Effective and Efficient Microprocessor Design Space Exploration Using Unlabeled Design Configurations

TIANSHI CHEN, YUNJI CHEN, and QI GUO, Institute of Computing Technology, Chinese Academy of Sciences
ZHI-HUA ZHOU, Nanjing University
LING LI and ZHIWEI XU, Institute of Computing Technology, Chinese Academy of Sciences

Ever-increasing design complexity and advances of technology impose great challenges on the design of modern microprocessors. One such challenge is to determine promising microprocessor configurations to meet specific design constraints, which is called Design Space Exploration (DSE). In the computer architecture community, supervised learning techniques have been applied to DSE to build regression models for predicting the qualities of design configurations. For supervised learning, however, considerable simulation costs are required for attaining the labeled design configurations. Given limited resources, it is difficult to achieve high accuracy. In this article, inspired by recent advances in semisupervised learning and active learning, we propose the COAL approach which can exploit unlabeled design configurations to significantly improve the models. Empirical study demonstrates that COAL significantly outperforms a state-of-the-art DSE technique by reducing mean squared error by 35% to 95%, and thus, promising architectures can be attained more efficiently.

Categories and Subject Descriptors: C.1.0 [**Processor Architectures**]: General; B.2.2 [**Arithmetic and Logic Structures**]: Performance Analysis and Design Aids—*Simulation*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Design space exploration, microprocessor design, simulation, machine learning, semisupervised learning, active learning

ACM Reference Format:
Chen, T., Chen, Y., Guo, Q., Zhou, Z.-H., Li, L., and Xu, Z. 2013. Effective and efficient microprocessor design space exploration using unlabeled design configurations. ACM Trans. Intell. Syst. Technol. 5, 1, Article 20 (December 2013), 18 pages.
DOI: http://dx.doi.org/10.1145/2542182.2542202

1. INTRODUCTION

When designing a microprocessor, the first and probably the most important step is to decide appropriate design configurations satisfying different performance/power/temperature/reliability constraints, which is called as *Design Space Exploration* (DSE).

This work was partially supported by the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA06010400), the NSFC (61073097, 61021062, 61100163, 61173006, 61133004, 61003064, 60921002), the 973 Program (2010CB327903, 2011CB302502, 2011CB302803), the 863 Program (2012AA012202), and the National S&T Major Project of China (2009ZX01028-002-003, 2009ZX01029-001-003, and 2010ZX01036-001-002).
Authors' addresses: T. Chen, Y. Chen, Q. Guo (corresponding author, joyguoqi@gmail.com), L. Li, and Z. Xu, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, 100190, China; Z.-H. Zhou, National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210046, China.

DOI: http://dx.doi.org/10.1145/2542182.2542202

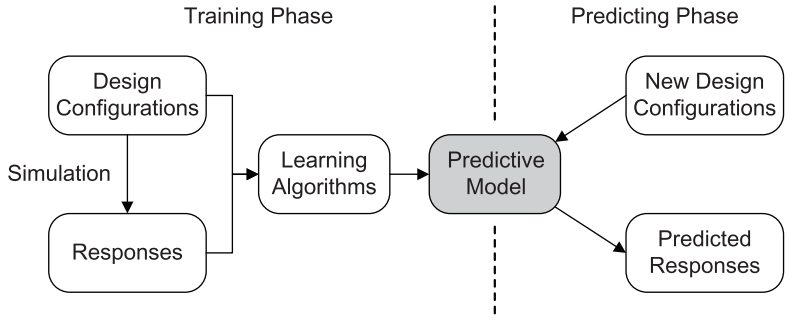

Fig. 1. A framework of predictive modeling for DSE.

It has become a great challenge to computer architects, since the size of design space grows exponentially with the number of interactive design parameters (e.g., cache size, queue size, issue width and so on), and the simulation required by evaluating the quality of each configuration of design parameters is quite time-consuming. Moreover, the difficulty of DSE task is further exacerbated by the increasing amount and complexity of computer workloads with significantly different characteristics.

Traditionally, computer architects employed large-scale cycle-accurate architectural simulations on representative benchmarks to explore the design space. However, time-consuming simulations make it intractable to explore the entire design space. For instance, during the design of Godson-3, which is a 16-core chip-multiprocessor (CMP) with a reconfigurable architecture [Hu et al. 2009], it takes several weeks to simulate only one design configuration on SPEC CPU2000 benchmark suite. To reduce the simulation costs, several fast simulation approaches were proposed to reduce the number of simulated instructions with respect to each design configuration [Hamerly et al. 2006; Genbrugge and Eeckhout 2009; Joshi et al. 2006]. However, the number of design configurations to simulate is still quite large. To reduce the number of simulated configurations and thus reduce the overall simulation costs, predictive modeling was proposed [Joseph et al. 2006; İpek et al. 2006; Lee et al. 2008]. As illustrated in Figure 1, a predictive modeling approach contains two phases, that is, training phase and predicting phase. In the training phase, some design configurations are simulated. Along with the corresponding responses (e.g., performance or energy response) obtained by simulations, these labeled design configurations are utilized to train a regression model that characterizes the relationship between the design parameters and processor responses. In the predicting phase, such a regression model is employed to predict the responses of new design configurations that are not involved in the training set. Since simulations are only required in the training phase, predictive modeling is relatively efficient in comparison with traditional approaches. However, considerable simulation costs are required to attain the labeled design configurations for supervised learning, which encumbers the models from achieving high prediction accuracies given limited computational resources and stringent design-to-market pressure. In fact, the design configurations that have not been simulated may also be effective in enhancing the prediction accuracy of a regression model, which are overlooked by previous investigations on DSE.

To circumvent these deficiencies of previous techniques, in this article, we propose the COAL (CO-training regression tree with Active Learning) approach for the challenging DSE problem. The key intuition is that similar architectural configurations would behave similarly, and thus, some unlabeled design configurations can be labeled for enhancing the prediction accuracy. Generally speaking, COAL works in the cotraining

style [Blum and Mitchell 1998], which is an algorithm falling into the disagreement-based learning paradigm [Zhou and Li 2010], where two learners label unlabeled instances for each other. At each iteration, COAL also simulates the most informative unlabeled instance on which the two learners exhibit the largest disagreement, which is inspired by active learning. To enable the learned model to be comprehensible for computer architects and designers, COAL employs M5P regression trees [Wang and Witten 1997] as the base learner. At the beginning of a typical DSE procedure, COAL initializes two regression trees by using an initial data set containing several labeled design configurations. After that, COAL starts the semisupervised active learning process consisting of a number of iterations. In every iteration, each regression tree is refined by not only the design configuration labeled by the other regression tree, but also the configuration newly simulated by the processor simulator. The key issue here is how to select appropriate unlabeled configurations to label (semisupervised learning) or simulate (active learning), which will be elaborated in Section 3.

To demonstrate the effectiveness of COAL, we conduct comprehensive experiments to explore a typical superscalar microprocessor design space. Experiments show that, given the same simulation budget to attain the labels of training design configurations, COAL can reduce by 35–95% mean squared error of the state-of-the-art ANN DSE technique. Furthermore, we also conduct experiments to demonstrate that COAL can achieve better prediction accuracy compared with pure cotraining semisupervised learning and pure active learning approaches, which further demonstrates the superiority of COAL approach.

The rest of the article proceeds as follows. Section 2 introduces some related work. Section 3 presents our COAL approach. Section 4 reports our empirical results. Section 5 concludes this article.

