	MB	0.25::2x::4	5

Table 1. Design space parameters where $i::j::k$ denotes a set of values from i to k in steps of j.

This prior work assumes logic is perfectly divisible and frequency increases linearly with depth. Power scales superlinearly as pipeline width increases using scaling factors derived for an architecture with clustered functional units [18]. Cache power and latencies scale with array size according to CACTI [16]. We do not leverage any particular feature of the simulator in our models and our framework may be generally applied to other simulators with similar effect.

We use R, an open-source software environment for statistical computing, to script and automate statistical analyses [15]. Within this environment, we use the Hmisc and Design packages implemented by Harrell [5]. We evaluate performance in billions of instructions per second (bips) and power in Watts (w).

2.2 Benchmark Suite

We consider SPECjbb, a Java server benchmark, and eight compute intensive benchmarks from SPEC2k (*ammp, applu, equake, gcc, gzip, mcf, mesa, twolf*). We report experimental results based on PowerPC traces of these benchmarks. The traces used in this study were sampled from the full reference input set to obtain 100 million instructions per benchmark program [6]. Systematic validation was performed to compare the sampled traces against the full traces to ensure accurate representation. Our benchmark suite is representative of larger suites frequently used in the microarchitectural research community [14].

2.3 Spatial Sampling

Table 1 identifies seven groups of parameters varied simultaneously. Parameters within a group are varied together to avoid fundamental design imbalances. The range of values considered for each parameter group is specified by a set of values, $S_1, \ldots, S_7$. The Cartesian product of these sets, $S = \prod_{i=1}^{7} S_i$, defines the entire design space of 375,000 points. We report experimental results from models formulated with $1,000$ samples obtained uniformly at random from the design space S. Spatial sampling effectively decouples design space size from the number of simulations

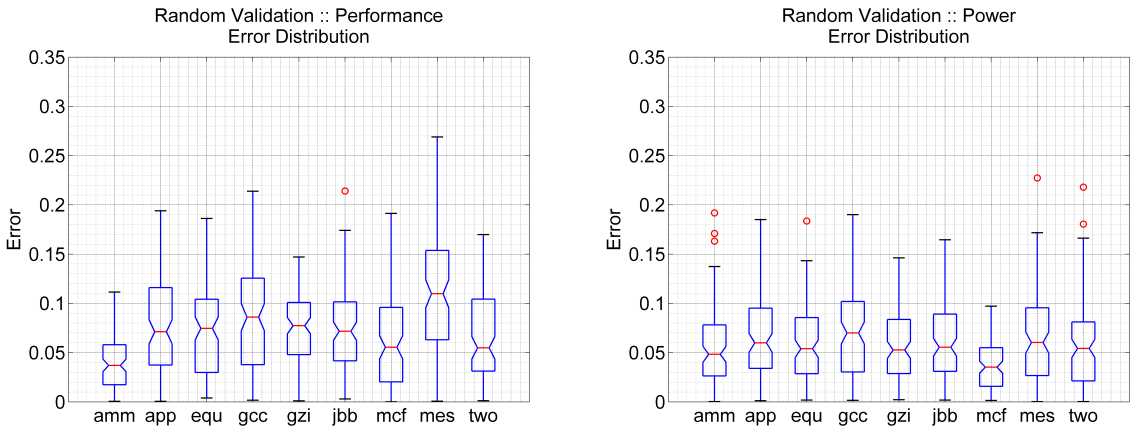

Figure 1. Distribution of performance (a) and power (b) modeling errors for 100 random validation designs.

required to identify a trend within the space, thus controlling exponential increases in space size [9].

2.4 Regression Modeling

We apply techniques in statistical inference proposed by Lee and Brooks, constructing spline-based regression models from sparsely simulated samples of the design space [9]. These models predict a performance or power response as a function of design parameter values. Within this framework, interactions between predictors are captured by products terms specified in the models' functional form using domain-specific knowledge. For example, pipeline depth interacts with cache sizes since depth determines pipeline sensitivity to cache misses. Non-linearity is captured by piecewise cubic polynomial transformations on the predictors. Figure 1 illustrates model accuracy when validated against simulation for 100 randomly selected validation points, demonstrating median errors of 7.2 and 5.4 percent for performance and power prediction, respectively.[1] Such models have been previously applied to practical design studies, demonstrating accuracy sufficient for early stage design optimization [10].

2.5 Gradient Background

We consider gradient analyses to better understand the performance and power topologies of a large, comprehensive microarchitectural design space. The gradient of a function is a vector that points in the direction of greatest increase in the function. The magnitude of this vector is the greatest rate of change. Mathematically, the gradient of a p-dimensional function $f(x_1,\ldots,x_p)$ is $\nabla f(x_1,\ldots,x_p) = (\delta f/\delta x_1,\ldots,\delta f/\delta x_p)$.

[1]Boxplots display location (median) and dispersion (interquartile range), identify possible outliers, and indicate the symmetry or skewness of the distribution. Boxplots are constructed by with horizontal lines at median and at upper, lower quartiles. Circles denote outliers.

In the microarchitectural context, $f(x_1,\ldots,x_p)$ is a performance or power-performance efficiency function in p dimensions corresponding to p design parameters. If $(\tilde{x}_1,\ldots,\tilde{x}_p)$ specifies a design point with particular parameter values, then $\nabla f(\tilde{x}_1,\ldots,\tilde{x}_p)$ specifies the microarchitectural changes to each parameter value that maximizes an increase in performance or efficiency at this design point. We approximate performance and efficiency functions $f(x_1,\ldots,x_p)$ with regression models. The partial derivatives are approximated using one-sided and centered differences for the boundary and interior of the design space, respectively.

