

# On Extending Amdahl’s law to Learn Computer Performance

Chaitanya Poolla<sup>a</sup>, Rahul Saxena<sup>a</sup>

<sup>a</sup>*Intel Corporation, 3600 Juliette Ln, Santa Clara, CA 95054, USA*

---

## Abstract

The problem of learning parallel computer performance is investigated in the context of multicore processors. Given a fixed workload, the effect of varying system configuration on performance is sought. Conventionally, the performance speedup due to a single resource enhancement is formulated using Amdahl’s law. However, in case of multiple configurable resources the conventional formulation results in several disconnected speedup equations that cannot be combined together to determine the overall speedup. To solve this problem, we propose to (1) extend Amdahl’s law to accommodate multiple configurable resources into the overall speedup equation, and (2) transform the speedup equation into a multivariable regression problem suitable for machine learning. Using experimental data from fifty-eight tests spanning two benchmarks (*SPECCPU 2017* and *PCMark 10*) and four hardware platforms (*Intel Xeon 8180M*, *AMD EPYC 7702P*, *Intel CoffeeLake 8700K*, and *AMD Ryzen 3900X*), analytical models are developed and cross-validated. Findings indicate that in most cases, the models result in an average cross-validated accuracy higher than 95%, thereby validating the proposed extension of Amdahl’s law. The proposed methodology enables rapid generation of multivariable analytical models to support future industrial development, optimization, and simulation needs.

*Keywords:* Computer performance, analytical modeling, Amdahl’s law, machine learning, regression

---

## 1. Introduction

Modern computers consist of complex interdependent subsystems working in tandem to provide a seamless user experience. These subsystems realize

a variety of underlying implementations and architectures in hardware [1]. Computer performance is evaluated by benchmarking, a process that subjects computer systems to standard load conditions and monitors performance metrics of interest [2]. Depending on the end goal, performance metrics could represent aspects such as response time, throughput, or power consumption.

The system performance can be obtained by different methods such as collecting measurements, running simulations, or predicting from analytical models [3]. In case of measurements, the workload is configured to run on the test system under standard conditions and the performance is reported. While the accuracy could be relatively high, the experimenter has limited feasible configurations of the system to choose from. In case of simulations, the subsystems and their interactions can be modeled extensively. Depending on the complexity of the simulator, there is a trade off between accuracy, coverage, development time, and evaluation time. For example, complex simulators often require more time to achieve a higher accuracy, and simpler simulators require less time at the expense of lower accuracy. Further, simulators require extensive validation for reliable use which is time-consuming [4] [5]. However, simulators allow for investigating performance of next generation systems with hypothetical system configurations.

Given the limited scope of measurement configurations and the complex nature of simulations, it is of interest to explore simpler analytical models partly based on benchmark measurements. Such models allow one to investigate a larger scope of system configurations in a short time and hence blend the advantages of both measurement and simulation-based methods. Previous research suggests that easy to use and understandable analytical models that can explore large design spaces are imperative for architectural analysis [6] [7]. In this study, we focus on the problem of developing analytical models for parallel computing in the context of multi-core processors based on limited benchmark measurements.

For a given benchmark, the performance of a multi-core processor depends on several factors related to the system configuration, hardware architecture, and the software stack used [8]. After fixing the hardware and software stack, measurements are obtained by running the benchmark at a specified system configuration. The system configuration is varied by variables such as the number of cores, threads per core, core frequency, memory frequency, and cache sizes [9]. Each of these variables is set to a feasible value prior to the execution of the benchmark. Thus, each system configuration can be viewed as an instance of the corresponding machine. Accordingly, the same

computer hardware can be used to build several system configurations each of which provides a corresponding set of system resources for executing the benchmark.

The execution of a benchmark on a given system configuration generates performance values, commonly known as scores. These scores allow for comparison of performance across different configurations, architectures, and software stacks for a common benchmark. Typically, a baseline and a test system are chosen for executing the benchmark to obtain the baseline and test score respectively. When the benchmark performance metric measures the execution time or latency, the ratio of the baseline to test scores provides the performance speedup on the test system in comparison to the baseline system.