2. RELATED WORK

2.1. Exploiting Unlabeled Data

In many real-world applications, the overhead of collecting a labeled example is very expensive, while a large number of unlabeled instances are very easy to obtain. To exploit the unlabeled instances and improve the accuracies of learners, Semi-Supervised Learning (SSL) and Active Learning (AL) have been intensively investigated by the machine learning community.

SSL is a mainstream methodology for exploiting unlabeled data to improve the prediction accuracy. Generally, SSL can be classified into four categories [Zhou and Li 2010], that is, generative methods [Fujino et al. 2005; Miller and Uyar 1997; Nigam et al. 2000], S3VMs (Semi-Supervised Support Vector Machines) [Xu and Schuurmans 2005; Joachims 1999; Chapelle and Zien 2005], graph-based methods [Zhu et al. 2003; Zhou et al. 2004], and disagreement-based methods [Blum and Mitchell 1998; Zhou and Li 2010]. Generative methods conduct maximum likelihood estimation to determine the parameters of models, where the labels of unlabeled data are treated as missing values. S3VMs usually utilize unlabeled data to adjust the decision boundary built from labeled examples. In graph-based methods, the SSL problem can be addressed by propagating the label information in a graph constructed from labeled and unlabeled data where each node corresponds to one instance. The key of disagreement-based methods is to generate multiple learners, let them collaborate to exploit unlabeled data, and maintain a disagreement among the base learners. This line of research started by Blum and Mitchell [1998]'s seminal work on cotraining, which is a multi-view learning algorithm. Zhou and Li [2005a] proposed a Semi-Supervised Regression (SSR) approach, COREG, which employs two kNN regressors to conduct the data labeling and the predictive confidence estimation. COREG does not require multiviews, it utilizes

kNN as the base regressor since it is easy to update and smoothly consistent with the manifold assumption of SSL. In COREG, the most confidently labeled example is determined as the one which makes the regressor most consistent with labeled data. Though studies of disagreement-based SSL approaches started from multiview setting [Blum and Mitchell 1998], there are many successful algorithms that do not require multiviews [Zhou and Li 2005a, 2010]. Recently, theoretical studies showed that multiview is not really needed for disagreement-based algorithms [Wang and Zhou 2007, 2010b].

As another well-known methodology of leveraging unlabeled data, AL improves the prediction accuracy by actively querying the oracle (in the context of DSE, the oracle refers to the simulator) the labels of some unlabeled instances. According to the concrete way of selecting the instance-to-query, existing approaches of AL can roughly be categorized into three types. The first type of approaches query the label of the most informative instance which is expected to reduce the uncertainty of a statistical model to the most. For example, in cotesting [Muslea et al. 2000], a multiview active learning algorithm, the most informative instance is the unlabeled instance on which the two learners trained from different views exhibit the largest disagreement. Query-by-committee [Seung et al. 1992; Freund et al. 1997], uncertainty sampling [Lewis and Catlett 1994] and committee-based sampling [Dagan and Engelson 1995] also fall into this type. The second type of approaches query the label of the most representative instance which is expected to characterize the pattern in the data to the best, where the most representative instance is often selected based on the cluster structure of unlabeled data [Nguyen and Smeulders 2004; Dasgupta and Hsu 2008]. The third type of approaches query instances that are both informative and representative [Donmez et al. 2007; Huang et al. 2010]. Theoretical aspects of active learning have been studied by many researchers [Balcan et al. 2006; Hanneke 2007; Dasgupta et al. 2007], and recently it has been shown that multiview active learning is able to achieve an exponential improvement of sample complexity in setting close to real tasks [Wang and Zhou 2010a].

Zhou et al. [2006] showed that it is possible to combine disagreement-based SSL and AL to get a promising performance in content-based image retrieval. Wang and Zhou [2008] theoretically verified that such a combination is really able to get an improved sample complexity than pure SSL or pure AL.

2.2. Design Space Exploration

Many investigations reduce the simulation costs for DSE by analyzing program characteristics. Hamerly et al. [2006] simulated only some representative program phases rather than the whole program. From the perspective of program, Joshi et al. [2006] found a reduced representative subset of programs by cluster analysis based on inherent microarchitecture-independent characteristics. Moreover, statistical simulation was employed to construct a synthesized shorter program to emulate the execution characteristics of the original program [Genbrugge and Eeckhout 2009]. Unlike the above approaches, predictive modeling techniques reduce simulated design configurations by learning the relationship between design parameters and processor responses. Following the supervised learning framework, the preceding task was accomplished by linear regression model [Joseph et al. 2006] or Artificial Neural Networks (ANNs) [İpek et al. 2006; Khan et al. 2007; Cho et al. 2007; Dubach et al. 2011], where ANNs are most widely used. Inspired by active learning, İpek et al. [2006] proposed the intelligent sampling technique to enhance the accuracy of the supervised ANN approach. This technique repeatedly updates an ensemble of 10 ANNs trained by 10-fold cross validation over the labeled design configurations, and iteratively labels (simulates) the unlabeled configurations on which the ANNs present largest

disagreements. Benefited from these techniques, architects no longer need to simulate an excessively large number of design configurations. However, since the usefulness of unlabeled design configurations is ignored, the above approaches still suffer from either high simulation costs (for achieving high accuracies) or low prediction accuracy (given limited computational resources).

3. OUR PROPOSAL

3.1. COAL

Let $L = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|L|}, y_{|L|})\}$ be the labeled example set, where $\mathbf{x}_i$ is the i-th design configuration with d interested design parameters, and y_i is the corresponding processor response, for instance, performance metric like Instruction-per-Cycle (IPC). Let U be the unlabeled data set, that is, the set of design configurations without simulation.

The most critical issue in semisupervised regression is how to estimate the labeling confidence such that the most confident unlabeled instance can be selected to label. Following Zhou and Li [2005a]'s approach, we consider that the error of the regressor on the labeled data set should decrease the most if the most confident unlabeled instance is labeled. Formally, for each unlabeled instance $\mathbf{x}_s$, the quality of $\mathbf{x}_s$ can be measured using a criterion as shown in Equation (1):

$$\Delta(\mathbf{x}_s) = \frac{1}{|L|} \sum_{\mathbf{x} \in L} \Big(\big(y - h(\mathbf{x})\big)^2 - \big(y - h'(\mathbf{x})\big)^2 \Big), \tag{1}$$

where h is the original regressor, and h' is the regressor refined with instance $\mathbf{x}_s$ and its label. In other words, the instance with the best *mean squared error* (MSE) reduction on the labeled set will be selected to label.

Another challenge during the design of COAL is that, repeatedly measuring the MSE of regression tree on the entire labeled data set in each iteration is time-consuming. To address this problem, COAL utilizes the local information of constructed regression tree to improve the update efficiency. Specifically, given an unlabeled instance which would be labeled by the original regression tree, it should fit into a linear model that lies in a leaf of the tree. Thus, we approximately measure the labeling confidence of each unlabeled instance by computing only the MSE of its *siblings* locating in the same leaf of the tree, where siblings refer to those labeled examples. In this case, the Promising Labeled Example (PLE) of an unlabeled data set, denoted by $\widetilde{\mathbf{x}}$, is determined by maximizing the local error reduction ($\widetilde{\Delta}(\mathbf{x}_s)$) defined in Equation (2):

$$\widetilde{\Delta}(\mathbf{x}_s) = \frac{1}{|\Omega_s|} \sum_{\mathbf{x} \in \Omega_s} \Big(\big(y - m(\mathbf{x})\big)^2 - \big(y - m'(\mathbf{x})\big)^2 \Big), \tag{2}$$

where Ω_s is sibling set of $\mathbf{x}_s$ in the original tree, m is the original regressor, and m' is the regressor refined by $(\mathbf{x}_s, y_s)$, $y_s = m(\mathbf{x}_s)$. Following the cotraining paradigm, COAL uses two base regression trees, each of which will label the PLE for the other tree during the learning process. Notice that, COAL does not require two views. Similar to COREG [Zhou and Li 2005a] and other single-view disagreement-based SSL approaches, the validity of COAL can be justified by the recent theoretical results [Wang and Zhou 2007, 2010b]. More information on cotraining and other disagreement-based SSL techniques can be found in the recent survey [Zhou and Li 2010].

