Accelerating Level-1 Cache Design Space Exploration using Machine Learning Approach for Miss Rate and Power Prediction

A Synopsis

For the award of

Doctor of Philosophy

In

Electronics and Communication Engineering

Submitted by

Hetal Vinubhai Dave

[Enrollment No.: 209999915011]

under the supervision of

Dr. Nirali A. Kotak,

L. D. College of Engineering, Ahmedabad

submitted to

Gujarat Technological University

Ahmedabad, Gujarat, India

March 2026

Table of Contents

1. Title of the Thesis and Abstract

Title of the thesis: Accelerating Level-1 Cache Design Space Exploration using Machine Learning Approach for Miss Rate and Power Prediction

Abstract: This work presents a rapid method for predicting miss rates and power consumption of level-1 cache memory for an application, aiming to accelerate design space exploration during the early design phase by leveraging machine learning. We proposed two predictive models that utilize cache configurations and the application's characteristics. We considered 28 cache configurations defined by size, line size, and associativity, along with 20 different applications. The proposed models achieve high accuracy for L1D (correlation coefficients of 0.989 and 0.994 for miss rate and power, respectively) and L1I (correlation coefficients of 0.987 and 0.992 for miss rate and power, respectively), with lower error (MAE and RMSE) for both miss rate and power consumption prediction. They also exhibit good generalization, rapidly predicting the miss rates and power consumption of new (test) applications, achieving 12x to 148x speedups over the traditional architectural simulator. It significantly reduces the designer's effort in exploring the cache design space by minimizing the need for time-consuming simulations. Furthermore, by applying Pareto-optimal selection, we identified the miss-rate-power-optimal level-1 cache configurations for L1D and L1I, demonstrating the method's ability to accelerate cache design.

2. Introduction to the Topic

Modern devices require high performance with low-power consumption, low cost, and a shorter time-to-market. Modern processors deliver excellent performance, and memory systems require a similar level of performance. The speed gap arises because the processor and memory system, such as DRAM, operate at different speeds. This speed gap will widen as CMOS technology approaches its fundamental limits. That means the processor has to wait most of the time for data from the main memory. Consequently, the limitation of the memory speed significantly reduces the processor's performance. The memory hierarchy, particularly the cache, is vital to processor performance and system cost, as it is a fast, small, static random-access memory (SRAM). The cache is essential in bridging this gap by storing recently referenced data or instructions. It is designed to provide both reuse and locality, both on-chip and off-chip, for the processor. The cache supports multiple levels of a memory

hierarchy to balance size, access time, and cost, which are categorized as level-1 (L1), level 2 (L2), level 3 (L3), and, occasionally, level 4 (L4). To improve performance, the level-1 (L1) (on-chip) cache can be unified or separated. The separated L1 is divided into two parts: a data cache (L1D) for storing data and an instruction cache (L1I) for storing instructions. The L1 cache is particularly critical for processor performance. It drastically enhances memory access time from 60 ns in dynamic random access memory (DRAM) to 10 ns in L1 cache. However, L1 cache occupies a significant portion of the processor dies and consumes a considerable amount of the processor's total power, i.e., 20% to 40% [1][2][3]. Its parameters largely influence the processor's performance. Hence, cache is an excellent candidate for consideration [3] as its design significantly influences performance and energy efficiency.

A large cache leads to greater silicon area, power consumption, and system cost, while a small one results in lower performance [4], necessitating careful cache selection. Hence, it is essential to balance the trade-off among cache design parameters, the performance improvement offered by the cache, and system cost. An optimal cache design comprises size, line size, associativity, replacement policy, and cache level. The possible combinations of these design parameters are called design points, and all design points together form the design space. Each design point in such a design space corresponds to a unique cache configuration. A typical design space can have hundreds of configurations. In practice, the number of design points can increase exponentially as design complexity rises. A system designer must analyze various cache configurations in terms of performance before designing a cache. Therefore, finding a suitable cache configuration within the vast design space that meets the system's application requirements is difficult. Accordingly, rapid and competent design space exploration (DSE) is required [5]. The DSE is a tedious task employed during the early design cycle [6]. Moreover, different applications need different optimal cache configurations. As miss rate and power consumption are essential to the cache design of an application, an optimal cache configuration must be identified and analyzed using an architectural simulator during cache DSE. These simulators generate an accurate result. Thus, it eliminates the need for actual hardware. However, simulating the given application with different cache configurations is very time-consuming, as it requires simulating each configuration individually, slowing the analysis of large configurations. i.e., during our experiment, a bfs application took nine hours to simulate 28 cache configurations. In today's world, machine learning (ML) is utilized in nearly every field. Over the years, engineers and researchers have begun adopting ML methods to solve DSE and computer-

architecture-related problems [7][8][9][10]. The authors [7] broadly surveyed the adoption of ML in computer architecture through a new understanding of the ML-systems relationship.

ML predictive models can learn relationships in the training dataset. It is mentioned in [6] that one can train an ML model by preparing a training dataset using its microarchitecture parameters and the application's profiling data to forecast the processor's performance. We conducted our work based on this opinion and prepared the training dataset using microarchitecture parameters, such as cache size, line size, and associativity. Moreover, we included application-specific characteristics in the training dataset to predict L1 cache performance and power consumption.

This work aims to accelerate key metrics of the level-1 cache — miss rate and power consumption prediction — using an ML method, as it is a promising, rapid, and accurate solution for accelerating DSE in computer architecture today. The proposed models provide the miss rate and power consumption of all 28 cache configurations of an application. Some configurations yield the optimal miss rate in this cache design space, while others produce the optimal power. On the other hand, a miss-rate-power-optimal cache configuration is often required. We applied Pareto-optimal analysis to determine the cache configuration (a specific combination of cache design parameters [size, line size, associativity]) that achieves a balanced trade-off between miss rate and power consumption for an application based on predicted values, thereby assisting in rapid, informed cache design decisions.

3. Literature Review and Research Gap

A cache optimization and DSE problem has been solved primarily using conventional methods, including analytical models, evolutionary techniques, and simulation-based approaches. Recently, many works have adopted ML as an intelligent method to solve this problem.

An analytical approach is used to evaluate the performance of different cache configurations and to carry out the cache DSE by developing a mathematical model. This method is known for its rapid evaluation of a large number of design points; however, it is less accurate compared to simulation. In earlier work [11][12], an analytical and heuristic-based cache modelling was presented. The work [2] proposed an analytical method using a control flow graph (CFG) for L1I cache DSE. They experimented with SimpleScalar and CACTI using the MiBench and SPEC2000 benchmark suites. Focused parameters include L1I size, line

size, and associativity. They provided the foundational work for cache DSE, with high accuracy and fast estimation compared to simulation using an analytical approach. However, their work is only for L1I and requires a complex analytical model. A meta-heuristic based on grammatical evolution is proposed by [1] to find the optimal cache configuration. Their proposed approach confirms the appropriateness of the evolutionary method for optimizing cache performance. Although their approach is multi-objective cache optimization, they rely on simulation-based evaluation. This limitation underscores the need for data-driven, ML-based models.

In the simulation-based approach, performance evaluation of different cache configurations under various benchmarks is conducted using architectural simulators to identify optimal configurations. Simulators can provide detailed and accurate analysis, but they are very slow. The simulation-based technique for evaluating cache configurations is described in [4][13][14]. The aim of the work [14] is to employ a simulation-based approach to evaluate and compare two CPU cache hierarchies (Intel and AMD). The author focused on architectural performance evaluation rather than cache optimization. They evaluated miss behaviour and execution characteristics by varying the core count using the CAPBench benchmark suite in gem5. The work demonstrates that one can analyze cache behaviour without real hardware by employing a simulation-based approach and using tools such as gem5 to evaluate processor and cache performance.

Many recent works have successfully employed ML in cache optimization and computer architecture-related problems [7][8][9][10]. Among various ML models, the supervised learning approach has gained broad adoption in cache DSE, where models are trained on a labeled dataset generated by simulation or profiling. The supervised learning models are categorized into regression problems, which predict continuous values such as cache miss rates or power consumption, and classification problems, which select an optimal cache configuration from a discrete set. As ML-based predictive models provide acceptable accuracy and reduce exploration time, they offer an alternative solution to repeated simulations for evaluating memory performance and DSE.

The work, NAPEL [15], proposed by Singh et al. for quick instruction per cycle (IPC) and power consumption prediction in near-memory computing (NMC) architectures, using an ML-based regression framework. They trained their RF model using hardware-independent characteristics of 12 applications, gathered through LLVM-based analysis, along with microarchitecture parameters. To reduce simulation time, they use representative sampling from a large sample of application input space by applying the design of experiments (DoE)

method. To ensure robust evaluations and generalization across applications, the authors use a leave-one-out approach: they train the model on all applications, leave one out for testing as a new application, and then iteratively consider all applications. The trained NAPEL can predict 220 times faster than the NMC simulator without additional simulation and accurately predict the IPC and power consumption of new applications, demonstrating the capacity of ensemble learning, such as RF, to capture the nonlinear relationships in the training data. These represent an essential step towards the ML-based architectural performance evaluation and DSE. We adopt the same principle for our leave-one-out application validation approach, as used in this work. The artificial neural network (ANN) model is trained in [16] using 15 applications to predict CPU IPC. This work is among the earliest to employ an ML-based regression approach to effectively explore an ample architectural and memory system design space through time-intensive simulation. The authors model the simulator as a nonlinear function of architectural parameters, rather than simulating each design point. They trained their model on a subset of selected design points and used it to predict the IPC for the remaining design points in the design space. For robust training and accurate prediction, they use cross-validation. Essentially, the authors demonstrate that ML modeling can reduce prediction time compared to simulation techniques, such as SimPoint, without compromising paramount accuracy.