Although one-dimensional analysis is widely used, often in sensitivity studies, such an analysis has severe limitations since it does not account for interactions between parameters. For completeness, we describe one-dimensional sensitivity to establish the foundation for more high-dimensional analysis. Demonstrating the traditional role of gradients in microarchitectural design analysis, we consider an example of one-dimensional gradients used to assess performance and power effects from design parameter tuning. The compromise between metrics for a given parameter X_i may be expressed as its sensitivity:

$$S_{Xi}(x) = \left| \left(\frac{\delta Perf/\delta X_i}{Perf} \right) \times \left(\frac{\delta Power/\delta X_i}{Power} \right)^{-1} \right| \quad (1)$$

where $Perf$, $Power$, and their one-dimensional derivatives are functions evaluated at a particular design point $x = (\tilde{x}_1,\ldots,\tilde{x}_p)$. Sensitivity is the absolute magnitude of the percentage change in performance for a percentage change in power. Sensitivities of an optimized design should be balanced such that the marginal power costs of performance from all tunable parameters are equal [11, 20]. Computing these derivatives with respect to each parameter, we identify each parameter's marginal performance benefit at a particular design point. Large S_X indicates parameters from which significant performance gains are possible

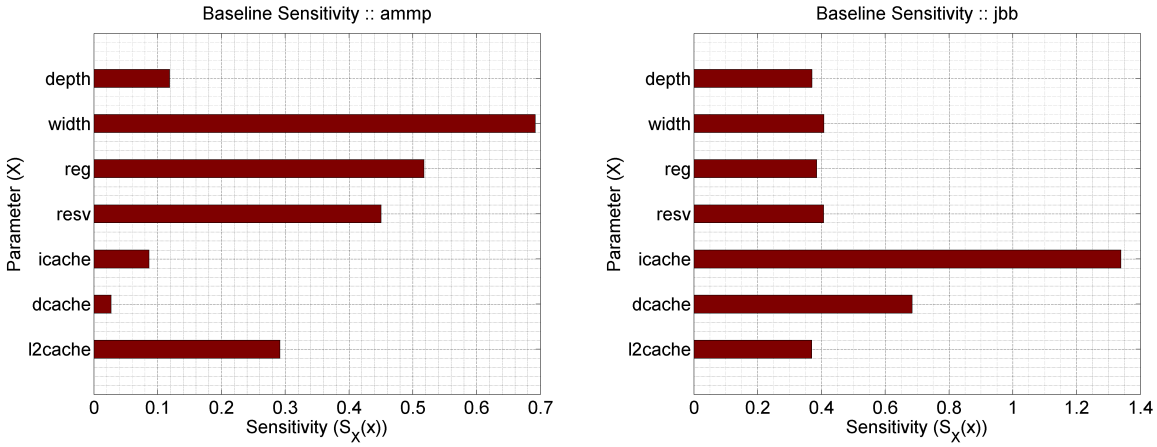

Figure 2. Sensitivity computed at a POWER4-like design for *ammp* (a) and *jbb* (b).

with relatively modest power costs. Sensitivities must be recomputed after every parameter optimization since relative sensitivities change as parameters are tuned.

Figure 2 presents parameter sensitivities for *ammp* and *jbb* evaluated at a design point resembling the POWER4. *Ammp* performs well at the POWER4-like design and there are few opportunities to further tune performance. Observing superscalar width is most sensitive and L1 data/instruction cache sizes are least sensitive, any additional tuning should first enhance width and then re-assess sensitivity. In contrast, sensitivities for *jbb* indicate opportunities to tune the cache hierarchy by increasing L1 cache sizes. Intuitively, this analysis suggests the L1 caches provide much greater performance benefits for every percentage change in power cost. In contrast, large L2 caches deliver performance in a relatively power inefficient manner.

One-dimensional optimization simply shifts sensitivities and efficient tuning should occur at higher dimensions by considering all design parameters simultaneously. This point is well understood by designers in principle and as more complex, high-dimensional topologies are considered, accompanying techniques are needed to quantify topology roughness.

3 Roughness Metrics

The roughness of microarchitectural performance and power topologies has direct implications for modeling and optimization. Rough topologies require greater model flexibility in the form of additional spline knots. Conversely, rough regression models imply an underlying topology at least as rough as the derived model. Rough contours also imply more challenging inputs to optimization heuristics as hills and valleys in the topology increase the likelihood of heuristics converging to local sub-optima. While contour maps are feasible for low-dimensional analysis, roughness metrics extend to higher dimensions and provide a more complete assessment of a topology. By quantifying topology roughness that would otherwise be assessed subjectively in contour maps, these metrics lay the foundation for robust modeling and optimization for rough topologies.

We apply the roughness metrics used by Green and Silverman for smoothed non-parametric regression [4]. R_1 defines a measure of roughness for a one-dimensional function $f(x)$. This measure of roughness is unaffected by the addition of a constant or linear function since R_1 depends on the second derivative. Furthermore, this definition has a basis in mechanical engineering; if a thin piece of flexible wood is bent to the shape of $f(x)$, the leading term in the strain energy is proportional to R_1.

$$
\begin{aligned}
R_1 &= \int_x \left(\tfrac{\delta^2 f}{\delta x^2}\right)^2 dx \\
R_2 &= \int_{x_2}\int_{x_1} \left\{\left(\tfrac{\delta^2 f}{\delta x_1^2}\right)^2 + 2\left(\tfrac{\delta^2 f}{\delta x_1 x_2}\right)^2 + \left(\tfrac{\delta^2 f}{\delta x_2^2}\right)^2\right\} dx_1 dx_2 \\
R_d &= \int_{x_d}\cdots\int_{x_1} \sum \tfrac{m!}{v_1!\ldots v_d!}\left(\tfrac{\delta^m f}{\delta x_1{}^{v_1}\ldots \delta x_d{}^{v_d}}\right)^2 dx_1 \ldots dx_d
\end{aligned}
$$

R_2 is a two-dimensional extension of the one-dimensional definition. Intuitively, R_2 captures roughness since the second derivatives in R_2 are large if the function f exhibits high local curvature. As in R_1, the bending energy of a thin plate is, to first order, proportional to R_2. R_d provides a more general d-dimensional metric based on m-th derivatives, where $2m > d$ [4]. The sum within the integral is over all non-negative integers $v_1, \ldots, v_d$ such that $v_1 + \ldots + v_d = m$. For example, this particular work considers a seven-dimensional design space and computes R_7 based on 4-th derivatives. Throughout, we report relative roughness rankings since the absolute roughness values are not known to have an intuitive physical interpretation in the microarchitectural context.

3.1 Numerical Approximations

Derivatives are approximated using numerical gradients with centered differences at the interior and one-sided differences at the design space boundary.

$$\left.\frac{\delta f}{\delta x}\right|_{x=x^*} \approx \frac{f(x^*+h)-f(x^*-h)}{2h} \quad (2)$$

$$\left.\frac{\delta f}{\delta x}\right|_{x=x^*} \approx \frac{f(x^*+h)-f(x^*)}{h} \quad (3)$$

The integrals are approximated with Riemann sums as shown for one dimension in Equation (4). Riemann sums divide the domain of x into n intervals of equal width $\Delta x = (max[x] - min[x])/n$. This approach identifies approximation points $x_1^*, x_2^*, \ldots, x_n^*$ such that x_i^* lies in the i-th interval.

$$\int_x f(x)dx \approx \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i^*)\Delta x \quad (4)$$

This approximation is valid if f is a continuous, positive function defined over the domain of x. In the microarchitectural context, the function f is a regression model for performance, power, or efficiency. The function f is continuous because our models are constructed with piecewise cubic polynomials using smooth, continuous knot connections. The function is also positive because values for our design metrics are positive. Although f is a regression model that can be evaluated at arbitrarily high resolution, accuracy is constrained by the design space resolution used to construct the model. The limit in Equation (4) suggests higher resolution spaces with parameter values observed at finer granularity may lead to more accurate approximations.