On the other hand, at each iteration COAL selects and simulates an unlabeled configuration. To be specific, the unlabeled configuration on which two regression trees present the largest disagreement, denoted by x_a, will be labeled by simulation:

$$\mathbf{x}_a = \arg\max_{\mathbf{x} \in U} |m_1(\mathbf{x}) - m_2(\mathbf{x})|, \tag{3}$$

where U is a large set of unlabeled instances.

ALGORITHM 1: Pseudo-code of COAL Algorithm

Data: L : Set of labeled instances.;
U : Pool of all unlabeled instances.;
p : Size of the pool with unlabeled instances.;
t : Number of learning iterations.;
M_1, M_2 : Minimal number of examples in the leaf of regression trees.;
begin
  $L_1 \leftarrow L$; $L_2 \leftarrow L$;
  Create a pool U' with p unlabeled instances;
  Train regression tree m_1 and m_2 with labeled set L by parameters M_1 and M_2, respectively;
  for $l \leftarrow 1$ **to** t **do**
    `/* Co-Training Semi-supervised Learning Phase */`
    for $j \in \{1, 2\}$ **do**
      for *each* $\mathbf{x}_s \in U'$ **do**
        $\Omega_s \leftarrow Sibling(m_j, \mathbf{x}_s)$;
        $y_s \leftarrow m_j(\mathbf{x}_s)$;
        Obtain new model m'_j by adding $(\mathbf{x}_s, y_s)$;
        $\widetilde{\Delta}(\mathbf{x}_s) \leftarrow \frac{1}{|\Omega_s|} \sum_{\mathbf{x} \in \Omega_s} ((y - m_j(\mathbf{x}))^2 - (y - m'_j(\mathbf{x}))^2)$;
      end
      if *there exists a* $\widetilde{\Delta}(\mathbf{x}_s) > 0$ **then**
        $\hat{\mathbf{x}}_j \leftarrow \arg\max_{\mathbf{x}_s \in U'} \widetilde{\Delta}(\mathbf{x}_s)$; $\hat{y}_j \leftarrow m_j(\hat{\mathbf{x}}_j)$;
        $\pi_j \leftarrow \{(\hat{\mathbf{x}}_j, \hat{y}_j)\}$; $U' \leftarrow U' - \{\hat{\mathbf{x}}_j\}$;
      end
      else
        $\pi_j \leftarrow \phi$;
      end
    end
    $L_1 \leftarrow L_1 \cup \pi_2$; $L_2 \leftarrow L_2 \cup \pi_1$;
    Update m_1, m_2 by new set L_1, L_2, respectively;
    Replenish pool U' to size p with randomly selected candidate unlabeled instances from U;
    `/* Active Learning Phase */`
    $\mathbf{x}_a = \arg\max_{\mathbf{x} \in U} |m_1(\mathbf{x}) - m_2(\mathbf{x})|$;
    Simulate $\mathbf{x}_a$ to obtain corresponding responses as y_a;
    $U \leftarrow U - \{\mathbf{x}_a\}$;
    $L_1 \leftarrow L_1 \cup \{(\mathbf{x}_a, y_a)\}$; $L_2 \leftarrow L_2 \cup \{(\mathbf{x}_a, y_a)\}$;
    Update m_1, m_2 by new set L_1, L_2, respectively;
  end
  Output $m^*(\mathbf{x}) \leftarrow \frac{1}{2}(m_1(\mathbf{x}) + m_2(\mathbf{x}))$;
end

Algorithm 1 presents the pseudo-code of COAL algorithm. As the first step, the M5P algorithm is used to construct two diverse regression trees (say, m_1 and m_2) via employing distinct parameters M_1 and M_2 respectively, where M_1 and M_2 determine the minimal number of examples in each leaf of m_1 and m_2, respectively [Wang and Witten 1997]. Let L_1 and L_2 be the current labeled data sets of m_1 and m_2 respectively, and L_1 and L_2 are initialized by the same labeled data set L. In each iteration, COAL uses the PLE determined by m_1 to augment the labeled set L_2, and vice versa. After that, we should replenish the pool U' with unlabeled instances to size p for next iteration. After using the latest labeled sets to update two regression trees, COAL selects the unlabeled configuration with the largest disagreement to simulate, and updates the regression trees using the newly labeled configuration. It is notable that the unlabeled data selected for simulation is chosen from the entire unlabeled data set U instead of

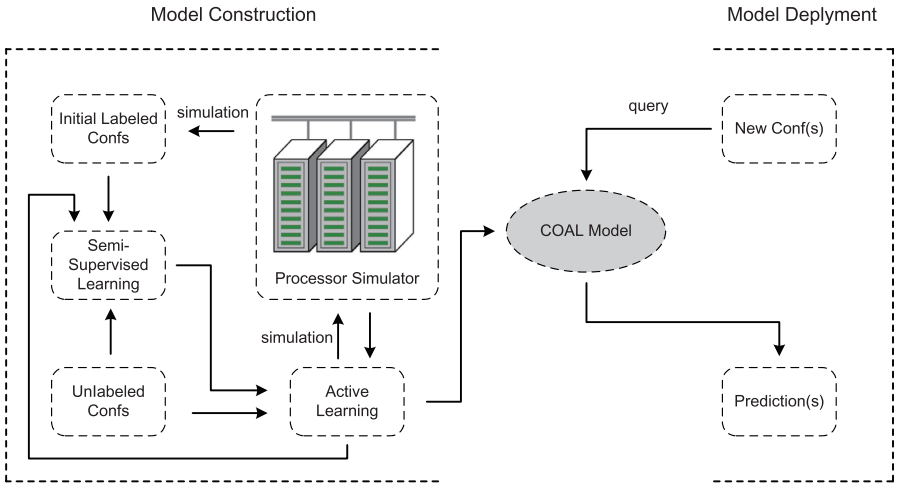

Fig. 2. The framework of COAL for DSE.

the pool U'. Finally, we average the prediction on each updated regression tree as the final prediction.

It is possible to employ more base learners for disagreement-based semisupervised learning as tritraining [Zhou and Li 2005b] and coforest [Li and Zhou 2007] did, and for active learning as query-by-committee [Seung et al. 1992] did. However, the combination of such kind of semisupervised learning and active learning techniques will encounter a big challenge, that is, how to maintain the diversity among the learners in the learning process, as it is well known that the diversity is crucial for disagreement-based approaches to succeed [Wang and Zhou 2007]. Thus, our approach only uses two base learners, and such a process has a sound theoretical support [Wang and Zhou 2008].