Sen and Imam [17] proposed ML-based prediction for a hybrid main-memory system, which is combined by DRAM and non-volatile memory parameters and trained RF, support vector machine (SVM), MLP, and gradient-boosting to predict latency, power consumption, bandwidth, and memory read/write, by preparing a training dataset using NVmain and gem5, and considered two applications. The authors employed gem5 to gather memory traces and NVmain to simulate the hybrid memory. They trained their models using learning-curve hyperparameter tuning to avoid overfitting. In their framework, the best-fitting models for prediction are an MLP for bandwidth prediction, an RF for latency prediction, GB and RF for memory read/write prediction, and an SVM for power consumption prediction. The authors reported their experimental results, showing that the proposed ML models can predict memory responses with reasonable accuracy and faster than simulations. Although this work aims for hybrid memory rather than cache memory, it sets an example by employing a regression-based ML method as a substitute for time-consuming simulation in architectural performance evaluation and DSE.

Similarly, in work [18], a regression-based ML framework for DSE in DRAM and non-volatile memory systems was proposed, aiming to achieve faster evaluation than simulation.

The authors trained support vector machine (SVM), random forest (RF), and gradient boosting (GB) models using training data generated by gem5, integrated with NVMain. The trained models characterized the relationships between memory design parameters and performance metrics, including bandwidth, average and total latency (predicted using SVM), memory reads, and writes (predicted using RF), and power consumption (predicted using GB). The authors reported their experimental results, showing that the proposed regression-based ML models can predict memory responses more accurately and faster than simulations. However, the approach targeted mainly DRAM and used only two applications. Conversely, our approach targeted the cache and used 20 diverse domain applications from 5 standard benchmark suites. The authors [17] and [18] successfully used multiple ML models to predict different target values, suggesting that training the model according to target values is practical and acceptable for prediction.

Some works have addressed the cache DSE problem by using a classification ML model and reporting the trained model's evaluation metrics, such as F-score, precision, and recall. The work [19] proposed a classification ML model for optimal cache line size selection that employs a multilayer perceptron (MLP) neural network. They collected memory access traces using Intel's Pin tool and obtained cache performance metrics using the SMPCache trace-driven simulator. The proposed model was trained using the Weka 3.8 toolkit and evaluated on data mining applications from the NUmineBench benchmark. The reported prediction accuracy for cache line size selection is 95%. Although their approach is accurate, it only considers one cache parameter, such as cache line size, based on the miss rate, without considering the power consumption value of the cache, and relies on memory traces, which require high storage. The authors determined the L1I and L1D cache line sizes in [20] by training an ANN and addressed the cache optimization problem. They reported the classification accuracy for cache line selection is up to 86% for L1I and 91% for L1D. The authors [21] determined the optimal number of cache levels and sizes for commercial applications using a data mining-based approach. They prepared the training dataset using the SimICS x86 simulator, then trained a classification-based decision tree (DT) with the See5 tool. Using the greedy algorithm, they determined the most suitable cache size for each level. However, its applicability to a broader range of applications is unclear, as experiments have been conducted on only two representative commercial data mining applications, C4.5 and Apriori. The authors [22] proposed an RF model based on the classification method to find the L1D cache configuration (cache size, line size, and associativity). They used gem5 and CACTI to generate a training dataset. Their classification-based approach can

effectively identify a suitable cache configuration that meets application requirements in a shorter timeframe than a simulation-based approach. The authors [23] proposed a classification-based random subspace model to determine the optimal cache level for PARSEC benchmark applications running on a multicore architecture. The authors considered cache size, associativity, line size, and cache type as design parameters, and performance metrics such as execution time (generated using gem5) and power consumption (determined using CACTI) to determine the most suitable cache level. They used 10-fold cross-validation to train the model and reported precision, recall, and F-measure as evaluation metrics. Their experimental results show that a classification-based method achieves high accuracy in determining the cache level. However, the approach targets cache optimization as a discrete classification problem, aiming to predict the cache level rather than continuously predicting cache performance or power consumption, which limits its applicability to fine-grained cache DSE.

The author [24] proposed an ANN model to predict LRU conflict misses in L1D and articulated the cache performance prediction problem as a regression problem. The features and labels for the training dataset were prepared using a stack distance histogram of 15 applications generated via gem5 simulations and 9 cache configurations. While they successfully employed a regression ML model that accurately forecasts an L1D miss rate closer to simulation and is effective for modeling cache performance, their work primarily focuses on predicting miss rates. They do not consider power consumption or cache DSE. The quantitative comparison, based on RMSE values, between our model and this work [24] is shown in Figure 4 (Section 8, "Key results and observations of the study").

The work in [25][26][27][28][29] employed a Pareto-optimal solution for micro-architectural optimization.

Table 1 lists all the approaches mentioned in the literature review for evaluating the performance and DSE, facilitating quick comparison.

Table 1. Different approaches for performance evaluation and DSE

Reference	Approach type	Target system	Methodology/ Algorithm	Simulator/ Tools	Benchmark suit	No.of applications used for training dataset	Target metrics
Liang & Mitra (2013) [2]	Analytical	L1I	CFG	SimpleScalar, CACTI	MiBench, SPEC2000	NA	Hit rate
Alvarez et al. (2016) [1]	Evolutionary	L1I &L1D	GE	Trimaran, Dinero IV, SimpleScalar, CACTI	Mediabench	NA	Execution time & energy
Vieira et al. (2023) [14]	Simulation-based evaluation	Cache hierarchies	Simulation	gem5	CAP Bench	NA	Miss rate & execution characteristics
Singh et al. (2019) [15]	ML regression	Near Memory Computing	DoE, RF	LLVM, Ramulator, Pintool, Scikit-learn	PolyBench and Rodinia	12	IPC & power
Ipek et al. (2006) [16]	ML predictive modelling	Memory hierarchy and CPU	Processor design parameters, ANN	SESC, CACTI	SPEC CPU 2000,SPEC OMP,NAS	15	IPC
Sen & Imam (2019) [17]	ML multi-model regression	Hybrid main-memory	SVM, RF, ANN and GB	NVMain, gem5	STREAM and HPGC	2	Power, latency, bandwidth, & memory read/write
Hasan et al. (2021) [18]	ML multi-model regression	DRAM	SVM, RF, and GB	NVMain, gem5, Scikitlearn	CosmoGAN and LeNet	2	Power, latency, bandwidth, & memory read/write
Khakhaeng & Chantrapornchai (2016) [19]	ML classification	L1	Trace collection, MLP	Pintool, SMPCache, Weka	NUMineBench	3	Line size
Vazquez et al. (2019) [20]	ML classification	L1	Execution characteristics collection of applications, ANN	Simplaesscalar, Matlab	EEMBC Mibench	15	Line size
Elakkumanan et al. (2005) [21]	Data mining	Cache hierarchy	Cache access statistics, DT	SimICS_x86, see5	C4.5, Apriori	2	Number of cache levels & sizes
Navarro et al. (2020) [22]	ML classification	L1D	Frequency of dynamic instructions, RF	gem5, CACTI, Weka	Mibench, another C program	507	Optimal L1D
Thasneema & Bhaskaran (2021) [23]	ML classification	Cache hierarchy	Execution time and power, Random subspace	gem5, CACTI, Weka	PARSEC	7	Optimum cache levels
Ling et al. (2017) [24]	ML regression	L1D	Stack distance theory, ANN	gem5, Matlab	Mibench, Mediabench, Mobybench	`15	LRU-cache conflict misses
Note: NA shows approaches that do not need ML training							

In summary, earlier studies have proposed analytical, evolutionary, and simulation-based methods for analyzing cache configurations and cache DSEs, and have demonstrated their effectiveness. The existing literature either emphasizes selective cache parameters, relies on complex mathematical equations and slow architectural simulations, or restricts itself to specific design points, thereby revealing limitations in scalability across diverse applications and large cache configurations, as well as high exploration costs. Many prior works have successfully applied ML-based methods to reduce exploration time. They effectively applied ML, underscoring its success in predicting cache performance and in DSE. However, many ML studies have considered miss rate and power consumption individually, with no integrated ML-based predictive DSE method. No complete DSE framework is explicitly suited to L1. Many ML prior works considered only one or two applications.

Due to distinct benchmark applications, classification-based model evaluation metrics, different feature sets, higher-level (CPU/GPU) performance metrics, and trace-based studies, the direct quantitative comparison of results across most studies is limited. We have discussed these studies qualitatively in terms of methodology and problem framing. In one way or another, the prior ML-based studies gathered data for their training datasets from gem5, CACTI, and Pintool, encouraging the use of the same toolchain in this work.