Ideally, the functional mapping $f(x_1, \ldots, x_p)$ would be provided by the simulator. Due to computational costs of extensive simulation, we approximate this function with regression models formulated from sparsely simulated design space samples. This approximation may smooth non-linear, non-monotonic trends in the design space. Assessing the roughness of the underlying microarchitectural design topology through regression models will capture only high-level roughness since more detailed roughness may be obscured by smoothing in least squares fitting. Thus, we should treat these roughness metrics as conservative estimates of the true design space roughness.

3.2 Roughness and Regression Error

Piecewise cubic splines provide the flexibility needed to capture non-linear trends in the design space. Regression models that most utilize this flexibility for a non-linear design topology are likely trying to capture more difficult trends, resulting in greater modeling error. Table 2 ranks benchmarks by seven-dimensional roughness and compares these rankings against median and maximum error. Roughness is ranked in ascending order. For example, *ammp* and *jbb*'s performance topology is least and most rough, respectively. Similarly, error measures are ranked in ascending order. These rankings are positively correlated with coefficients of 0.38 and 0.45 for median performance and power errors, respectively. Similar trends are observed for maximum errors. The greater correlations for power suggests topology roughness is a more significant contributor to power modeling error.

	Perf. Roughness & Error			Power Roughness & Error		
Benchmarks	Rough	Median	Max	Rough	Median	Max
ammp	1	1	1	5	2	7
applu	4	3	5	4	7	5
equake	3	6	6	3	6	4
gcc	6	4	7	6	9	6
gzip	5	7	2	1	3	2
jbb	9	8	8	9	5	3
mcf	2	9	9	2	1	1
mesa	7	2	3	8	8	9
twolf	8	5	4	7	4	8
roughness correlation	1.00	0.38	0.22	1.00	0.45	0.62

Table 2. Correlations between roughness rank and error rank for performance and power. Values increase with rank (*e.g.* roughness rank 1 corresponds to smoothest topology or lowest error).

Given this relationship between roughness and model accuracy, the model specification may be optimized based on its quantified roughness. A rough model likely reflects underlying design space roughness. Design space roughness may be more accurately captured by increasing the knot count, thereby increasing the flexibility of the model specification. Conversely, we might reduce the knot count for parameters that do not exhibit rough trends. This reduced knot count reduces model size (by using fewer terms in the spline transformation), thereby reducing training costs and improving prediction speed. Candidate parameters for knot count tuning may be identified from easily interpreted, low-dimensional contour maps ranked by their roughness.

4 Roughness and Visualization

Contour maps enable efficient and comprehensive visualizations of microarchitectural performance and power topologies, identifying microarchitectural bottlenecks and enabling workload comparisons based on resource requirements at the microarchitectural level. After demonstrating these applications of contour analysis, we quantify roughness and ensure visually rough contours are also highly ranked with respect to our roughness metrics.

This quantitative assessment of roughness eliminates much of the subjectivity in contour analysis and provide a rigorous mechanism for focusing designer attention on the most significant parameters and interesting topologies.

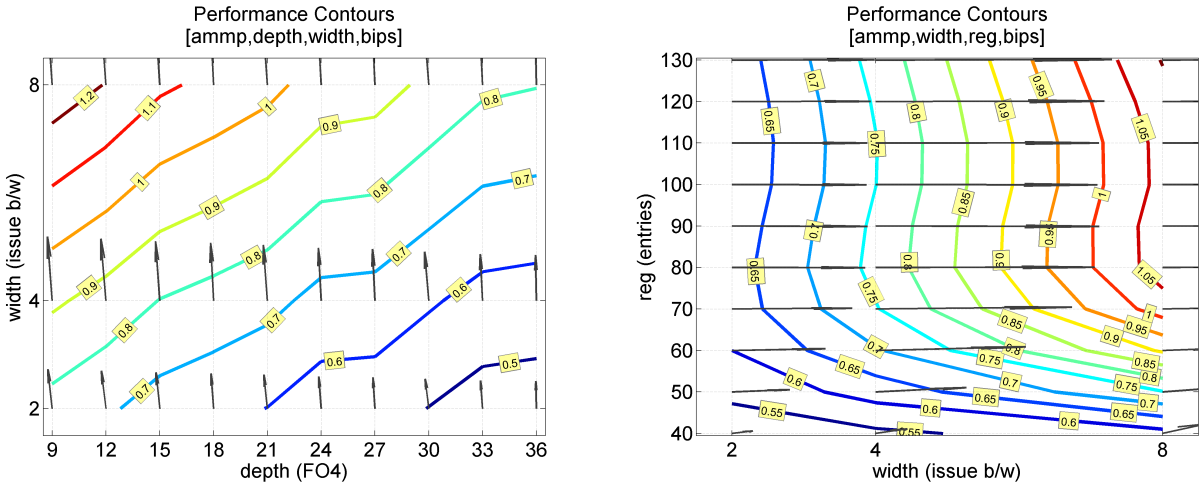

Figure 3. *Ammp* performance contours for depth and width (a), register file and width (b).