3.2. DSE Framework via COAL

Figure 2 illustrates how COAL deploys a regression model for DSE. As the first phase, COAL labels a number of randomly generated design configurations by the simulator (simulation cluster), which constructs the initial labeled data set. At each learning phase of COAL, the semisupervised learning is utilized to select and label two unlabeled design configurations. After that, COAL updates each regression tree by the configuration labeled by the other tree. Using the updated trees, COAL invokes the active learning engine, via which COAL finds and simulates the unlabeled design configurations on which the trees show the largest disagreement. This newly labeled design configuration is then used to further update both trees. Such a procedure is repeated for several times until the predefined iteration is reached. Benefited from the collaboration of cotraining semisupervised learning and active learning, COAL provides us accurate predictions of processer responses at the cost of simulating only a few design configurations.

4. EMPIRICAL STUDY

In this section, we carry out empirical study to evaluate the performance of COAL. The performance of COAL is first compared against that of a state-of-the-art DSE approach. After that, we evaluate the impacts of active learning and semisupervised learning on the performance of COAL, respectively.

Table I. Investigated Microprocessor Design Space

Abbr.	Parameter	Value
WIDTH	Fetch/Issue/Commit Width	2,4,6,8
FUNIT	FPALU/FPMULT Units	2,4,6,8
IUNIT	IALU/IMULT Units	2,4,6,8
L1IC	L1-ICache	1,2,4,8,16,32KB
L1DC	L1-DCache	1,2,4,8,16,32KB
L2UC	L2-UCache	256,512,1024,2048,4096KB
ROB	ROB size	16-256 with a step of 16
LSQ	LSQ size	8-128 with a step of 8
GSHARE	GShare size	1,2,4,8,16,32K
BTB	BTB size	512,1024,2048,4096
Total	**10 parameters**	**70,778,880 Options**

4.1. Evaluation Methodology

4.1.1. Processor Simulator and Design Space. The most common way to evaluate the performance of a processor during DSE is to measure the execution time of many popular benchmarks on cycle-accurate simulators. Here we employ renowned Simplescalar [Austin et al. 2002] as the prototype simulator of a modern superscalar microprocessor, which is run on a cluster with Intel Xeon processors. Each design configuration considered in DSE, as shown in Table I, consists of 10 different design parameters, and the total number of design configurations in the design space exceeds 70 million.

4.1.2. Benchmarks. According to Hennessy and Patterson [2003], the performance of a general-purpose processor should be evaluated over a number of representative benchmark programs. The most famous benchmark suite covering various application fields was created by SPEC (Standard Performance Evaluation Corporation) [Hennessy and Patterson 2003]. SPEC benchmarks consist of real programs with slight modification for portability, and are widely utilized by architects, researchers, and computer vendors for performance evaluation. For example, vendors of desktop computers and servers periodically submit the performance results measured by SPEC benchmarks to *www.spec.org* to provide fair comparisons with other machine products.

SPEC CPU2000 is the fourth generation benchmark suite of SPEC series, which consists of a set of 11 integer benchmarks (CINT2000) and 14 floating-point benchmarks (CFP2000). SPEC CPU2000 aims at providing fair evaluations of general-purpose processors. To validate the effectiveness of COAL on different programs, we consider 14 representative programs with distinct behaviors from SPEC CPU2000 as *applu*, *apsi*, *bzip2*, *crafty*, *eon*, *gzip*, *lucas*, *mesa*, *mgrid*, *parser*, *perlbmk*, *sixtrack*, *swim* and *vortex*. SPEC CPU2000 offers 3 inputs with different sizes for each benchmark, that is, *test*, *train*, and *ref*, and the corresponding size increases in order. Since the size of *ref* input is the largest but the closest to the size of real-world input, it can offer a promising approximation to the behavior of a real application. Here it is adopted as the input in our experiments. Meanwhile, due to the extremely slow simulation speed, we adopt a statistical program sampling technique called SimPoint [Sherwood et al. 2002] to only simulate 100 million representative instructions (which is typically about 1% of all dynamic instructions) selected from the entire execution of each benchmark to obtain the approximate responses (e.g., performance or power)[1].

[1]Our preliminary work [Guo et al. 2011] did not employ such a technique.

4.1.3. Metric. In our study, we use a conventional metric, called the Mean Square Error (MSE), to measure the prediction accuracy of each DSE approach:

$$MSE := \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2, \tag{4}$$

where y_k represents the predicted response (e.g., IPC) of the k-th configuration ($k = 1, \ldots, N$), and $\hat{y}_k$ represents the actual response of the k-th configuration, which is obtained by cycle-by-cycle simulation. To compare the prediction accuracy of COAL with that of the state-of-the-art (ANN), the MSE of COAL on the testing data will be normalized to that of the ANN DSE approach introduced by İpek et al. [2006].

4.1.4. Parameter Setting. In our experiments, the initial training set of COAL consists of 300 randomly generated design configurations which have been labeled by cycle-accurate simulations on the processor simulator. The number of iterations (t in Algorithm 1) in which SSL+AL is carried out is set to 100, which implies that 100 extra unlabeled design configurations, selected in the AL process, will be labeled by cycle-accurate processor simulations; 200 unlabeled design configurations will be labeled by the regression trees in the cotraining SSL process. The entire unlabeled data set (U in Algorithm 1) consists of 100K randomly generated unlabeled design configurations, which are prepared for both SSL and AL processes. For the SSL, the size of pool containing unlabeled design configurations (p in Algorithm 1) is set to 100 as in [Guo et al. 2011]. Moreover, M_1 and M_2 (the minimal numbers of examples in each leaf of two trees, respectively) are set to 4 and 10, respectively, to obtain two diverse M5P regression trees. In order to test the performance of COAL, additional 100 different design configurations are simulated as the testing data. We did not finely tune the parameters because our task is quite different from common machine learning problems owing to the great simulation cost. For example, simulating a design configuration on a SPEC benchmark taking the *ref* input may consume hundreds of hours.

To demonstrate the effectiveness of COAL, we compare it with the ANN DSE approach proposed by İpek et al. [2006], and a supervised M5P regression tree. Following the setting utilized by İpek et al. [2006], the ANN adopts one 16-unit hidden layer, a learning rate of 0.001, and a momentum value of 0.5. The minimal number of examples in each leaf of the M5P tree is set to 4. In our experiments, both ANN and M5P models are constructed by a training set consisting of 400 labeled design configurations. Among the 400 labeled configurations, 300 labeled design configurations are the same to the configurations in the initial training set of COAL, and other 100 configurations are generated randomly and labeled by cycle-accurate processor simulations. In addition, we also provide the performance data of the intelligent-sampling-based ANN DSE approach (ANN-IS for short), a variant of the ANN DSE approach proposed by İpek et al. [2006], as a reference. Following the setting suggested by İpek et al., ANN-IS repeatedly updates an ensemble of 10 ANNs trained by 10-fold cross validation over the labeled design configurations, and iteratively labels (simulates) the unlabeled configurations on which the ANNs present largest coefficients of variance (ratio of the standard deviation to mean). ANN-IS shares the same initial training set with COAL, and all approaches share the same testing data with COAL.