To address the limitations of conventional methods for cache evaluation and DSE, we propose regression-based predictive models to rapidly estimate an application's level-1 cache miss rate and power consumption across various cache configurations and application-specific characteristics. Moreover, because of the regression-based method, we could explore the design space in more detail by facilitating Pareto-optimal analysis.

4. Definition of the Problem

Conventional methods are analytical and simulation-based. An analytical method provides good DSE speedup but requires a complex mathematical formula and is less accurate. The simulation-based method yields precise results, but it is very time-consuming. These limitations of traditional methods show their limitation in early cache DSE. Furthermore, selecting miss-rate-power-optimal cache configurations from a vast design space is challenging. To address these challenges, this work aims to accelerate the prediction of L1D and L1I cache miss rates and power consumption across different cache configurations and applications using ML methods, thereby reducing the need for additional simulations. Furthermore, the research will involve selecting Pareto-optimal cache configurations.

5. Objective and Scope of Work

1. To train and propose regression ML models to forecast the L1D and L1I miss rates and power consumption values for an application.
2. To compare and evaluate different regression models and to choose the most fitted ML model.
3. To examine the generalization ability of the proposed ML models by quickly predicting the miss rate and power consumption value of new (test) applications using different cache configurations and applications' characteristics as accurately as possible.
4. To identify miss-rate-power-optimal cache configurations by using Pareto-optimal analysis.
5. To provide faster cache DSE than an architectural simulator.

Scope of Work

- The work is carried out for level-1 data and instruction cache memory, with cache size, line size, and associativity as the design configuration.
- The evaluation is based on 20 applications from 5 widely used benchmark suites (GraphBig, Rodinia, Mibench, Cortexsuite, and PolybenchC) across diverse domains, including graph processing, embedded computing, ML, and linear algebra.
- Applications are simulated in the gem5 simulator to obtain L1D and L1I miss rates, while power consumption is measured using gem5 and CACTI.
- Applications are profiled using the Intel Pintool to collect their characteristics, such as the number of data transfers, control flow, ALU instructions, basic blocks, and dynamic instructions.
- Two predictive regression models are developed using the Weka tool to predict the target values: an additive regression with random forest as the base learner for the miss rate, and a multilayer perceptron for power consumption.
- A comparative evaluation with other ML models is conducted using popular and suitable metrics, such as the correlation coefficient, mean absolute error (MAE), and root mean squared error (RMSE).

- Miss-rate-power-optimal cache configurations are selected using a Pareto-optimal approach and validated against the architectural simulator.

6. Original Contribution

1. Target values generation using a simulation and analytical tool

- The miss rate is collected using the gem5 (widely used open-source simulator), and power consumption-related statistics are gathered with gem5 and CACTI (open-source) to prepare a training dataset for ML models.

2. Feature set generation to prepare the training dataset to train ML models

- The feature includes cache configurations (varying cache size, line size, and associativity) and the application's characteristics.
- Application's characteristics are gathered using the Intel Pintool (an open-source tool).

3. Regression ML-based method for level-1 cache performance prediction

- Trained the regression ML models using the Weka tool (open-source) for L1D and L1I cache miss rate and power consumption using the application's characteristics and different cache configurations.

4. A comparative evaluation with other ML models for model selection

- By applying 10-fold cross-validation, multiple regression models are systematically trained and compared.
- The most suitable model for the miss rate is additive regression with a random forest base learner (ARRF), and power consumption prediction is achieved using a multilayer perceptron (MLP), yielding a high correlation coefficient (≈ 0.99) and low error metrics (MAE and RMSE).

5. Generalization of the proposed models

- The proposed predictive models can quickly estimate the miss rate and power consumption of new (test) applications, yielding estimates closer to simulation results.

6. Selection of the miss-rate-power-optimal cache configuration

- Pareto-optimal analysis (multi-objective optimization) was employed to identify the L1I and L1D cache configurations for an application that provides a balanced trade-

off between miss rate and power consumption, based on the proposed model's predicted values.

- A case study demonstrated the model's capacity to select a cache configuration of an application similar to gem5.

7. Considerable acceleration over simulation

- Demonstrated that the proposed ML models predict results for 28 cache configurations of a bfs application in less than five seconds, compared to 9+ hours using gem5.
- Achieved 12x-148x speedup in cache DSE without compromising prediction accuracy.

7. Research Methodology

We proposed an ML regression method to predict the L1D miss rate in our previous work [30]. In this work, we extend the methodology for L1D power prediction, L1I miss rate & power consumption prediction, and for Pareto-optimal analysis as a multi-objective optimization.

The key goal of our methodology is to quickly and accurately predict miss rate and power, which are used to identify an application's miss-rate-power-optimal level-1 cache configurations for cache DSE. It requires a feature set (cache design parameters and application characteristics), label values (miss rate and power consumption), and an ML model to identify patterns among them and predict the target values across the entire design space (28 cache configurations). In this section, we describe research methodology as shown in Figure 1. The methodology involves several steps, including generating the training dataset, training the ML model, evaluating the proposed models on a new application, and selecting Pareto-optimal cache configurations.

7.1 Training Dataset Generation

Firstly, applications were studied, and 20 suitable applications were selected from 5 widely known benchmark suites spanning diverse domains, including graph processing, embedded computing, ML, and linear algebra. The selected applications are listed in Table 2. The applications were downloaded with their provided input file and Makefile. The application's C/C++ source code was compiled using its Makefile with GCC/G++ and converted into an

executable. The executable code of an application, along with its input file, was simulated using the gem5 v22.1 simulator with the x86 ISA, the TimingSimple CPU, and SE (syscall emulation) mode, with L1D and L1I cache configurations varying in size, line size, and associativity. Likewise, each benchmark application was simulated using the gem5 simulator.

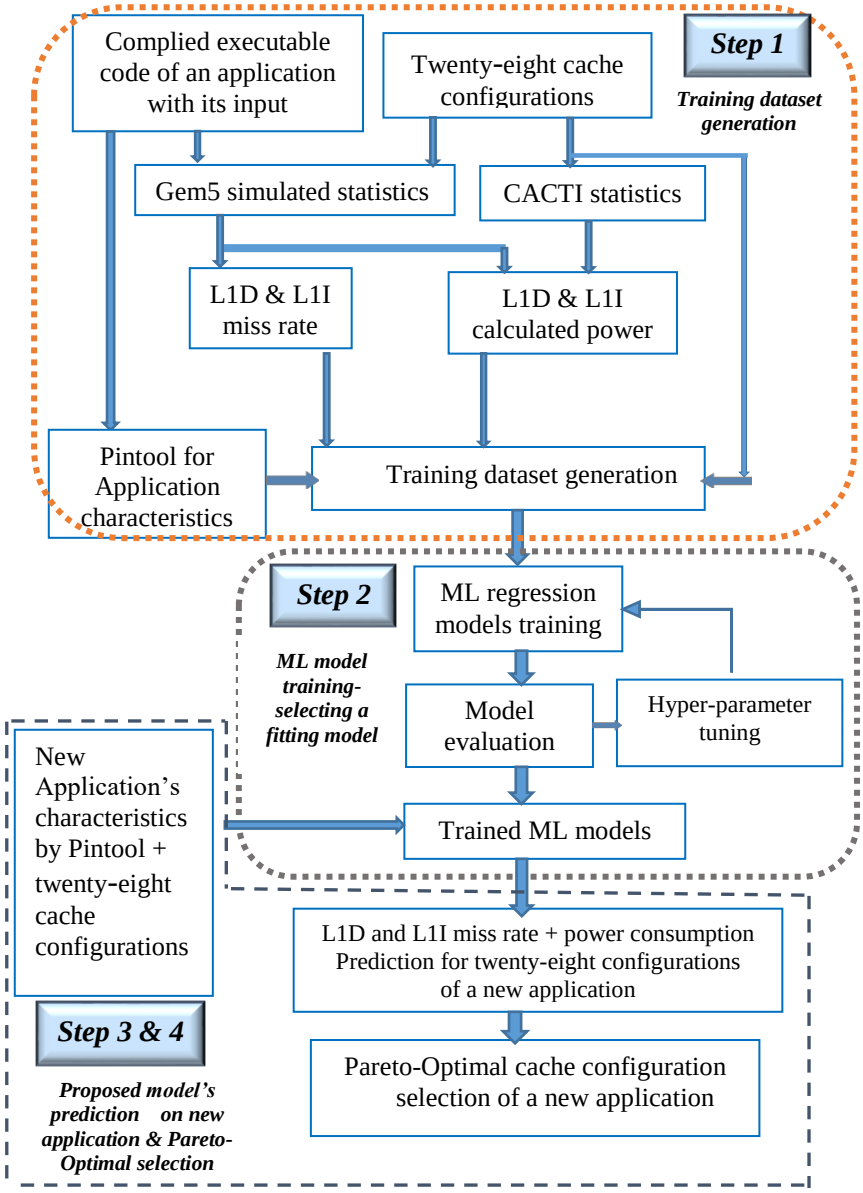

Figure 1. The proposed methodology for predicting L1 cache miss rates and power consumption

Gem5 [31] is a well-known simulator written in C++ and Python. It is actively maintained by developers and widely accepted in academia, industry, and research institutions. All 20 applications were simulated across 28 L1D and L1I cache configurations, varying cache size, line size, and associativity, as listed in Table 3. Their values are powers of two. We chose a

fixed LRU replacement policy. After completing the simulation, gem5 generates simulation-related statistics in the stats.txt output file in the m5out folder. The miss rate values of L1D & L1I, the number of read accesses of L1D & L1I, the number of write accesses of L1D, and the simulation time (sim-second is the time required for the application to execute, as computed by gem5, using a cache configuration) were extracted from the stats.txt output file.