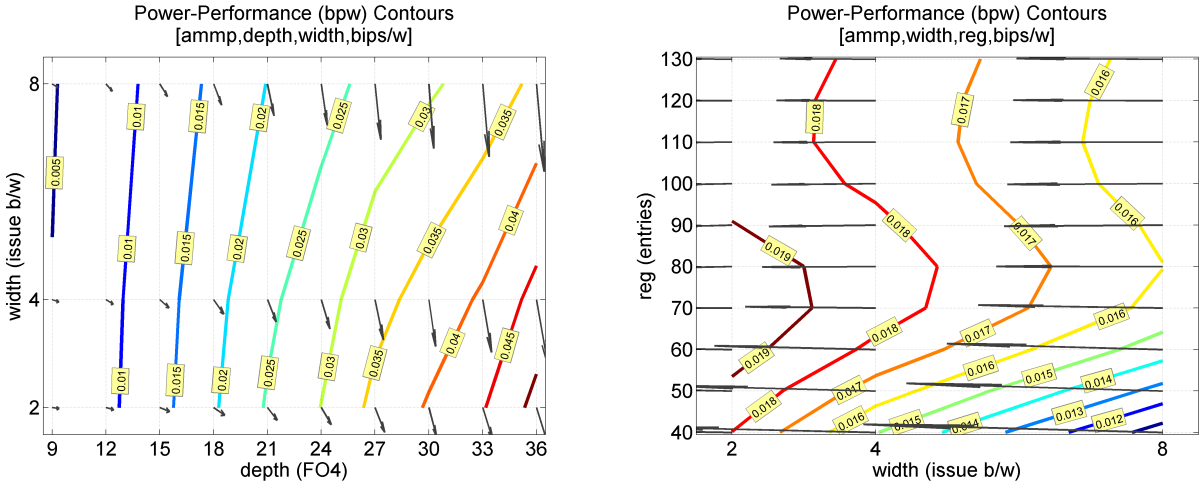

Figure 4. *Ammp* power-performance contours for depth and width (a), register file and width (b).

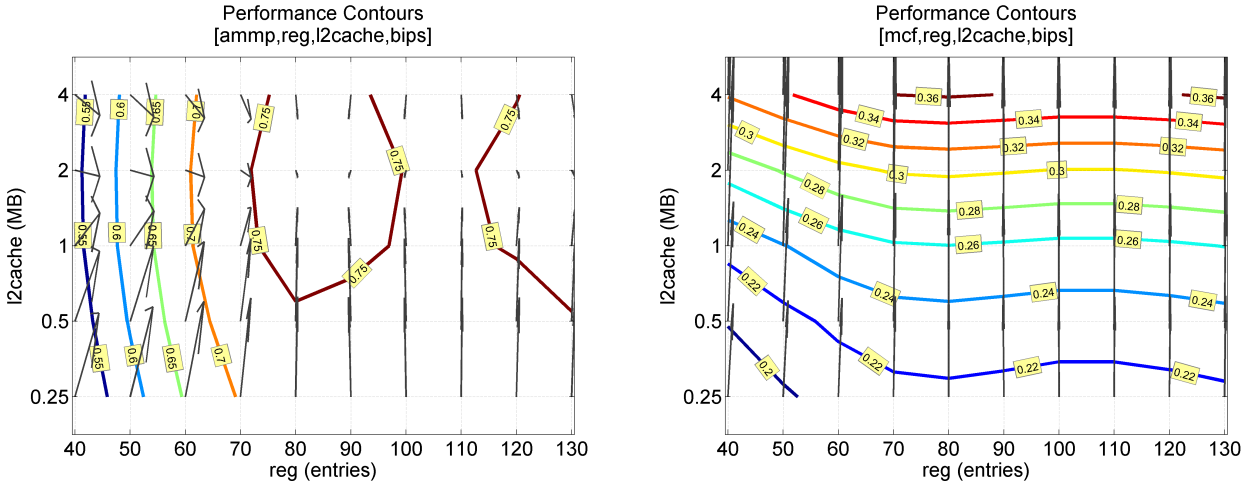

Figure 5. Performance contours for L2 cache and register file for *ammp* (a) and *mcf* (b).

This focus is especially important since contours are often considered for low k-dimensional projections of a p-dimensional space. When p is large and k is small (≤ 3), the number of possible contour maps $C_k^p = \frac{p!}{(p-k)!k!}$ becomes difficult to manage.

4.1 Contour Maps

Contour maps are constructed by exhaustively evaluating regression models for performance and power. Two-dimensional projections of the design space lay the foundation for analysis in higher dimensions. These maps illustrate the topology for a metric of interest, thereby revealing non-linearities that may arise from non-monotonic trends in the topology (hills, valleys) or diminishing marginal returns with increasing resource sizes (plateaus).

The visualization also identifies a path to optimality from any initial design point. Figure 3(a) presents *ammp's* performance contours for a two-dimensional projection of pipeline depth and superscalar width from the seven-dimensional design space of Table 1, revealing a clear path to optimal performance in the direction of deeper, wider pipelines.[2] This particular space favors balanced pipeline dimensions, indicating depth and width should increase together to most effectively achieve higher performance. While we plot contours that span all width values, in practice, we are likely to examine only feasible design points in the contour maps (*e.g.*, 2-, 4-, 8-wide designs).

In multiple dimensions, gradients produce vector fields from scalar fields. We compute these vector fields using numerical gradients and visualize them using arrows in the contour maps. Note that gradients point orthogonally between contour levels since this is the steepest approach to higher levels. Furthermore, contour levels closely spaced together imply steeper slopes since smaller microarchitectural changes are required to achieve the same performance or efficiency increase. Thus, closely spaced contour levels result in larger gradient magnitudes. These vector fields complement the contour maps and enable designers to identify paths to design optima more quickly.

4.2 Bottleneck Analysis

The path to optimality reveals the changing source of bottlenecks for the metric of interest. Since contour roughness often arises from interesting parameter interactions and shifting bottlenecks, we can use roughness metrics to rank and choose interesting two-dimensional contours from the C_2^p possibilities for bottleneck analysis. Figure 3(b) presents *ammp's* performance contours for a two-dimensional projection of the register file and superscalar width. We observe two distinct regions divided horizontally by the 80-entry register file. The *ammp* benchmark experiences significant register pressure when running on designs in the lower region of this map. Performance in this lower region is most effectively optimized by increasing both register file sizes and superscalar width. In contrast, we observe diminishing marginal returns in performance from larger register file sizes in the upper region of the map. This is illustrated by vertical contour levels in which changing register file sizes cannot lead to higher performance. The register file is no longer a bottleneck in this region and other design parameters should be analyzed.

[2]Depth is measured in fan-out-of-four delays per logic stage and smaller FO4's correspond to deeper pipelines with a larger number of short logic stages.

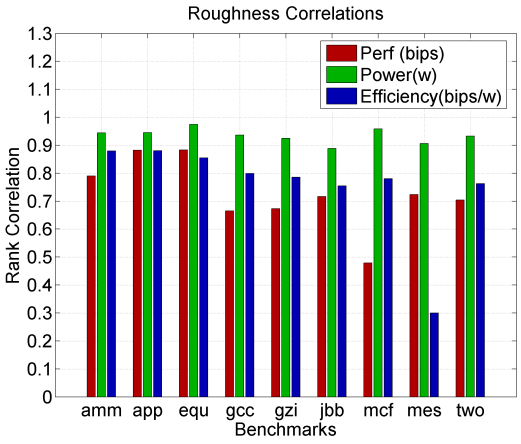

Figure 6. Correlation between roughness rank and contour range.