4.2. Results

Figure 3 compares the normalized MSEs of ANN, ANN-IS, the supervised M5P tree and COAL. We can clearly see that COAL outperforms other approaches over all 14 benchmarks. In average, COAL reduces the MSE by 58% of ANN. Most notably, COAL reduces the MSE by 95% of ANN on the benchmark *swim*. Even on the benchmark

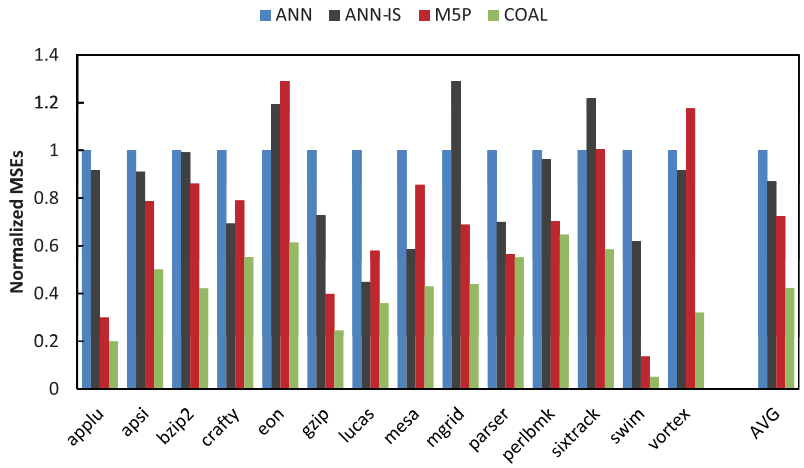

Fig. 3. Normalized MSEs of ANN, ANN-IS, M5P and COAL (normalized to the corresponding MSE of ANN).

with the least MSE reduction, that is, *perlbmk*, the MSE reduction can still achieve 35%. Comparing with ANN-IS, COAL reduces the MSE by 51% in average. Hence, we can conclude that COAL is much more practical than state-of-the-art DSE techniques due to its high accuracy.

Figure 3 also presents the comparison between supervised M5P regression tree and COAL. It can be observed that COAL reduces the MSEs of M5P over all 14 benchmarks. In average, COAL reduces the MSE by 42% of M5P. Hence, it can be concluded that the exploitation of unlabeled configurations in COAL can significantly improve the prediction accuracy.

4.3. Sensitivity to Training Iteration *t*

Training iteration t determines the number of unlabeled design configurations that are used for improving prediction accuracy. The experiment conducted here follows the same parameter setting (except t) as the previous experiments.

Figure 4 shows the relationship between the MSE of COAL and t, where 8 benchmarks, *applu*, *apsi*, *bzip2*, *crafty*, *eon*, *lucas*, *mgrid* and *vortex*, are considered. In general, the MSE (red curves in Figure 4) roughly decreases as the number of iterations increases. For example, on benchmark *applu*, the MSE of the COAL trained with 100 iterations is 63.5% of that of the COAL trained with only 10 iterations. However, when the number of iterations has become large enough, on several benchmarks the MSE may decrease with a relatively slower speed. This is not difficult to understand. On one hand, according to theoretical studies [Wang and Zhou 2007], disagreement-based approaches require a large difference between the learners, while when a large number of iterations has been executed, the learners will become too similar to enable a further performance improvement. On the other hand, after a large number of labeled examples are obtained, the gains from exploiting unlabeled data become smaller.

Figure 4 also illustrates the training cost of COAL given different training iterations. We can clearly see that the training cost of COAL grows almost linearly with respect to t (black curves in Figure 4). Hence, if the computational resources are limited, it would be beneficial if t is determined by trading off the gained prediction accuracy and training costs, which can be achieved by setting more sophisticated stopping criterion for COAL. The related investigation will be conducted in our future work.

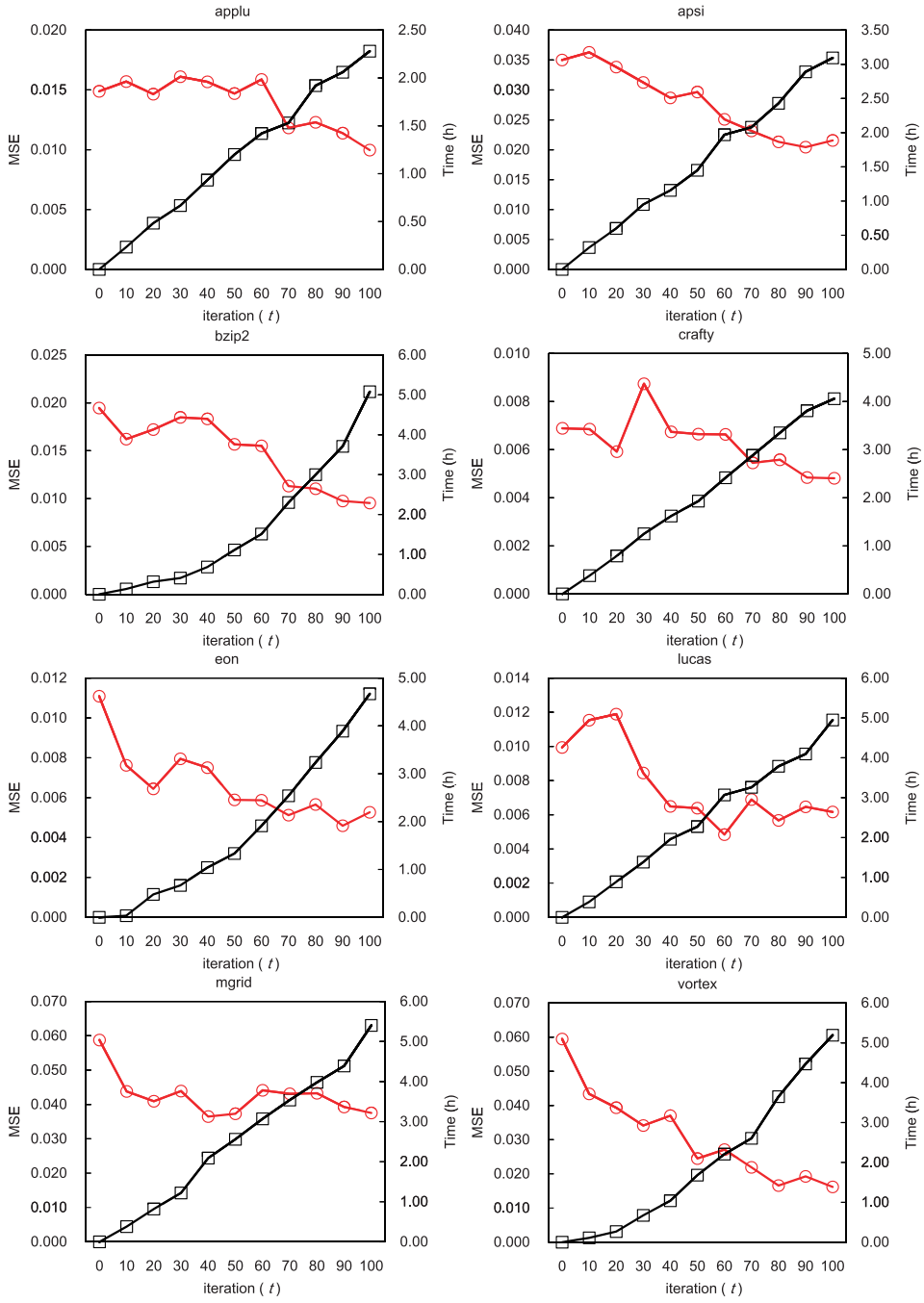

Fig. 4. Prediction accuracy (MSE) and training cost (Time) of COAL with respect to different values of iteration t.