Table 2. Benchmark applications [30]

Benchmark suits	Application
GraphBIG[32]	Connected Component (cc)
	Degree Centrality (dc)
	Depth-First Search (dfs)
	Graph Construction (gc)
	Graph Update (gu)
	Page Rank (pr)
	sssp
	Triangle Count (tc)
	ufind
	ut
PolyBenchC[33]	atax
	bicg
	gemver
	gesummv
	mvt
Mibench[34]	dijkshatra
	patricia
	qsort
Cortexsuit [35]	Principal Component Analysis (pca)
Rodinia[36]	Breadth-First Search (bfs)

Table 3. Applied cache configurations used for experimentation and as features in the training dataset

Cache configuration parameters	Value
Cache size	1KB to 1024KB
Line size	8B to 64B
Associativity	1-way to 16-way

We applied the same methodology as in our previous work [30] to generate the miss rate as explained above. For power prediction, we used the CACTI tool to generate dynamic read & write energy, and leakage power, for all 28 L1D and L1I cache configurations, separately for each. CACTI is a cache analytical tool that provides power, area, and access & cycle time information, which was developed in Hewlett-Packard labs. CACTI [37]-generated statistics were combined with the gem5 output statistics to calculate power consumption for

the target values in the training dataset. We considered the dynamic and leakage (static) power of the L1 cache. The power calculation is derived from equations 1 and 2.

For L1D cache:

$$PL1D_{total} = PL1D_{dynamic} + PL1D_{leakage} \ldots\ldots\ldots\ldots\ldots\ldots\ldots Equ.1$$

Where,

$$PL1D\_dynamic = \frac{(NL1Dr \times EL1Dr) + (NL1Dw \times EL1Dw)}{T}$$

For L1I cache:

$$PL1I_{total} = PL1I_{dynamic} + PL1I_{leakage} \ldots\ldots\ldots\ldots\ldots\ldots\ldots Equ.2$$

Where,

$$PL1I\_dynamic = \frac{(\mathrm{N}L1Ir \times \mathrm{E}L1Ir)}{T}$$

NL1Dr = number of data reads (read access), *NL1Dw* = number of data writes (write access), *EL1Dr* = dynamic energy per data read (joules/read), *EL1Dw* = dynamic energy per data write (joules/write), *PL1D*_leakage = data cache leakage power (watts), *NL1Ir* = number of instruction reads, *EL1Ir* = dynamic energy per instruction read, *PL1I*_leakage = instruction cache leakage power (watts), *T* = sim-seconds (s), and *(N×E)/T* has units J/s = Watts. Since instruction caches are designed for read operations only, we used read-related statistics to achieve this goal. The dynamic cache power is calculated as shown in the above equations by following the architecture-level approach of Wang et al. [38] and the 'accesses × energy-per-access' formulation of Michael et al. [39].

Thus, we collected target values (miss rate and power consumption) for 20 applications × 28 cache configurations = resulting in 560 samples for the training dataset. The simulator takes a long time to produce its output, so we deliberately considered 28 useful cache configurations to obtain a practical target value for the training dataset. Then, we collected the application's characteristics data, such as dynamic instruction count, basic block count, and category mix (data transfer, control flow, ALU instruction count, AVX instruction count), load-store (memory read/write, memory write, memory read, no memory instruction count) from MypinTool, Catmixtool, and ldstmixtool of Intel Pintool v3.31 (a dynamic instrumentation tool) [40][41]. We applied feature construction to ensure that all application characteristics could be covered. Thus, the training dataset consists of independent variables

(cache configurations and application's characteristics) and dependent variables (miss rate and calculated power consumption). We prepared a training dataset for L1D and L1I. For L1D, two training datasets are prepared: one for the miss-rate target and another for the power-consumption target. Same for L1I.

7.2 ML Models Training Process

Thus, four training datasets were finally prepared for the training process. The generated training dataset serves as input to Weka v3.8.6 [42][43] for training and evaluating our model. Weka is a data mining package implemented in Java. It offers a comprehensive collection of ML algorithms, along with data preprocessing and evaluation tools. We applied a leave-one-application-out [15][30] concept. We considered 19 applications in the training dataset and reserved one for testing. Likewise, iteratively, each application was tested. We removed redundant and irrelevant features (predictors) from all the collected ones. We selected only suitable features using a feature selection filter: the ClassifierAttributeEval attribute evaluator with the Ranker search method in Weka. Training an ML model with relevant features reduces overfitting, lowers computational cost, improves model performance, and enhances generalization. The finalized training set contains 9 features (cache configuration [cache size, line size, associativity], and application's characteristics [input file size, dynamic instruction count, basic block count, data transfer, control flow, ALU instruction count]) [30]. We trained multiple regression ML models and selected the best-fitting model for our training dataset using Weka's 10-fold cross-validation for robust, reliable evaluation. Then, after calculating the average, we reported evaluation metrics, including the correlation coefficient, MAE, and RMSE, for all trained models. The comparison among various trained models is provided in the next section. The ARRF and MLP best predict the desired miss rate and power consumption value, respectively. We tuned the hyperparameter values of ARRF and MLP to train the best-fitting model (a high correlation coefficient and lower error). For the additive regression model, we set the learning rate (shrinkage) to 0.9 and used a random forest as the base learner with 200 iterations, as in our previous work [30]. For MLP, we tuned the learning rate to 0.01, the number of hidden layers to 4, and the number of training epochs to 400, and set normalizedAttributes to the true value (as it can improve the network's performance). We kept the remaining hyperparameters at their default value for both models. Once the model's training process is complete, the time-consuming simulation part is no longer required. The

new application's feature set can be prepared using cache configurations and application characteristics (from Pintool) for the proposed model.

7.3 Pareto-Optimal Analysis

After predicting the target values for an application, we can determine the cache configurations that satisfy constraints on the miss rate and power consumption. A configuration is called Pareto-optimal if it cannot be improved in any objective without compromising another objective (in this case, miss rate and power consumption). It is impossible to find a single cache configuration that simultaneously optimizes both miss rate and power consumption. It can be obtained using Pareto-optimal analysis [26][44][45]. The Pareto-optimal set comprises cache configurations that represent different balances among the objectives. In Pareto optimization, two cache configurations are compared based on whether one dominates the other in terms of performance. It is used to significantly narrow the cache design search space and identify the miss-rate-power-optimal cache configurations. Using the predicted miss rate and power consumption values from the proposed models across various cache configurations, we conducted a Pareto-optimal analysis to determine the optimal cache configurations for L1D and L1I cache DSE in terms of miss rate and power consumption. To perform a Pareto analysis, we used Python code that implements the Brute-Force Non-Dominated Sorting algorithm and the weighted-sum method with min-max normalization. The input parameters for the Python code were 28 cache configurations, miss rate values, and power consumption values. We used the Brute-Force Non-Dominated Sorting, a multi-objective optimization method [45], to identify Pareto dominance by comparing each cache configuration to all others. This method found configurations that are not dominated by any other configuration, chose those that are not simultaneously inferior in both miss rate and power, and formed the Pareto-optimal set. In the cache DSE process, from these Pareto-optimal configurations, one can choose, based on the cache design requirements, either the low-miss-rate configuration (for high-performing applications), the low-power configuration (for prioritized battery life), or the balanced configuration. It depends on the constraints of the target application. The balanced configurations support the most performance benefits without exponentially increasing power penalties. After finding the Pareto optimal set, the Python code used the weighted sum method with min–max normalization (a Multi-Criteria Decision Making technique) - a score-based ranking method - to select the final configuration that provides a balanced trade-

off between miss rate and power consumption. We also performed the same analysis on gem5-simulated miss rate and power consumption values for comparison. In the next section, the case study compares the identified optimal cache configuration for the application, based on predicted values, with the simulated data.

8. Key Results and Observations of the Study

This section presents the study's key results by comparing the performance of trained regression models and evaluating the generalization capacity of the proposed models across all applications. This section also identifies optimal cache configurations via Pareto analysis, reports the achieved speedup relative to the architecture simulator, and compares the proposed model with prior work quantitatively.

8.1 Comparison of Regression Models

Table 4 (for L1D) and Table 5 (For L1I) compare various trained models using 10-fold cross-validation on the training dataset to find the best-fitted model of our training dataset, among them ARRF and MLP outperformed in predicting miss rate and power consumption, respectively, by giving a high correlation coefficient and low errors, as highlighted in the tables.