The theoretical speedup expected from a test system with improved resources is provided by Amdahl's law [10] [11]. In case of multi-core processors, the speedup could result from scaling up a system resource such as the number of cores. In the conventional formulation of the Amdahl's law, a program is comprised of two parts: A parallelizable part whose performance is affected by scaling up system resources and a sequential part whose performance remains unaffected by scale. It is important to note that the problem size or workload remains fixed in the formulation of Amdahl's law. If the problem size increases parallelly with the resource, Gustafson's model is to be used to compute speedup [12]. In this work, we assume a fixed problem size within the scope of Amdahl's law. Over the recent years, variants of the Amdahl's law have been proposed to consider other aspects of system such as the memory wall problem [13] or the synchronization problem [4]. More recently, a survey of methods to improve the semantic power of the p-fraction and computation capability improvement index can be found in [14]. It is important to note that while these variants are based on sound theoretical considerations they lack corroboration to empirical data from silicon measurements. Further, in these cases Amdahl's law is primarily employed to predict the theoretical speedup by changing one system resource at a time. This results in several equations, each corresponding to scaling an isolated system resource given fixed values of other system resources. Since each equation stems from different assumptions regarding the values of the other system resources, it is not feasible to combine these equations to account for the effect of multiple system variables simultaneously [15]. Thus, there is a need for a framework that can account for the effects of multiple variables simultaneously.

To this end, we propose an extension to Amdahl's law by adopting a

multi-variable data-driven approach. From the viewpoint of program execution, the need for this extension arises because real-world applications view the system not as comprised of isolated resources but as multiple resources interacting together to support the application. Therefore, the modeling procedure requires integrating these interaction effects. Further, such an extension allows data from real-world experiments to determine the model coefficients and hence offer insights on both the parallelizable fractions and expected scores for new configurations. In this manner, the extension accommodates multiple system resources simultaneously and offer a medium for experimental data to predict system performance on new system configurations. Related works on regression-based analytical modeling are found in [16] and [17]. In [16], a regression-based predictive modeling approach is proposed consisting of less than 30 predictors, human-specified interactions, and 2000-4000 simulated observations. Each predictor was chosen based on the strength of its marginal correlations without accounting for the variation in other predictors. In the case of Stargazer [17], a forward selection-based stepwise regression was employed to study the GPU design space tradeoffs. The study used 10 architectural parameters and 30-300 data points to achieve 85-99% accuracy based on sample-train-test method. The use of stepwise regression is susceptible to model bias due to incorrect specification in the best subset selection problem. Further, both regression-based studies do not employ cross-validation, resulting in lack of unbiased generalization error. By contrast, we employ Amdahl’s law-based models to achieve an average of  $\approx 95\%$  cross-validation accuracy with relatively fewer data points (20-200) given seven independent resources variables. To the best of the authors knowledge, this is the first work to reveal a direct connection between multivariable Amdahl’s law and machine learning-based modeling. The contributions of this work are as follows:

1. An extension of Amdahl’s law to accommodate simultaneous scalings across multiple system resources and their interactions.
2. Statistical regression techniques to generate multi-variable analytical models based on measurements data.
3. A demonstration of cross-validation and prediction efficacy of the proposed analytical models based on data from experiments with industry standard benchmarks and hardware.

The remainder of the paper is structured as follows: Section 2 proposes the methodology based on Amdahl’s law and its reformulation into a multiple

regression framework. Section 3 describes the experiment setup including the Systems Under Test (SUTs) and the benchmarks considered. Section 4 discusses the model design, training, and validation. Section 5 presents the results along with the scope and modeling assumptions followed by the conclusion in Section 6.

## 2. Methodology

Consider a system with configuration denoted by  $\mathcal{C}$ . Let this configuration be specified by  $k$  resource variables represented by the sequence  $\{r_i^{\mathcal{C}}\}_{i=1}^k \subset \mathbb{R}^{+k}$ . Within a multi-core processor environment, these resource variables may represent core count, multi-threading state, core frequency, cache size, memory frequency, and other configurable variables. For a given hardware architecture, benchmark and software stack, let the baseline and the test system configurations be represented by  $\mathcal{C}_b$  and  $\mathcal{C}_t$ , respectively. Accordingly, the baseline and test system resources are specified by  $\{r_i^{\mathcal{C}_b}\}_{i=1}^k$  and  $\{r_i^{\mathcal{C}_t}\}_{i=1}^k$ , respectively. Further, let the baseline and test performance be specified by  $P^{\mathcal{C}_b} \in \mathbb{R}^+$  and  $P^{\mathcal{C}_t} \in \mathbb{R}^+$ , respectively.