Power metrics are easily included into the analysis. The power contours alone are less interesting as power often increases monotonically with increases in microarchitectural resources. Figure 4 considers the same projections of Figure 3 using a $bips/w$ efficiency metric. Figure 4(a) examines trade-offs for pipeline dimensions, suggesting this design space favors shallower, narrower pipelines once the optimization metric accounts for the power costs of deeper, wider pipelines. Pipeline depth is the primary efficiency bottleneck for the more aggressive designs as illustrated by the vertical contours in the 9 to 15 FO4 region. As we consider increasingly shallow pipelines, superscalar width begins to limit efficiency and we see a more pronounced trend toward fewer instructions issued per cycle (*i.e.*, issue bandwidth from 8 to 2) in the 24 to 36 FO4 region.

Figure 4(b) illustrates performance and power trade-offs for the register file and superscalar width. The $bips/w$ efficiency metric strongly favors narrower pipelines. This metric also more clearly identifies the optimal register file size between 60 and 90 entries based on the power costs of over-provisioning this resource for the *ammp* benchmark. In contrast, the performance analysis of Figure 3(b) favors more than 80 entries and does not capture power costs that make 130-entry register files unattractive.

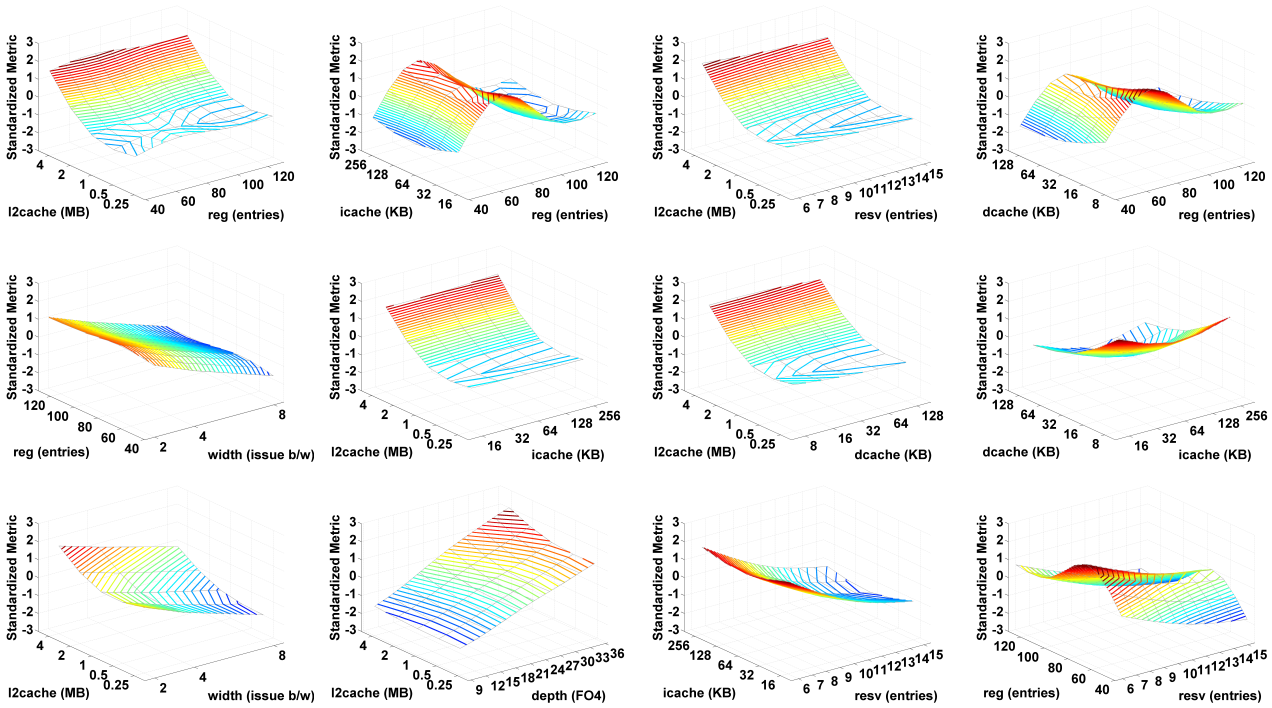

Figure 7. Standardized $(bpw - E[bpw])/SD[bpw]$ contours for *mcf*; ranked by R_2 (left to right, top to bottom).

4.3 Workload Characterization

Roughness and bottleneck analysis with contour maps also enables workload comparisons based on their resource requirements at the microarchitectural level. As before, roughness rankings identify interesting contours for comparison. Figure 5 illustrates such a comparison between *ammp* and *mcf* performance by examining demands on the register file and L2 cache. As observed earlier, the register file is a bottleneck for *ammp*. The vertical contour levels indicate negligible performance advantages from larger caches until register pressure has been relieved with at least 80 entries. Once the register file contains 80 entries, performance is maximized by cache sizes larger than 1 MB. In contrast, the horizontal contour levels for *mcf* reveal significant performance advantages from larger caches, indicating the workload is relatively memory bound. We also observe more subtle indicators that 80-entry register files are optimal for *mcf*. The vertical dips in the horizontal contour levels all occur at 70 or 80 physical registers, indicating greater performance at this file size for any L2 cache size.

While microarchitecture independent workload characterizations examine and assess performance through fundamental program characteristics such as instruction mix and register traffic, comparing comprehensive contour maps enable a more direct analysis based on computational resource requirements at the microarchitectural level. For example, microarchitecture independent metrics to assess instruction level parallelism (ILP) might include the number of independent instructions within an instruction window [14]. In contrast, a contour map of pipeline dimensions would make an assessment of ILP based on the optimality of various pipeline depths and superscalar widths, thereby revealing a workload's characteristics and its direct implications for microarchitectural design. Similarly, a microarchitecture independent assessment of instruction mix might reveal an application's relative memory intensity, but contour maps of the L2 cache would also reveal this trend while exposing the implications of memory intensity to cache design.

4.4 Roughness and Contours

Prior work demonstrated analysis with microarchitectural contours, but did not provide any objective mechanism for focusing designer attention [13]. In contrast, we propose roughness metrics for identifying interesting contours. We assess the effectiveness of our roughness metrics by computing R_2 for contour maps and validating against graphically observed roughness. This analysis reveals two contributors to larger roughness values: range and variability. Range is the difference between minimum and maximum metric values in a given contour plot. Variability is curvature in the topology, manifested in contour non-linearity or non-monotonicity. Empirically, we observe a significant range component in the roughness metrics. For example, we find R_2(Figure 4(a)) $>$ R_2(Figure 4(b)) and R_2(Figure 5(b)) $>$ R_2(Figure 5(a)) due to range differences. We construct all two-dimensional contours from

Trials	Number of random starting points. Gradient ascent returns best result across all trials.
Iterations	Number of steps required before a trial converges to a point with no significantly better neighbors.
Deficiency	Difference between optima from gradient ascent and exhaustive search.