Table 4. Comparison of regression models for miss rate and power prediction of L1D

Target	ML models	Correlation coefficient	MAE	RMSE
Miss rate [30]	ARRF	0.989	0.006	0.011
	RF	0.976	0.011	0.024
	RandomSubspace	0.913	0.027	0.046
	Knn	0.881	0.027	0.055
	MLP	0.854	0.037	0.056
Power	MLP	0.994	0.004	0.006
	ARRF	0.986	0.005	0.008
	SMOReg	0.986	0.004	0.008
	RandomSubspace	0.966	0.023	0.037
	Knn	0.964	0.007	0.026

The AR is a gradient boosting technique that can fit a model to the residuals left by the base classifier from the previous iteration and make predictions by combining them, thereby improving its performance with each iteration [43]. The RF is known to be used for

regression and classification-based problems. It builds many decision trees from subsets of the data and combines their outputs, thereby reducing overfitting and improving accuracy. It is highly significant and widely used for producing reliable results. It uses bagging, making it an ensemble learning method. An ensemble learning technique can enhance the model's performance by combining different models. Thus, the combination of additive regression and RF (an ensemble learning technique) in our proposed model yields better performance than other ML models.

Table 5. Comparison of regression models for miss rate and power prediction of L1I

Target	ML models	Correlation coefficient	MAE	RMSE
Miss rate	ARRF	0.987	0.002	0.005
	RF	0.977	0.003	0.007
	RandomSubspace	0.896	0.010	0.017
	Knn	0.923	0.007	0.014
	MLP	0.742	0.017	0.023
Power	MLP	0.992	0.005	0.006
	ARRF	0.989	0.007	0.010
	SMOReg	0.980	0.008	0.009
	RandomSubspace	0.968	0.030	0.046
	Knn	0.966	0.009	0.029

An MLP is a type of ANN [7] that uses backpropagation for training and supports a feedforward NN with one or more hidden layers. ANNs are recognized as effective models for predicting power consumption. Compared to linear regression, an ANN can model nonlinearity. Compared to a decision tree, an ANN can be trained to output a continuous range of values, whereas a decision tree is typically trained to output a few discrete values [3].

8.2 Generalization Ability of Proposed Models

As mentioned in the previous section, using the leave-one-out concept [15], we evaluated the proposed model's generalization ability to predict the target value by calculating its MAE and RMSE using the formulas provided in [17]. Figure 2 compares the proposed ARRF model's MAE with two other ML models (ARRF vs. an RF model and a random subspace) across 20 applications for miss rate prediction on L1I (a) and L1D (b). Figure 3 compares the proposed MLP model's MAE with those of two other ML models across 20 applications for power prediction, specifically L1I (a) and L1D (b). These comparisons demonstrate that,

in most cases, the MAE values are lower than those of other ML models, confirming that the proposed models are the best fit for predicting the target values.

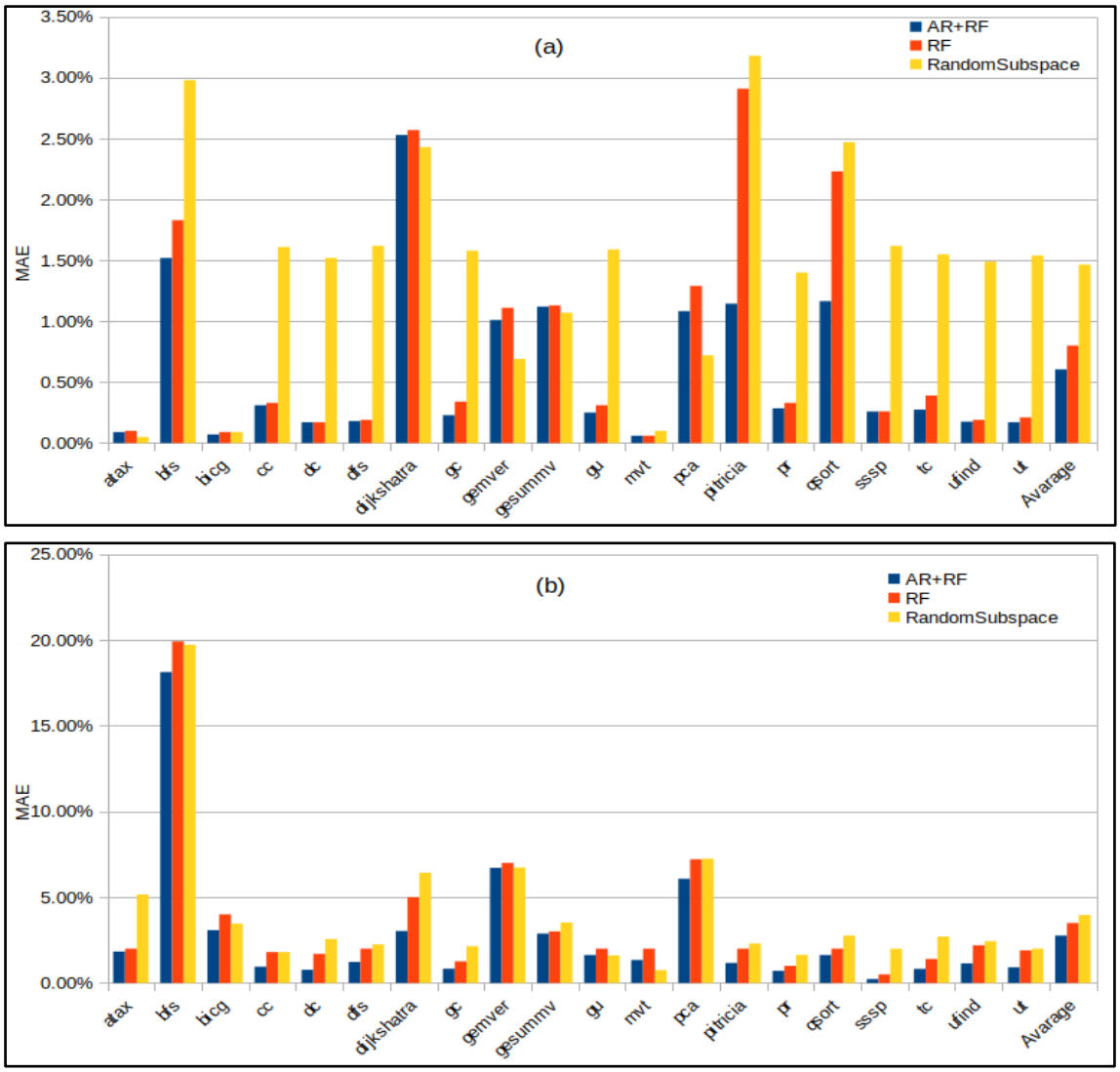

Figure 2. MAE for miss rate prediction using ARRF vs. other methods for L1I (a) and L1D (b)

Some applications, like bfs, dijkstra, and gemver, showed higher error values due to their irregular memory access patterns and higher instruction counts, making their cache activity more challenging for the trained model to learn, given models trained on a limited set of features and benchmarks.

Table 6 illustrates the proposed model's generalization ability by calculating the average MAE and RMSE for miss rate and power consumption prediction on new data for L1I and L1D. Overall, the proposed model's average MAE across all applications for L1I and L1D indicates adequate generalization to new applications.