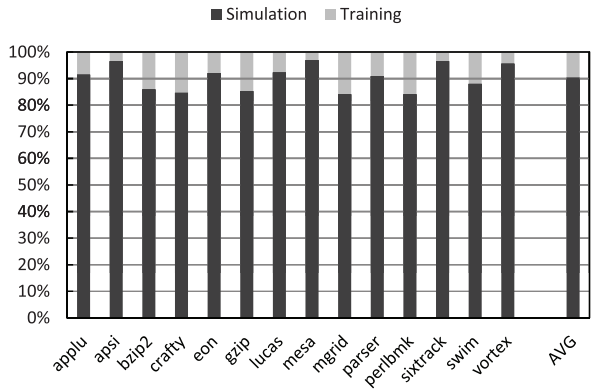

Fig. 5. Simulation cost and training cost of COAL on different benchmarks.

4.4. Computational Cost of COAL

The computational cost of COAL for solving a DSE problem contains two parts, the simulation cost spent on obtaining the actual labels of design configurations, and the training cost spent on selecting unlabeled data and updating the regression trees. Figure 5 compares the simulation cost and training cost consumed by COAL on different benchmarks. We can see that the simulation cost always overwhelmingly dominates the training cost. In average, the simulations consume 90% of the computational cost. On benchmark *mesa*, the simulation cost even exceeds 97% of the entire computational cost. It is worth noting that for each benchmark (benefited from SimPoint [Sherwood et al. 2002]) we only simulate 100 million representative dynamic instructions, while in industry all dynamic instructions (more than 10 billion for most benchmarks) should be simulated. Under this circumstance, the training cost of COAL can be completely neglected in practice.

4.5. Impacts of Active Learning and Semisupervised Learning

So far we have validated that the graceful combination of SSL and AL contributes significantly to the high accuracy of COAL, and found promising solution to the DSE problem. However, it is still not clear whether AL and SSL are both necessary to COAL. In order to gain more insight on this issue, we design additional experiments to study the impact of AL and impact of SSL, respectively. To be specific, we turn off the AL phase in Algorithm 1 and obtain a DSE approach called noAL (as shown in Algorithm 2, noAL is identical to COMT proposed in our preliminary study [Guo et al. 2011]); We turn off the SSL phase in Algorithm 1 and obtain a DSE approach called noSSL (whose flow is shown in Algorithm 3).

In our experiments, noAL follows the same parameter setting as COAL, except that noAL uses the size-400 training set as supervised ANN and M5P while COAL uses an initial training set with 300 labeled configurations; noSSL follows the same parameter setting as COAL (but does not require the parameter p anymore). A summary of performance of COAL, noAL, noSSL, supervised ANN, ANN-IS and M5P is presented in Table II. According to this summary, COAL performs the best among the 5 DSE approaches on 8 out of 14 benchmarks, noAL and noSSL perform the best on 3 benchmarks, respectively. In the average sense, COAL significantly outperforms all other approaches, that is, it reduces the average MSE by 58%, 42%, 51%, 21% and 28% compared with ANN, M5P, ANN-IS, noAL and noSSL, respectively. Moreover, it is worth noting that approaches leveraging unlabeled configurations (COAL, noAL and noSSL) significantly outperform approaches do not exploit any unlabeled configuration

ALGORITHM 2: Pseudo-code of noAL Algorithm

begin
  $L_1 \leftarrow L$; $L_2 \leftarrow L$;
  Create a pool U' with p unlabeled instances;
  Train regression tree m_1 and m_2 with labeled set L by parameters M_1 and M_2, respectively;
  for $l \leftarrow 1$ **to** t **do**
    /* Co-Training Semi-supervised Learning Phase */
    for $j \in \{1, 2\}$ **do**
      for *each* $\mathbf{x}_s \in U'$ **do**
        $\Omega_s \leftarrow Sibling(m_j, \mathbf{x}_s)$;
        $y_s \leftarrow m_j(\mathbf{x}_s)$;
        Obtain new model m'_j by adding $(\mathbf{x}_s, y_s)$;
        $\widetilde{\Delta}(\mathbf{x}_s) \leftarrow \frac{1}{|\Omega_s|} \sum_{\mathbf{x} \in \Omega_s} ((y - m_j(\mathbf{x}))^2 - (y - m'_j(\mathbf{x}))^2)$;
      end
      if *there exists a* $\widetilde{\Delta}(\mathbf{x}_s) > 0$ **then**
        $\hat{\mathbf{x}}_j \leftarrow \arg\max_{\mathbf{x}_s \in U'} \widetilde{\Delta}(\mathbf{x}_s)$; $\hat{y}_j \leftarrow m_j(\hat{\mathbf{x}}_j)$;
        $\pi_j \leftarrow \{(\hat{\mathbf{x}}_j, \hat{y}_j)\}$; $U' \leftarrow U' - \{\hat{\mathbf{x}}_j\}$;
      end
      else
        $\pi_j \leftarrow \phi$;
      end
    end
    $L_1 \leftarrow L_1 \cup \pi_2$; $L_2 \leftarrow L_2 \cup \pi_1$;
    Update m_1, m_2 by new set L_1, L_2, respectively;
    Replenish pool U' to size p with randomly selected candidate unlabeled instances from U;
  end
  Output $m^*(\mathbf{x}) \leftarrow \frac{1}{2}(m_1(\mathbf{x}) + m_2(\mathbf{x}))$;
end

ALGORITHM 3: Pseudo-code of noSSL Algorithm

begin
  $L_1 \leftarrow L$; $L_2 \leftarrow L$;
  Train regression tree m_1 and m_2 with labeled set L by parameters M_1 and M_2, respectively;
  for $l \leftarrow 1$ **to** t **do**
    /* Active Learning Phase */
    $\mathbf{x}_a = \arg\max_{\mathbf{x} \in U} |m_1(\mathbf{x}) - m_2(\mathbf{x})|$;
    Simulate $\mathbf{x}_a$ to obtain corresponding responses as y_a;
    $U \leftarrow U - \{\mathbf{x}_a\}$;
    $L_1 \leftarrow L_1 \cup \{(\mathbf{x}_a, y_a)\}$; $L_2 \leftarrow L_2 \cup \{(\mathbf{x}_a, y_a)\}$;
    Update m_1, m_2 by new set L_1, L_2, respectively;
  end
  Output $m^*(\mathbf{x}) \leftarrow \frac{1}{2}(m_1(\mathbf{x}) + m_2(\mathbf{x}))$;
end

(ANN and M5P). The above observations again validate the usefulness of unlabeled configurations, and strongly support to exploit unlabeled design configurations in DSE.

Compared with the noAL approach, COAL outperforms it on 11 out of 14 benchmarks, for instance, on benchmarks *swim* and *vortex* COAL can reduce the MSE by about 58% of that of noAL; Compared with noSSL approach, COAL also outperforms it on 11 out of 14 benchmarks, for instance, on the benchmark *vortex* COAL can reduce

Table II.
Normalized MSEs of COAL, noAL, noSSL, supervised ANN, ANN-IS and M5P over 14 benchmarks. All MSEs are normalized to the corresponding MSE of ANN.