Table 3. Gradient ascent definitions.

the seven-dimensional design space and rank them by R_2 roughness. Figure 6 plots the correlation between roughness and range rankings of these contours. The large correlation coefficients (most exceeding 0.65) suggest a strong relationship between the range and roughness of contour maps for various benchmarks and design metrics.

We can isolate contributions from the variability component by standardizing all contours to comparable ranges. We subtract the mean and divide by the standard deviation to produce similar ranges primarily between -3.0 and 3.0, the dominant range of a standard normal distribution. Figure 7 presents the 12 roughest *mcf* contours from the $C_2^7 = 21$ possible maps ranked by decreasing roughness from left to right, top to bottom. We observe qualitative differences in non-linearity and non-monotonicity as we progress from highly rough to least rough contours. The 12-th contour appears rougher than its ranking implies, but is an exception and not the general case. Excluding this contour from the original 21, all rankings corroborate graphically observed roughness. We observe increasing roughness in contours 1 to 11 and very smooth trends in contours 13 to 21 (not shown). Hills, valleys and plateaus are frequently observed for rough contours while regular vector fields and vertical, horizontal contour levels dominate smooth contours. This qualitative validation of roughness metrics against observed topologies in the easily visualized two-dimensional contours builds confidence in the proposed metrics. Thus, roughness metrics enable quantitative and objective comparisons of contour range and variability.

5 Roughness and Optimization

The idea of traversing hills and valleys to find optima in two dimensions extends naturally to higher dimensions. Gradient ascent, also known as steepest ascent or hill climbing, begins at an initial point and steps toward a local maximum by moving in the direction of the steepest change specified by the gradient. In practice, the gradient is approximated by identifying a move direction toward a point's best neighbor in the p-dimensional space. Since the heuristic may identify a local maximum, robust implementations iterate and start at different initial points to ensure a reported maximum is reached consistently, thus increasing the likelihood of an accurate approximation to the global maximum.

Intuitively, greater topology roughness will reduce the effectiveness of search heuristics. To assess this link, we must first establish measures of effectiveness. The computational efficiency of regression models provide an opportunity to assess gradient ascent in a manner previously not possible by

```
for (t = 1 to max_trials)
   x_old = <random initial design>
   e_old = EVAL(x_old)
   term_flag = 0;

   while(term_flag == 0) {
      // evaluate optimal neighbor
      N = neighborset(x_old);
      x_new = argmax{N | EVAL(N)};
      e_new = EVAL(x_new);

      // compare new against previous
      // and terminate if difference is less
      // than threshold (e.g., 1.001)
      if (e_new/e_old < 1.001) {
         term_flag = 1;
      }
      x_old = x_new; e_old = e_new;
    }
    metric(t) = e_old; design(t) = x_old;
}

// identify optimal design across trials
m = argmax(t | metric(t));
return(metric(m), design(m));
```

Figure 8. Gradient ascent implementation.

1. comparing gradient ascent results against the true global optimum from exhaustively evaluating models.
2. assessing the likelihood of identifying the true global optimum across multiple trials of gradient ascent beginning at random starting points.
3. assessing the distribution of convergence times (*i.e.*, path length of a trial) across multiple trials of gradient ascent beginning at random starting points.

5.1 Implementation

Figure 8 outlines our standard implementation of gradient ascent. Each trial begins at a randomly chosen design. Each iteration in the `while` loop of this trial will exhaustively compare the current design against all neighboring designs, identifying and selecting the best neighbor for the next iteration of the loop. A trial terminates if the new design's metric differs from old design's metric by less than some threshold (*e.g.* 0.1 percent). At termination, the maximum identified by the trial is logged into a list of search results. Gradient ascent returns the best of these results after `max_trials`.

We implement gradient ascent to search the performance ($bips$) and efficiency ($bips/w$) topologies. The appropriate regression models are used to compute `EVAL(N)` for each neighbor. We define the set of neighbors as all designs that can be reached by changing any combination of parameter values by at most one step where parameter step sizes are specified in the range column of Table 1. Define each neighbor in a p-dimensional design space using a p-element vector where each element can take one of three

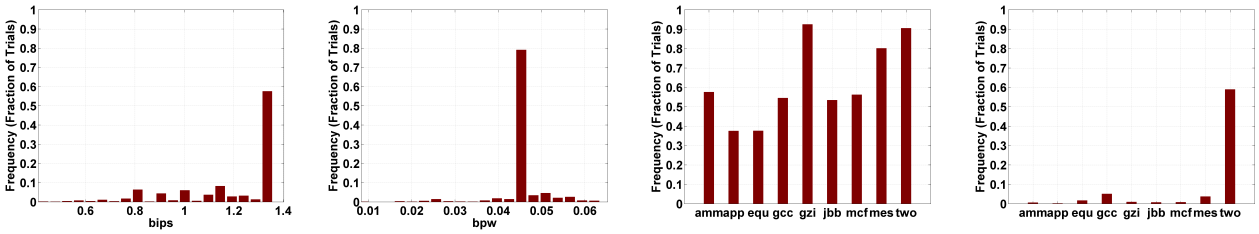

Figure 9. Histogram of *ammp* gradient ascent results for performance (a), efficiency (b). Percentage of trials with performance (c), efficiency (d) in the optimal bin.

	$bips$		$bips/w$	
Benchmark	Def (%)	Trials	Def (%)	Trials
ammp	0.00	100	0.00	400
applu	0.00	100	0.81	500
equake	0.00	100	12.94	500
gcc	0.00	100	2.72	200
gzip	0.00	100	5.92	300
jbb	0.00	100	1.89	900
mcf	0.00	100	0.00	400
mesa	0.00	100	0.97	100
twolf	0.00	100	0.91	300

Table 4. Summary of gradient ascent results.