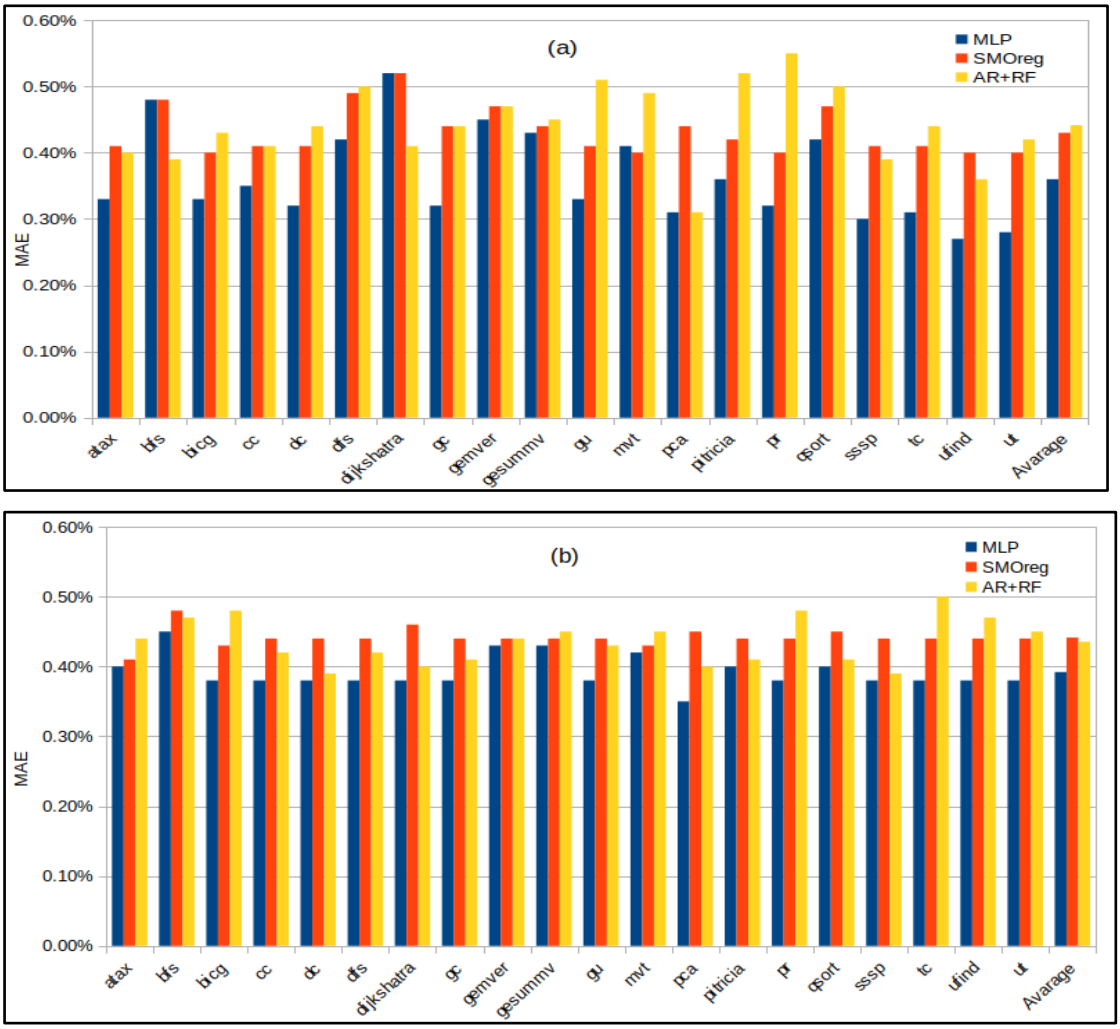

Figure 3. MAE for power prediction using MLP vs. other methods for L1I (a) and L1D (b)

Table 6. Average MAE and RMSE values from the proposed model on the new applications

Target	MAE	RMSE
For L1I		
Miss rate	0.61%	0.87%
Power	0.36%	0.48%
For L1D		
Miss rate	2.7%	3.2%
Power	0.39%	0.58%

Two different ML models are proposed to predict two target values: one for the miss rate and the other for power consumption, since the miss rate is more sensitive to application behaviour. At the same time, cache configurations influence the power consumption. The authors [17] and [18] successfully applied multiple ML models to predict different target values, suggesting that training the model according to target values is practical and acceptable for prediction.

8.3 Pareto-Optimal Cache Configuration Selection

This analysis identifies the set of Pareto-optimal (miss-rate-power-optimal) cache configurations for an application, as explained in the previous section. The Pareto-optimal analysis for cache DSE was performed based on our proposed model-predicted and gem5-simulated miss rates and power consumption. The predicted miss rates and power values for each application were evaluated across all 28 cache configurations using Pareto-optimal analysis. From the set of Pareto-optimal cache configurations, we select one that provides a balanced trade-off between miss rate and power consumption. For the case study, the predicted and simulated miss-rate-power-balanced L1I cache configurations, along with their miss rates and power values for four applications (ut, dc, dfs, and bicg), are shown in Table 7. An ut, dc, and dfs show superior alignment between predicted and simulated optimal configurations. The miss rate and power values obtained from the proposed method's predictions were very close to those from simulations. It shows that ML-based prediction and selection can be effectively applied to cache DSE in these applications. For bicg, the simulated Pareto-optimal selection was 32KB-64B-4-way, while the proposed model predicted 32KB-64B-16-way. Both configurations showed a balanced trade-off between miss rate and power, but differed slightly in associativity.

Table 7. Miss-rate-power-balanced L1I cache configurations for representative applications

Application	Method	Balanced Cache Configuration (Size-Line-Assoc.)	Miss Rate	Power(W)
ut	Predicted	32KB-64B-8-way	0.0001	0.0156
	Simulated	32KB-64B -8-way	0.0001	0.0159
dc	Predicted	32KB-64B-4-way	0.0004	0.0150
	Simulated	32KB-64B-4-way	0.0002	0.0152
dfs	Predicted	32KB-64B -8-way	0.0001	0.0156
	Simulated	32KB-64B -8-way	0.0002	0.0159
bicg	Predicted	32KB-64B -16-way	0.0002	0.0193
	Simulated	32KB-64B -4-way	0.0005	0.0154

Table 8 presents the predicted and simulated balanced Pareto-optimal L1I cache configurations for the remaining applications. In some cases, such as bfs, dijkstra, gemver, and qsort, the proposed models suggested a larger, more associative cache as optimal, whereas simulations favoured a smaller, direct-mapped cache. i.e., for the bfs application, according to the proposed model, the balanced cache configuration is 32KB-64B-16-way,

whereas, according to the simulation, the optimal one is 16KB-64B-1-way. The proposed model identified alternative yet valid configurations, demonstrating its potential to expand the set of efficient design points. In particular, the ML-based approach significantly reduces the analysis time from hours (in simulation) to seconds, with minimal loss in accuracy, enabling its practical adoption in early design phases.

Table 8. Predicted & simulated miss-rate-power-balanced L1I cache configurations for remaining applications

Application	Predicted Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]	Simulated Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]	Application	Predicted Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]	Simulated Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]
atax	32-64-4	8-64-1	mvt	32-64-4	16-64-1
bfs	32-64-16	16-64-1	pca	8-64-1	32-64-1
cc	32-64-8	32-64-8	pitricia	8-64-1	32-64-16
dijkshatra	32-64-16	8-64-1	pr	32-64-8	32-64-8
gc	16-64-1	16-64-1	qsort	32-64-16	8-64-1
gemver	32-64-16	8-64-1	sssp	16-64-1	32-64-8
gesummv	8-64-1	8-64-1	tc	32-64-4	32-64-4
gu	32-64-8	32-64-8	ufind	32-64-4	32-64-4

Table 9 shows the predicted and simulated miss-rate-power-balanced L1D cache configurations, along with their miss rates and power values for dc, gu, sssp, ut, pr, and qsort applications. Table 10 represents the predicted and simulated balanced Pareto-optimal L1D cache configurations for the remaining applications. In most cases, the predicted and simulated Pareto-optimal results closely matched, demonstrating the reliability of the ML-driven DSE.

The predicted and simulated comparative analysis shows the effectiveness and limitations of the proposed ML-assisted DSE. Our ML-based method achieves high accuracy for applications with regular memory access, with predicted and simulated Pareto-optimal analyses nearly identical, providing sufficient precision for early-stage design decisions.

However, the ML models deviate for memory-intensive or irregular-access applications (e.g., bfs, dijkshatra, and gemver), leading to differences in prediction. This deviation highlights the challenge of replicating typical memory behaviours using models trained on a limited set of features and benchmarks.

Table 9. Miss-rate-power-balanced L1D cache configurations for representative applications

Applications	Method	Balanced Cache Configuration (Size-Line-Assoc.)	Miss Rate	Power(W)
dc	Predicted	32KB-64B-4-way	0.0180	0.0110
	Simulated	32KB-64B-4-way	0.0176	0.0114
gu	Predicted	32KB-64B-4-way	0.016	0.011
	Simulated	32KB-64B-4-way	0.017	0.012
sssp	Predicted	32KB-64B-8-way	0.0174	0.0125
	Simulated	32KB-64B-8-way	0.0180	0.0130
ut	Predicted	32KB-64B-4-way	0.017	0.011
	Simulated	32KB-64B-4-way	0.018	0.0123
pr	Predicted	32KB-64B-8-way	0.018	0.012
	Simulated	32KB-64B-8-way	0.026	0.013
qsort	Predicted	32KB-64B-4-way	0.002	0.011
	Simulated	32KB-64B-4-way	0.005	0.012

Table 10. Predicted & simulated miss-rate-power-balanced L1D cache configurations for remaining applications

Application	Predicted Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]	Simulated Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]	Application	Predicted Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]	Simulated Balanced Cache Configuration [Size (KB)-Line (B)-Assoc.]
atax	64-64-1	64-64-1	gemver	64-64-1	32-64-8
bfs	32-64-8	64-64-1	gesummv	32-64-8	32-64-8
bicg	32-64-2	32-64-8	mvt	32-64-8	32-64-8
cc	32-64-4	32-64-4	pca	32-64-4	32-64-4
dfs	32-64-4	32-64-4	pitricia	32-64-4	4-64-1
dijkshatra	32-64-8	32-64-4	tc	32-64-4	32-64-4
gc	32-64-4	32-64-4	ufind	32-64-4	32-64-4

8.4 Achieved Speedup of ML-based Cache Prediction over gem5

The gem5 simulator can simulate only one cache configuration at a time, making it very slow to analyze all 28 cache configurations for an application. i.e., 60 to 942 host clock minutes are required to simulate pca and gemver, respectively, during our experiments. At the same time, the proposed model simultaneously predicts the target values for all 28 cache configurations of an application in under two seconds. Some extra time is required to prepare the feature set for a new application to serve as input to the proposed model, e.g., collecting

the application's characteristics from the Pintool, which takes approximately 5 minutes. Table 11 presents the time required to obtain target values using the gem5 simulator and the proposed model for all evaluated applications.