Benchmarks	ANN	ANN-IS	M5P	noAL	noSSL	COAL
applu	1.000	0.917	0.299	0.260	0.273	**0.200**
apsi	1.000	0.910	0.787	0.789	0.785	**0.501**
bzip2	1.000	0.991	0.860	0.695	0.426	**0.421**
crafty	1.000	0.693	0.790	0.825	0.804	**0.551**
eon	1.000	1.192	1.290	**0.582**	0.642	0.613
gzip	1.000	0.728	0.398	0.265	**0.217**	0.245
lucas	1.000	0.449	0.580	0.484	**0.342**	0.359
mesa	1.000	0.586	0.855	0.497	0.745	**0.429**
mgrid	1.000	1.290	0.688	0.555	0.561	**0.440**
parser	1.000	0.699	0.565	**0.456**	0.572	0.551
perlbmk	1.000	0.963	0.702	**0.603**	0.937	0.646
sixtrack	1.000	1.217	1.005	0.613	**0.472**	0.584
swim	1.000	0.620	0.137	0.123	0.067	**0.051**
vortex	1.000	0.916	1.175	0.765	1.342	**0.320**
AVG	1.000	0.869	0.724	0.537	0.585	**0.422**
VAR	0	0.064	0.102	0.044	0.109	**0.029**

the MSE by about 76% of that of noSSL. One reason that COAL does not always improve the performance might owe to the fact that there are some similar architectural configurations with different functional behaviors. Such issues can be address by adding some additional features that are able to distinguish these architectural configurations. This will be one of our future work. Moreover, it is well known that sometimes semisupervised learning might degenerate performance; however, recently there are some proposals on safe semisupervised learning [Li and Zhou 2011], and it is possible to introduce these techniques for further improvement.

Fortunately, adopting both AL and SSL achieves the best performance at most time. More specifically, when COAL is beaten by noAL on benchmarks *eon*, *parser*, and *perlbmk*, it can still achieve significantly better performance than that of noSSL on these benchmarks. When COAL is beaten by noSSL on *gzip*, *lucas*, and *sixtrack*, it still significantly outperforms noAL on these benchmarks. The above observation reveals that the prediction accuracy may vary from one benchmark to another when exploiting unlabeled configurations by SSL or AL, but such performance variance can be significantly reduced if we combine SSL and AL together (cf. Table II). In other words, COAL exhibits more stable performance in comparison with noAL and noSSL, which will be favored by computer architects and engineers.

4.6. How COAL Helps to Find Better Configuration

Ideally, to validate the effectiveness of COAL, we have to simulate the entire design space consisting of 70 million design configurations, then compare the configuration deduced by COAL and the actual optimal configuration. However, this is infeasible due to intractably large simulation costs. A compromise is made by comparing the promising configuration deduced by COAL with the configuration deduced by the state-of-the-art DSE approach (ANN). To be specific, we simulate the two architectural configurations on two illustrative benchmarks, and compare the corresponding processor responses (performance and power) directly.

As an example, we employ the benchmarks *bzip2* and *crafty* in the experiments. Suppose the design specification is that "maximizing the performance with the constraint that the power consumption must be less than 100 watt", we can attain the promising

Table III.
Comparison of promising design configurations attained by the ANN-based approach and COAL for benchmark *bzip2* and *crafty*.

	bzip2		***crafty***	
Parameter	ANN-conf	COAL-conf	ANN-conf	COAL-conf
WIDTH	8	8	4	8
FUNIT	2	2	2	2
IUNIT	6	8	8	4
L1IC	1KB	1KB	32KB	32KB
L1DC	1KB	1KB	32KB	32KB
L2UC	256KB	256KB	256KB	256KB
ROB	80	48	128	48
LSQ	16	120	128	88
GSHARE	1024	1024	32768	1024
BTB	512	512	4096	512

Table IV.
Comparison of performance and power responses with respect to ANN-conf and COAL-conf for benchmark *bzip2* and *crafty*.

Benchmark	Configuration	Responses	Predicted	Actual	Error
bzip2	ANN-conf	Performance (IPC)	2.79	2.53	10.28%
		Power (Watt)	99.84	98.09	1.78%
	COAL-conf	Performance (IPC)	2.77	2.63	5.32%
		Power (Watt)	99.61	98.33	1.3%
crafty	ANN-conf	Performance (IPC)	2.60	2.14	21.50%
		Power (Watt)	98.57	96.38	2.27%
	COAL-conf	Performance (IPC)	2.28	2.21	3.2%
		Power (Watt)	99.72	99.75	0%

design configurations as shown in Table III, where the promising configurations found by its opponent (state-of-the-art DSE approach using ANN) are also presented. We can clearly see that ANN and COAL suggest different design configurations for these two benchmarks given the above design specification.

Table IV further presents the corresponding performance and power of these two configurations (namely, "ANN-conf" and "COAL-conf"). On both benchmarks, the actual performance (obtained by cycle-accurate simulations) of COAL-confs is better than that of ANN-confs given the 100 watt power constraints. It is notable that on benchmark *crafty*, the predicted and actual performance/power of COAL-conf are very close while those of ANN-conf are quite different. Quantitatively, the relative error of the predicted IPC of ANN-conf over the actual IPC is 21.5%. Hence, we can conclude that COAL is more effective than the state-of-the-art ANN-based approach.

5. CONCLUSION AND FUTURE WORK

In contrast to traditional DSE approaches that only consider labeled design configurations, this article proposes to leverage unlabeled design configurations to help improve the performance, which extends our preliminary research [Guo et al. 2011]. By exploiting unlabeled design configurations via both semisupervised learning and active learning, our proposed COAL approach significantly improves prediction accuracies and reduces excessive simulation costs of DSE. Experimental results on the design space of a modern superscalar microprocessor show that the MSE reduction of COAL over the state-of-the-art DSE approach (ANN) ranges from 35% to 95% over 14 evaluated benchmarks, and the average MSE reduction of COAL is 58%. COAL enables

effective and efficient deduction of the optimal architecture for every given benchmark program, which greatly benefits the DSE of microprocessor design. In practice, COAL has been utilized in the design of the next-generation Godson processor core.

Currently in our proposed method, the semisupervised learning and active learning components are executed with almost equal importance. It is possible to introduce a tradeoff parameter to enable them to have different amount of contributions, and this will be one of our future work. Besides, it is interesting to study how much minimum the labeled data would be required. Zhou et al. [2007] disclosed that when there are two sufficient views, semisupervised learning with even a single labeled example is possible. However, for our current proposed method, and particularly for DSE application, this also remains an interesting future issue.

It is worth noting that the current version of COAL is proposed for pre-silicon design phase of microprocessor, that is, deciding an appropriate design configuration before manufacturing the chip. In the future, we will continue to develop new versions of COAL to conduct power-efficient post-silicon microprocessor reconfiguration which can adapt to various programs or even some given program features. This task would be a crucial step towards the development of an *elastic processor* (by which we call a processor whose architecture parameters can be dynamically reconfigured to suit different programs) and a *computer tribe* (by which we call a series of downward compatible elastic processors). Unlike the Field Programmable Gate Array (FPGA) whose reconfiguration may have to modify millions of controlling parameters, an elastic processor only employs a moderate number of reconfigurable parameters, which alleviates the problem of dimension explosion when building performance/power regression models for guiding architecture reconfiguration.

REFERENCES

Austin, T., Larson, E., and Ernst, D. 2002. Simplescalar: An infrastructure for computer system modeling. *Computer 35,* 2, 59–67.

Balcan, M.-F., Beygelzimer, A., and Langford, J. 2006. Agnostic active learning. In *Proceedings of 23th International Conference on Machine Learning*. 65–72.

Blum, A. and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Learning Theory*. 92–100.

Chapelle, O. and Zien, A. 2005. Semi-supervised learning by low density separation. In *Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*. 57–64.