Benchmarks	Rough	Deficiency	Trials	Iterations
ammp	4	1	5	6
applu	2	3	7	2
equake	3	9	8	1
gcc	5	7	2	8
gzip	8	8	3	9
jbb	7	6	9	3
mcf	1	2	6	7
mesa	9	4	1	4
twolf	6	5	4	5
roughness correlation	1.00	0.35	-0.52	0.20

Table 5. Correlations between roughness rank and rank by gradient ascent (1) deficiency, (2) trial, (3) iterations. Values increase with rank (*e.g.* roughness rank 1 corresponds to smoothest topology.)

values: step-up, step-down, unchanged. Points at the interior of the design space have 3^p such vectors, resulting in $2,187$ neighbors for our seven-dimensional design space. The computational efficiency of our performance and power regression models enable us to evaluate such a large number of predictions per iteration until the trial converges. This process is repeated for `max_trials`=1,000.

5.2 Evaluation

Since each trial begins at a randomly chosen initial design, the search result varies from trial to trial. Figure 9(a-b) captures the distribution of identified $bips$ and $bips/w$ values from 1,000 representative *ammp* trials. There are obvious modes for both metrics, but Figure 9(a) suggests the $bips$ topology is more effectively explored by gradient ascent. Although sub-optimal local maxima are occasionally identified, 57.6 percent of trials converged to the same optimal value of 1.34 $bips$. In contrast, the $bips/w$ mode in Figure 9(b) is more pronounced with 79.1 percent of trials converging to 0.045 $bips/w$, but 14.0 percent of trials identify more efficient designs ranging from 0.048 to 0.062 $bips/w$.

The performance and power trade-offs likely increase the non-linearity of the $bips/w$ topology, producing modes below the max. Figure 9(c-d) summarizes the $bips$ and $bips/w$ differences across the suite of benchmarks by identifying the fraction of trials that report the best results (*i.e.*, the right-most bar of Figure 9(a-b) for each benchmark). 37.5 percent of $bips$ trials report the best result while, for most benchmarks, less than 1.0 percent of $bips/w$ trials do so. To correctly identify the global maximum, at least one gradient ascent trial must find a path to this optimum. A sub-optimal mode reduces the likelihood of such a trial and implies a larger number of trials is needed to find a global maximum with confidence.

Gradient ascent is a search heuristic that may identify local maxima. Multiple trials increase the likelihood of finding the global maximum, but this result is not guaranteed for non-monotonic topologies. Table 4 summarizes gradient ascent effectiveness, assessing stable deficiencies and trial counts needed for stabilization. Deficiency quantifies the difference between the results of gradient ascent and those of exhaustive search using regression models. Deficiency may decrease with trial count, but is often observed to stabilize at 400 to 500 trials when optimizing $bips/w$. In contrast, gradient ascent achieves zero deficiency when traversing the $bips$ topologies.

The number of iterations per trial illustrates gradient ascent's ability to converge to a maximum quickly when starting at various random points in the topology. Figure 10 uses boxplots to examine the distribution of times to convergence for 1,000 trials when optimizing $bips$ and $bips/w$. Figure 10(a) indicates the median number of iterations per trial is 10 and most trials converge and exit the `while` loop in fewer than 20 iterations. In contrast, the $bips/w$ topology of Figure 10(b) may be more difficult to traverse with a median convergence time of 9 iterations for five benchmarks (*applu, equake, jbb, mesa, twolf*), but a median convergence time of 30 for the remaining benchmarks (*ammp, gcc, gzip, mcf*). Convergence times exceeding 20 iterations is common for these latter benchmarks.

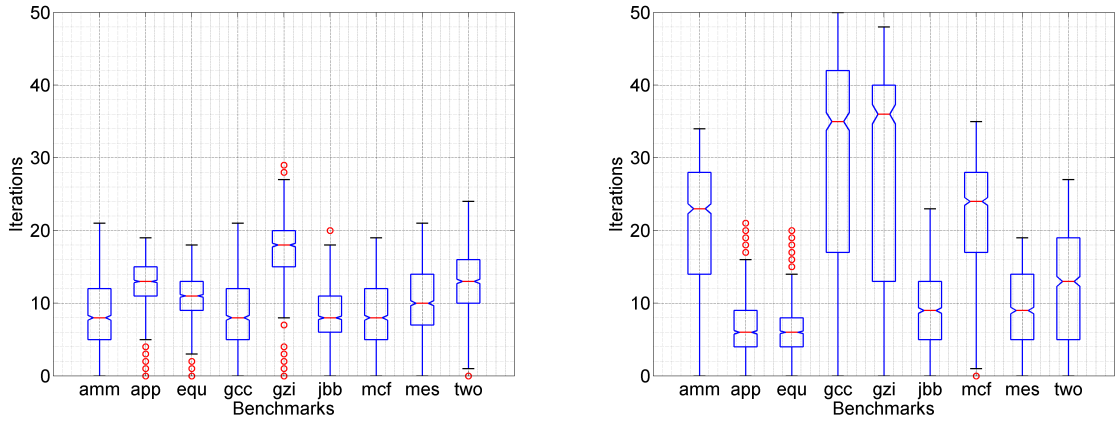

Figure 10. Distribution of gradient ascent convergence times for 1,000 trials performance (a) and efficiency (b).

5.3 Roughness and Heuristic Efficacy

Roughness may impact the effectiveness of optimization heuristics that traverse the topology. Table 5 quantifies roughness contributions to deficiencies in gradient ascent results. The benchmarks are ranked by their $bips/w$ deficiencies relative to optima from exhaustive search. The positive correlation of 0.35 suggests deficiency increases with roughness. If a model captures a rougher topology, gradient ascent is more likely to identify local maxima without realizing the global maximum. For example, Table 5 indicates *ammp* and *mcf* are relatively smooth with $bips/w$ roughness rankings of 4 and 1. Both benchmarks identify the global maximum with no deficiency as shown in Table 4. In contrast, *gzip* has roughness and deficiency rankings of 8, suggesting rough $bips/w$ topologies may result in a greater number of reported sub-optima from gradient ascent. This general trend is confirmed across all benchmarks with only one significant deviation in *equake* which has a relatively smooth $bips/w$ topology but exhibits significant gradient ascent deficiencies of 12.9 percent.

The number of predictions performed in gradient ascent is affected by the number of required trials to get the best overall result (*i.e.*, lowest deficiency) and the number of iterations needed for each trial to converge to its particular result. Table 5 indicates roughness is negatively correlated with the number of trials needed to identify stable deficiencies. Taken with its positive correlation with deficiency, rough topologies may result in deficient results that cannot be mitigated by more trials. Lastly, we observe a positive correlation between roughness and iterations per trial. This correlation indicates each traversal of a rougher space may require more steps before reaching a trial's maximum.