Table 11. Time taken to get targeted values using the gem5 simulator and the proposed model [30]

Applications	Simulation time in minutes	Prediction time in minutes
atax	662	< 5 for all applications
bfs	540	
bicg	661	
cc	140	
dc	288	
dfs	140	
dijkshatra	180	
gc	298	
gemver	942	
gesummv	311	
gu	242	
mvt	645	
pca	60	
pitrica	140	
pr	180	
qsort	150	
sssp	285	
tc	304	
ufind	270	
ut	254	

Thus, the overall speedup ranges from 12x to 148x compared to the simulator, enabling the target value to be achieved across all 28 cache configurations. In particular, the ML-based method significantly decreases the analysis time from hours (in simulation) to seconds with minimal loss in accuracy, enabling its practical adoption in the early cache design phase. Thus, a key finding is the need for a hybrid exploration strategy that leverages ML-based models to identify candidate configurations, refine the design space, and rapidly prune the cache configurations. Then, cycle-accurate simulations can be used for final validation. Thus, it balances exploration accuracy and efficiency assurance, making it valuable for large-scale or multilevel cache design studies.

8.5 Comparison with Prior ML-based Cache DSE Work

As mentioned in Section 3, the literature review, one of the early efforts to employ an ML regression model to predict the L1D miss rate was reported in [24], demonstrating the possibility of using a trained ML model in cache performance prediction, thereby reducing

reliance on time-consuming simulation. Figure 4 compares the average RMSE values of our L1D miss-rate proposed model [30] with those reported in [24] for both training and new data in the context of L1D miss-rate prediction.

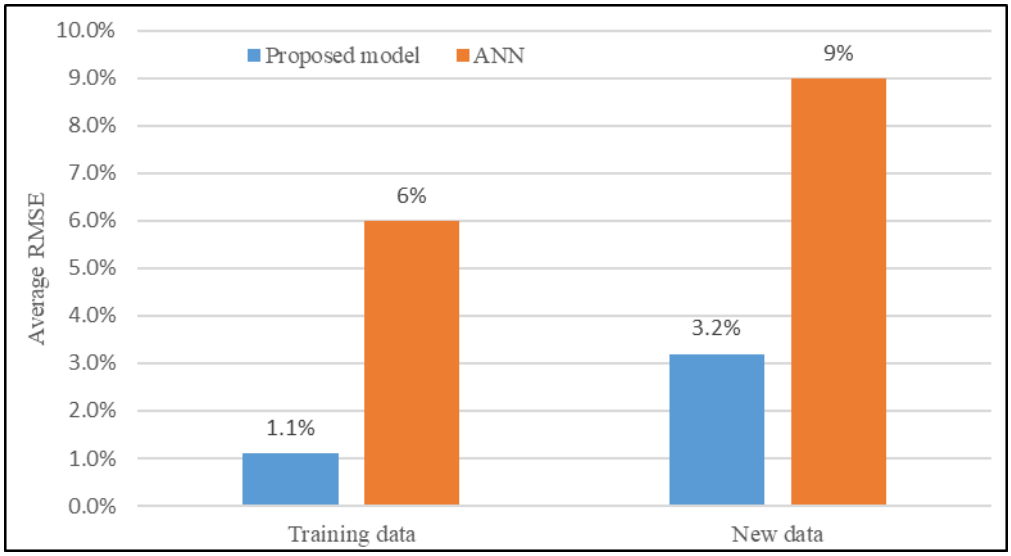

Figure 4. Comparison of average RMSE values between our proposed ML model [30] and the regression-based ANN model from prior work [24] for L1D miss rate

However, their work primarily targets miss-rate prediction across 15 applications in 3 benchmark suites and does not consider power consumption. In contrast, this work covers the scope of L1 cache DSE by predicting miss rate and power consumption using ML models, based on 20 diverse applications from 5 benchmark suites.

9. Conclusions

This work presents a novel approach for accelerating level-1 cache DSE using an ML method. The proposed models can predict miss rate and power consumption with high accuracy and low error using 10-fold cross-validation, and enable the selection of Pareto-optimal cache configurations, similar to those obtained via simulation. The proposed models can predict the target value for a new application across 28 cache configurations without requiring additional simulations, achieving speedups of 12x to 148x over gem5 simulations. It enables the designer to rapidly explore the design space, identify trade-offs between performance and power, and select optimal cache configurations. The findings demonstrate the potential of ML-assisted cache DSE, which reduces simulation overhead and supports early design decisions with reliable predictions.

This research work represents a significant step forward in further developing an ML-driven approach for evaluating cache performance parameters and DSE across various cache configurations, with the ultimate aim of designing an efficient cache with new capabilities for the next generation of applications.

Future Work

The mismatches in the predictions highlight opportunities for future work, including adding more features and applications to expand the training dataset and explore diverse cache configurations. That can improve the generalization ability. The framework can be further extended to predict additional cache performance metrics, such as execution time and IPC. It can also be applied to multilevel caches (e.g., L2 and L3). Another promising area of future work is integrating transfer learning or reinforcement learning techniques to adapt predictive models to new architectures and workloads.

10. List of publications

1. Hetal Dave and Nirali Kotak (2022). Critical analysis of cache memory performance in terms of miss rate and power consumption. International Journal of Embedded Systems (Scopus indexed, WoS), Inderscience Publication, ISSN: 1741-1076, 15(6), pp.516–524. Doi: https://doi.org/10.1504/IJES.2022.129810. (Status: Published).

2. Hetal Dave and Nirali Kotak (2024). An Analysis of Cache Configuration's Impacts on the Miss Rate of Big Data Applications Using Gem5. Serbian Journal of Electrical Engineering (Scopus indexed), University of Kragujevac Faculty of Technical Sciences in Cacak Publication, ISSN: 2217-7183, 21(2), pp.217–234. Doi: https://doi.org/10.2298/SJEE2402217D. (Status: Published).

3. Hetal Dave, Nirali Kotak, and Jay Teraiya (2025). Hybrid Machine Learning Model for Cache Performance Prediction and Design Space Exploration. Revista de Informática Teórica e Aplicada - RITA (Scopus indexed), Federal University of Rio Grande do Sul, Institute of Informatics Publication, ISSN: 2175-2745, 32(3), pp.11–23. Doi: https://doi.org/10.22456/2175-2745.146689. (Status: Published).

4. Hetal Dave, Nirali Kotak, and Jay Teraiya (2025). Accelerating Cache Design: ML-Based Prediction and Pareto Optimal Selection of Instruction Cache Configurations. Journal of

Integrated Circuits and Systems (Scopus indexed journal), Brazilian Microelectronics Society Publication, ISSN: 1872-0234, (Status: Under Review).

11. References

[1] D. Álvarez, J. Colmenar, J. Risco-Martín, J. Lanchares, and O. Garnica, "Optimizing L1 cache for embedded systems through grammatical evolution," *Soft Comput.*, vol. 20, no. 6, pp. 2451–2465, 2016, doi: 10.1007/s00500-015-1653-1.

[2] Y. Liang and T. Mitra, "An analytical approach for fast and accurate design space exploration of instruction caches," *Trans. Embed. Comput. Syst.*, vol. 13, no. 3, pp. 1–29, 2013, doi: 10.1145/2539036.2539039.

[3] R. Vazquez, A. Gordon-Ross, and G. Stitt, "Energy Prediction for Cache Tuning in Embedded Systems," in *37th International Conference on Computer Design (ICCD)*, 2019, pp. 630–637. doi: 10.1109/ICCD46524.2019.00091.

[4] H. Dave and N. Kotak, "An analysis of cache configuration's impacts on the miss rate of big data applications using gem5," *Serbian J. Electr. Eng.*, vol. 21, no. 2, pp. 217–234, 2024, doi: 10.2298/sjee2402217d.

[5] R. Piscitelli and A. D. Pimentel, "Design space pruning through hybrid analysis in system-level design space exploration," in *Design, Automation and Test in Europe, Dresden, Germany*, 2012, pp. 781–786. doi: 10.1109/date.2012.6176600.

[6] A. D. Pimentel, "Methodologies for design space exploration," in *Handbook of Computer Architecture*, A. Chattopadhyay, Ed. Singapore: Springer, 2022, pp. 1–31. doi: 10.1007/978-981-97-9314-3_23.

[7] N. Wu and Y. Xie, "A Survey of Machine Learning for Computer Architecture and Systems," *ACM Comput. Surv.*, vol. 55, no. 3, pp. 1–37, 2022, doi: 10.1145/3494523.

[8] R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, "Machine Learning and Manycore Systems Design: A Serendipitous Symbiosis," *Computer (Long. Beach. Calif).*, vol. 51, no. 7, pp. 66–77, 2018, doi: 10.1109/MC.2018.3011040.

[9] J. R. Doppa, J. Rosca, and P. Bogdan, "Autonomous Design Space Exploration of Computing Systems for Sustainability: Opportunities and Challenges," in *IEEE Design and Test*, 2019, vol. 36, no. 5, pp. 35–43. doi: 10.1109/MDAT.2019.2932894.

[10] T. S. Ajani, A. L. Imoize, and A. A. Atayero, "An overview of machine learning within embedded and mobile devices-optimizations and applications," *Sensors*, vol. 21, no. 13, pp. 1–44, 2021, doi: 10.3390/s21134412.