Cho, C.-B., Zhang, W., and Li, T. 2007. Informed microarchitecture design space exploration using workload dynamics. In *Proceedings of the 40th International Symposium on Microarchitecture*. 274–285.

Dagan, I. and Engelson, S. P. 1995. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the 12th International Conference on Machine Learning*. 150–157.

Dasgupta, S. and Hsu, D. 2008. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning.* 208–215.

Dasgupta, S., Hsu, D., and Monteleoni, C. 2007. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems 21*, 353–360.

Donmez, P., Carbonell, J. G., and Bennett, P. N. 2007. Dual strategy active learning. In *Proceedings of the 18th European Conference on Machine Learning*. 116–127.

Dubach, C., Jones, T., and O'Boyle, M. 2011. An empirical architecture-centric approach to microarchitectural design space exploration. *IEEE Trans. Comput. 60,* 10, 1445–1458.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. 1997. Selective sampling using the query by committee algorithm. *Machine Learn. 28,* 2–3, 133–168.

Fujino, A., Ueda, N., and Saito, K. 2005. A hybrid generative/discriminative approach to semisupervised classifier design. In *Proceedings of the 20th National Conference on Artificial Intelligence*. 764–769.

Genbrugge, D. and Eeckhout, L. 2009. Chip multiprocessor design space exploration through statistical simulation. *IEEE Trans. Comput. 58,* 12, 1668–1681.

GUO, Q., CHEN, T., CHEN, Y., ZHOU, Z.-H., HU, W., AND XU, Z. 2011. Effective and efficient microprocessor design space exploration using unlabeled design configurations. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. 1671–1677.

HAMERLY, G., PERELMAN, E., LAU, J., CALDER, B., AND SHERWOOD, T. 2006. Using machine learning to guide architecture simulation. *J. Machine Learn. Res. 7*, 343–378.

HANNEKE, S. 2007. A bound on the label complexity of agnostic active learning. In *Proceedings of the 24th International Conference on Machine Learning*. 353–360.

HENNESSY, J. L. AND PATTERSON, D. A. 2003. *Computer Architecture: A Quantitative Approach* 3rd Ed. Morgan Kaufmann, San Francisco, CA.

HU, W., WANG, J., GAO, X., CHEN, Y., LIU, Q., AND LI, G. 2009. Godson-3: A scalable multicore RISC processor with X86 emulation. *IEEE Micro 29,* 2, 17–29.

HUANG, S.-J., JIN, R., AND ZHOU, Z.-H. 2010. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems 23*, 892–900.

ÏPEK, E., MCKEE, S. A., CARUANA, R., SUPINSKI, B. R., AND SCHULZ, M. 2006. Efficiently exploring architectural design spaces via predictive modeling. In *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*. 195–206.

JOACHIMS, T. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of International Conference on Machine Learning*. 200–209.

JOSEPH, P., KAPIL, V., AND THAZHUTHAVEETIL, M. 2006. Construction and use of linear regression models for processor performance analysis. In *Proceedings of the 12th International Symposium on High Performance Computer Architecture*. 99–108.

JOSHI, A., PHANSALKAR, A., EECKHOUT, L., AND JOHN, L. K. 2006. Measuring benchmark similarity using inherent program characteristics. *IEEE Trans. Comput. 55,* 6, 769–782.

KHAN, S., XEKALAKIS, P., CAVAZOS, J., AND CINTRA, M. 2007. Using predictivemodeling for cross-program design space exploration in multicore systems. In *Proceedings of the 16th International Conference on Parallel Architecture and Compilation Techniques*. 327–338.

LEE, B. C., COLLINS, J., WANG, H., AND BROOKS, D. 2008. Cpr: Composable performance regression for scalable multiprocessormodels. In *Proceedings of the 41st International Symposium on Microarchitecture.* 270–281.

LEWIS, D. D. AND CATLETT, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*. 148–156.

LI, M. AND ZHOU, Z.-H. 2007. Improve computer-aided diagnosis with machine learning techniques using undiagnosed samples. *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans 37,* 6, 1088–1098.

LI, M. AND ZHOU, Z.-H. 2011. Towards making unlabeled data never hurt. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*. 1081–1088.

MILLER, D. J. AND UYAR, H. S. 1997. A mixture of experts classifier with learning based on both labelled and unlabelled data. In *Advances in Neural Information Processing Systems 9*, 571–577.

MUSLEA, I., MINTON, S., AND KNOBLOCK, C. A. 2000. Active learning using pre-clustering. In *Proceedings of the 17th National Conference on Artificial Intelligence*. 621–626.

NGUYEN, H. T. AND SMEULDERS, A. W. M. 2004. Active learning using pre-clustering. In *Proceedings of the 21th International Conference on Machine Learning*. 623–630.

NIGAM, K., MCCALLUM, A. K., THRUN, S., AND MITCHELL, T. 2000. Text classification from labeled and unlabeled documents using em. *Machine Learn. 39,* 2–3, 103–134.

SEUNG, H. S., OPPER, M., AND SOMPOLINSKY, H. 1992. Query by committee. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory*. 287–294.

SHERWOOD, T., PERELMAN, E., HAMERLY, G., AND CALDER, B. 2002. Automatically characterizing large scale program behavior. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*. 45–57.

WANG, W. AND ZHOU, Z.-H. 2007. Analyzing co-training style algorithms. In *Proceedings of the 18th European Conference on Machine Learning.* 454–465.

WANG, W. AND ZHOU, Z.-H. 2008. On multi-view active learning and the combination with semi-supervised learning. In *Proceedings of the 25th International Conference on Machine Learning*. 1152–1159.

WANG, W. AND ZHOU, Z.-H. 2010a. Multi-view active learning in the non-realizable case. In *Advances in Neural Information Processing Systems 24*, 2388–2396.

WANG, W. AND ZHOU, Z.-H. 2010b. A new analysis of co-training. In *Proceedings of the 27th International Conference on Machine Learning*. 1135–1142.

WANG, Y. AND WITTEN, I. 1997. Induction of model trees for predicting continuous classes. In *Proceedings of the 9th European Conference on Machine Learning*. 128–137.

XU, L. AND SCHUURMANS, D. 2005. Unsupervised and semi-supervised multi-class support vector machines. In *Proceedings of the 20th National Conference on Artificial Intelligence*. 904–910.

ZHOU, D., BOUSQUET, O., LAL, T. N., WESTON, J., AND SCHÖLKOPF, B. 2004. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, 321–328.

ZHOU, Z.-H., CHEN, K.-J., AND DAI, H.-B. 2006. Enhancing relevance feedback in image retrieval using unlabeled data. *ACM Trans Inf. Syst. 24,* 2, 219–244.

ZHOU, Z.-H. AND LI, M. 2005a. Semi-supervised regression with co-training. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. 908–913.

ZHOU, Z.-H. AND LI, M. 2005b. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Trans. Knowl. Data Engin. 17,* 11, 1529–1541.

ZHOU, Z.-H. AND LI, M. 2010. Semi-supervised learning by disagreement. *Knowledge Inf. Syst. 24,* 3, 415–439.

ZHOU, Z.-H., ZHAN, D.-C., AND YANG, Q. 2007. Semi-supervised learning with very few labeled training examples. In *Proceedings of the 22nd National Conference on Artificial Intelligence*. 675–680.

ZHU, X., GHAHRAMANI, Z., AND LAFFERTY, J. D. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of 20th International Conference on Machine Learning*. 912–919.

Received January 2012; revised March 2012; accepted May 2012