We might use the relationship between roughness and gradient ascent deficiency to improve the search heuristic. Additional trials in gradient ascent may reduce deficiencies by increasing the likelihood of an initial design chosen near the global maximum. Alternatively, we could consider simulated annealing or other stochastic variants to gradient ascent. Instead of evaluating all neighboring designs, simulated annealing will randomly select a neighbor, accepting a sub-optimal neighbor with some non-zero probability even with no benefit to the target metric. This probabilistic approach enables the search to escape sub-optimal local maxima, an important capability for rough topologies.

6 Related Work

Zyuban, *et al.*, propose measures of hardware and voltage intensity to quantify compromises between energy and delay resulting from circuit-level tuning and voltage scaling, respectively [20]. Intensity is computed as $\frac{D}{\delta D}\frac{\delta E}{E}$ where D is delay and E is energy. Intensity is used to derive conditions for optimal microarchitectural power-performance from mathematical relations. Markovic, *et al.*, derive sensitivity $\frac{\delta E/\delta X}{\delta D/\delta X}$ for tunable circuit parameters X such as gate sizing, supply voltage, and threshold voltage [11]. Optimal values for circuit parameters are those that equalize sensitivity. In contrast, we use statistical inference and heuristics at the macro block level.

Yi, *et al.*, identify significant parameters using Plackett-Burman design matrices [17]. Oskin, *et al.*, use statistical and symbolic simulation to construct contour maps [13]. We identify significant parameters using contour maps and construct these maps with predictive regression models. This allows us to quickly consider many contours in an exploratory manner.

Eyerman, *et al.*, apply synthetic trace simulation with various heuristics to search for global optima within a design space [3]. These authors compute gradient ascent deficiency relative to results from other heuristics while we assess effectiveness relative to exhaustive search with regression models. Eyerman, *et al.*, evaluate other techniques

including genetic algorithms and tabu search. We analyze gradient ascent since it is widely known and is likely the first method tried by a typical user.

Ipek, *et al.*, and Joseph, *et al.*, separately predict the performance of design spaces with automated neural networks trained by gradient descent and predicted by nested weighted sums [2, 7]. The approach we adopt from Lee and Brooks requires greater statistical analysis when constructing spline-based regression models, but is based on numerical linear algebra and may be more computationally efficient [9]. In contrast, Karkhanis, *et al.*, propose analytical models for superscalar design optimization [8].

7 Conclusions

We apply recent advances in statistical inference and machine learning to close the divide between microarchitectural simulation and classical optimization. We compute roughness metrics to assess the impact of design space non-linearity and non-monotonicity on analysis and optimization. In particular, we consider the effects of roughness on spline-based regression model error, the consistency of roughness when compared against graphically observed trends in contour maps, and the implications of rough topologies for optimization heuristics that traverse the design space. Overall, roughness metrics advance our understanding of the design space, guiding the design of robust predictive models and optimization heuristics.

Acknowledgements

This work is supported by National Science Foundation grant CCF-0048313 (CAREER), Intel, and IBM. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, Intel, or IBM.

References

[1] D. Brooks, P. Bose, V. Srinivasan, M. Gschwind, P. G. Emma, and M. G. Rosenfield. New methodology for early-stage, microarchitecture-level power-performance analysis of microprocessors. *IBM Journal of Research and Development*, 47(5/6), Oct/Nov 2003.

[2] E.Ipek, S.A.McKee, B. de Supinski, M. Schulz, and R. Caruana. Efficiently exploring architectural design spaces via predictive modeling. In *ASPLOS-XII: Architectural support for programming languages and operating systems*, October 2006.

[3] S. Eyerman, L. Eeckhout, and K. D. Bosschere. Efficient design space exploration of high performance embedded out-of-order processors. In *Design, Automation, and Test in Europe*, March 2006.

[4] P. Green and B. Silverman. *Nonparametric regression and generalized linear models: A roughness penalty approach*. Monographs on Statistics and Applied Probability, 1994.

[5] F. Harrell. *Regression modeling strategies*. Springer, 2001.

[6] V. Iyengar, L. Trevillyan, and P. Bose. Representative traces for processor models with infinite cache. In *Proceedings of the 2nd Symposium on High Performance Computer Architecture*, February 1996.

[7] P. Joseph, K. Vaswani, and M. J. Thazhuthaveetil. A predictive performance model for superscalar processors. In *Micro-39: International Symposium on Microarchitecture*, December 2006.

[8] T. Karkhanis and J. Smith. Automated design of application specific superscalar processors: an analytical approach. In *International Symposium on Computer Architecture*, June 2007.

[9] B. Lee and D. Brooks. Accurate and efficient regression modeling for microarchitectural performance and power prediction. In *ASPLOS-XII: International Conference on Architectural Support for Programming Languages and Operating Systems*, October 2006.

[10] B. Lee and D. Brooks. Illustrative design space studies with microarchitectural regression models. In *HPCA'07: International Symposium on High-Performance Computer Architecture*, February 2007.

[11] D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, and R. Broderson. Methods for true energy-performance optimization. *IEEE Journal of Solid-State Circuits*, 39(8), August 2004.

[12] M. Moudgill, J. Wellman, and J. Moreno. Environment for powerpc microarchitecture exploration. *IEEE Micro*, 19(3):9–14, May/June 1999.

[13] M. Oskin, F. Chong, and M. Farren. Hls: Combining statistical and symbolic simulation to guide microprocessor designs. In *ISCA-27: International Symposium on Computer Architecture*, June 2000.

[14] A. Phansalkar, A. Joshi, L. Eeckhout, and L. John. Measuring program similarity: experiments with spec cpu benchmark suites. In *ISPASS05: International Symposium on Performance Analysis of Systems and Software*, March 2005.

[15] R Development Team. *R Language Definition*.

[16] P. Shivakumar and N. Jouppi. An integrated cache timing, power, and area model. In *Technical Report 2001/2, Compaq Computer Corporation*, August 2001.

[17] J. Yi, D. Lilja, and D. Hawkins. A statistically rigorous approach for improving simulation methodology. In *HPCA: International Symposium on High-Performance Computer Architecture*, Feb 2003.

[18] V. Zyuban. Inherently lower-power high-performance superscalar architectures. In *Ph.D. Thesis, University of Notre Dame*, March 2000.

[19] V. Zyuban, D. Brooks, V. Srinivasan, M. Gschwind, P. Bose, P. Strenski, and P. Emma. Integrated analysis of power and performance for pipelined microprocessors. *IEEE Transactions on Computers*, Aug 2004.

[20] V. Zyuban and P. Strenski. Balancing hardware intensity in microprocessor pipelines. *IBM Journal of Research and Development*, 47(5/6), Oct/Nov 2003.