[11] M. D. Hill and A. J. Smith, "Evaluating Associativity in CPU Caches," *IEEE Trans. Comput.*, vol. 38, no. 12, pp. 1612–1630, 1989, doi: 10.1109/12.40842.

[12] C. Zhang, F. Vahid, and W. Najjar, "A highly configurable cache architecture for embedded systems," in *IEEE 30th Annual International Symposium on Computer Architecture*, 2003, pp. 136–146. doi: 10.1145/859634.859635.

[13] H. V. Dave and N. A. Kotak, "Critical analysis of cache memory performance concerning miss rate and power consumption," *Int. J. Embed. Syst.*, vol. 15, no. 6, pp. 516–524, 2022, doi: 10.1504/IJES.2022.129810.

[14] J. V. A. Vieira, M. A. Souza, and H. C. de Freitas, "Performance Evaluation of Intel and AMD Memory Hierarchies Using a Simulation-driven Approach With Gem5," in *High Performance Computing Systems*, 2023, pp. 17–24. doi: 10.5753/wscad_estendido.2023.235791.

[15] G. Singh *et al.*, "NAPEL: Near-memory computing application performance prediction via ensemble learning," in *ACM/IEEE 56th Design Automation Conference, Las Vegas, NV, USA*, 2019, no. 1–6. doi: 10.1145/3316781.3317867.

[16] E. Ipek, B. Supinski, S. A. Mckee, M. Schulz, and R. Caruana, "Efficiently Exploring Architectural Design Spaces via Predictive Modeling," in *ACM SIGOPS Operating Systems Review*, 2006, vol. 40, pp. 195–206. doi: https://doi.org/10.1145/1168917.1168882.

[17] S. Sen and N. Imam, "Machine learning based design space exploration for hybrid main-memory design," in *ACM International Symposium on Memory Systems*, 2019, pp. 480–489. doi: 10.1145/3357526.3357544.

[18] S. M. Hasan, N. Imam, R. Kannan, S. Yoginath, and K. Kurte, "Design Space Exploration of Emerging Memory Technologies for Machine Learning Applications," in *IEEE International Parallel and Distributed Processing Symposium Workshops, Portland, OR, USA*, 2021, pp. 439–448. doi: 10.1109/IPDPSW52791.2021.00075.

[19] S. Khakhaeng and C. Chantrapornchai, "On the finding proper cache prediction model using neural network," in *8th*

International Conference on Knowledge and Smart Technology, IEEE, Chiang Mai, Thailand, 2016, pp. 146–151. doi: 10.1109/KST.2016.7440514.

[20] R. Vazquez, A. Gordon-Ross, and G. Stitt, "Machine Learning-based Prediction for Dynamic Architectural Optimizations," *IEEE 10th Int. Green Sustain. Comput. Conf.*, pp. 1–6, 2019, doi: 10.1109/IGSC48788.2019.8957207.

[21] P. Elakkumanan, L. Liu, V. K. Vankadara, and R. Sridhar, "CHIDDAM: A data mining based technique for cache hierarchy determination in commercial applications," in *48th Midwest Symposium on Circuits and Systems, Covington, KY, USA*, 2005, vol. 2, pp. 1888–1891. doi: 10.1109/MWSCAS.2005.1594493.

[22] O. Navarro, J. Yudi, J. Hoffmann, H. Gerardo, M. Hernandez, and M. Hübner, "A machine learning methodology for cache memory design based on dynamic instructions," *ACM Trans. Embed. Comput. Syst.*, vol. 19, no. 2, p. 20, 2020, doi: 10.1145/3376920.

[23] V. T. Thasneema and S. Bhaskaran, "A Classification Model for Predicting Suitable Cache Level in a Multicore Architecture," in *IEEE International Conference on Communication, Control and Information Sciences, Proceedings*, 2021, vol. 1, pp. 1–6. doi: 10.1109/ICCISc52257.2021.9484900.

[24] K. Ji, M. Ling, Y. Zhang, and L. Shi, "An artificial neural network model of LRU-cache misses on out-of-order embedded processors," *Microprocess. Microsyst.*, vol. 50, pp. 66–79, 2017, doi: 10.1016/j.micpro.2017.02.005.

[25] H. Calborean, R. Jahr, T. Ungerer, and L. Vintan, "Optimizing a Superscalar System using Multi-objective Design Space Exploration," in *18th International Conference on Control Systems and Computer Science, Bucharest, Romania*, 2011, vol. 1, pp. 339–346.

[26] J. Michanan, R. Dewri, and M. J. Rutherford, "Understanding the power-performance tradeoff through Pareto analysis of live performance data," *Int. Green Comput. Conf.*, pp. 1–8, 2014, doi: 10.1109/IGCC.2014.7039164.

[27] J. Díaz Álvarez, J. L. Risco-Martín, and J. M. Colmenar, "Multi-objective optimization of energy consumption and execution time in a single level cache memory for embedded systems," *J. Syst. Softw.*, vol. 111, pp. 200–212, 2016, doi: 10.1016/j.jss.2015.10.012.

[28] A. Jooya, N. Dimopoulos, and A. Baniasadi, "Multiobjective GPU design space exploration optimization," *Microprocess. Microsyst.*, vol. 69, pp. 198–210, 2019, doi: 10.1016/j.micpro.2019.06.001.

[29] A. Deshwal, N. Kanappan Jayakodi, B. Kumar Joardar, J. Rao Doppa, and P. Pratim Pande, "Moos: A multi-objective design space exploration and optimization framework for NoC enabled manycore systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5, p. 23, 2019, doi: 10.1145/3358206.

[30] H. V. Dave, N. A. Kotak, and J. B. Teraiya, "Hybrid Machine Learning Model for Cache Performance Prediction and Design Space Exploration", Revista De Informática Teórica E Aplicada (RITA), vol. 32, no. 3, pp. 11–23, 2025, doi: 10.22456/2175-2745.146689

[31] J. Lowe-Power *et al.*, "The gem5 Simulator: Version 20.0+," 2020, [Online]. Available: http://arxiv.org/abs/2007.03152

[32] L. Nai, Y. Xia, I. Tanase, H. Kim, and C. Lin, "GraphBIG : Understanding Graph Computing in the Context of Industrial Solutions," in *International Conference for High Performance Computing, Networking, Storage and Analysis, Austin, TX, USA*, 2015, pp. 1–12. doi: 10.1145/2807591.2807626.

[33] M. Abella-Gonzalez, P. Carollo-Fernandez, L. N. Pouchet, F. Rastello, and G. Rodriguez, "PolyBench/Python: Benchmarking Python environments with polyhedral optimizations," in *30th ACM SIGPLAN International Conference on Compiler Construction, New York, NY, USA*, 2021, pp. 59–70. doi: 10.1145/3446804.3446842.

[34] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "MiBench: A free, commercially representative embedded benchmark suite," in *IEEE International Workshop on Workload Characterization, Austin, TX, USA*, 2001, pp. 3–14. doi: 10.1109/WWC.2001.990739.

[35] S. Thomas *et al.*, "Cortex Suite: A synthetic brain benchmark suite," in *International Symposium on Workload Characterization, IEEE, Raleigh, NC, USA*, 2014, pp. 76–79. doi: 10.1109/IISWC.2014.6983043.

[36] S. Che *et al.*, "Rodinia: A benchmark suite for heterogeneous computing," in *International Symposium on Workload Characterization, IEEE, Austin, TX, USA*, 2009, pp. 44–54. doi: 10.1109/IISWC.2009.5306797.

[37] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6 . 0 : A Tool to Model Large Caches," in *A Quarterly Journal In Modern Foreign Literatures*, 2009, no. HPL-2009-85, pp. 0–24. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.140.1426&rep=rep1&type=pdf

[38] W. Wang, S. Ranka, and P. Mishra, "Energy-aware dynamic reconfiguration algorithms for real-time multitasking systems," *Sustain. Comput. Informatics Syst.*, vol. 1, no. 1, pp. 35–45, 2011, doi: 10.1016/j.suscom.2010.10.006.

[39] M. D. Powell, S. Yang, B. Falsafi, K. Roy, and V. TN, "An Energy-Efficient High-Performance Deep-Submicron

Instruction Cache," *IEEE TVLSI Spec. issue low-power Des.*, vol. 9, no. 1, pp. 1–13, 2001.

[40] C. K. Luk *et al.*, "Pin: Building customized program analysis tools with dynamic instrumentation," in *SIGPLAN Conference on Programming Language Design*, 2005, vol. 40, no. 6, pp. 190–200. doi: 10.1145/1064978.1065034.

[41] "Pin 3.31 - A dynamic binary instrumentation." http://software.intel.com/En-Us/Articles/Pintool-Downloads

[42] "Weka- 3.8.6." https://sourceforge.net/projects/weka/files/weka-3-8/3.8.6/

[43] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software," *ACM SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009, doi: 10.1145/1656274.1656278.

[44] X. B. Hu, M. Wang, and E. Di Paolo, "Calculating complete and exact pareto front for multiobjective optimization: A new deterministic approach for discrete problems," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1088–1101, 2013, doi: 10.1109/TSMCB.2012.2223756.

[45] K. Deb, "Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction," in *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing*, 2011. doi: 10.1007/978-0-85729-652-8_1.