### 2.1. Amdahl's Law: Enhancement due to a single resource

Given the above notation, let  $f_i$  denote the fraction of the program enhanced exclusively by the  $i^{\text{th}}$  system resource. Assuming the performance metric is the inverse of the execution time, we can employ Amdahl's law [11] to derive the theoretical performance speedup as:

$$\mathcal{S}(i, f_i, \mathcal{C}_b, \mathcal{C}_t) = \frac{P^{\mathcal{C}_t}}{P^{\mathcal{C}_b}} \Big|_{i^c} = \frac{1}{(1 - f_i) + f_i \left( \frac{r_i^{\mathcal{C}_b}}{r_i^{\mathcal{C}_t}} \right)} \quad (1)$$

where,  $\mathcal{S}$  denotes the performance speedup,  $i$  denotes the index of the  $i^{\text{th}}$  resource, and  $r_i^{\mathcal{C}_b}$ ,  $r_i^{\mathcal{C}_t}$  denote the  $i^{\text{th}}$  resource of the baseline and test systems, respectively. The symbol  $\Big|_{i^c}$  refers to the operating condition that all resources other than the  $i^{\text{th}}$  are equal across the baseline and test systems. The ratio  $r_i^{\mathcal{C}_t}/r_i^{\mathcal{C}_b}$  denotes the resource enhancement. We note that in most cases even the parallel fraction  $f_i$  is not infinitely parallelizable [18]. In other words, improving the  $i^{\text{th}}$  resource by a factor of  $x$  does not necessarily result

in an equivalent speedup even within the parallelizable fraction  $f_i$  of the program. Nevertheless, the speedup equation offered by Amdahl's law is useful to formulate the effect of parallelism stemming from resource enhancements. Further, we note that the above equation only considers the speedup resulting from an enhancement exclusively due to the  $i^{th}$  resource - *at fixed values of the other resources*. Similarly, the speedup resulting from enhancing only the  $j^{th}$  resource may be given by:

$$\mathcal{S}(j, f_j, C_b, C_t) = \frac{P^{C_t}}{P^{C_b}} \Big|_{j^c} = \frac{1}{(1 - f_j) + f_j \left( \frac{r_j^{C_b}}{r_j^{C_t}} \right)} \quad (2)$$

### 2.2. Amdahl's law: Need for reformulation

Given the performance speedups exclusively due to  $i^{th}$  and  $j^{th}$  resources in equations 1 and 2 respectively, it is of interest to determine the overall speedup due to enhancing both  $i^{th}$  and  $j^{th}$  resources simultaneously. The overall speedup is particularly significant as it allows one to examine the effect of varying both resources simultaneously, unlike equations 1 or 2 which hold resources  $j$  or  $i$  at fixed values, respectively. From a statistical viewpoint, equations 1 and 2 can be viewed as simple linear regression models of the inverse of the performance speedup as a function of the  $i^{th}$  and  $j^{th}$  resource enhancements, respectively. The overall speedup equation due to varying resources  $i$  and  $j$  would need to combine the performance speedups due to the individual resource enhancements and the interactions between them. Unfortunately, there is no well-defined mechanism to combine the individual performance speedups in equations 1 and 2 directly to obtain the overall performance speedup [15]. Hence, we reconsider the Amdahl's law formulation to be able to incorporate multiple resource variables.

### 2.3. Amdahl's Law: Enhancement due to mutually exclusive resources

Consider a scenario where mutually exclusive fractions of the program are enhanced by the speedups from the individual resources. In this case, the resources corresponding to the indices  $1, 2, \dots, k$  enhance the respective fractions  $f_1, f_2, \dots, f_k$  of the program. Accordingly, the performance speedup may be expressed as:

$$\frac{P^{C_t}}{P^{C_b}} = \mathcal{S}(\cdot) = \frac{1}{(1 - f_1 - f_2 - \dots - f_k) + \sum_{m=1}^k f_m \left( \frac{r_m^{C_b}}{r_m^{C_t}} \right)} \quad (3)$$

where  $\mathcal{S}(\cdot)$  denotes the shorthand notation of the multi-resource speedup function  $\mathcal{S}\left(\{i\}_{i=1}^k, \{f_i\}_{i=1}^k, \mathcal{C}_b, \mathcal{C}_t\right)$ .

### 2.3.1. Enhancements due to resource interactions

The above formulation combines the individual effects of several resources into the overall performance speedup. However, during the program execution, interaction between multiple resources occur. Thus, the performance speedup formulation must include the effects of these resource interactions<sup>1</sup>. Theoretically, interactions can occur between a minimum of two factors and a maximum of  $k$ -factors, thereby resulting in possibly  ${}^kC_2 + {}^kC_3 + \dots + {}^kC_k = 2^k - k - 1$  unique combinations, which is exponential in the number of individual resources  $k$ . In order to deal with this exponential complexity, we resort to the *hierarchical ordering principle* whereby higher order interactions involving three or more factors are deemed very rarely significant and hence ignored [19]. Accordingly, the speedup equation based on Amdahl's law with single and two-factor resource interactions can be expressed as:

$$\frac{P^{\mathcal{C}_t}}{P^{\mathcal{C}_b}} = \mathcal{S}(\cdot) = \frac{1}{\left(1 - \sum_{m=1}^k f_m - \sum_{m=2}^k \sum_{n=1}^{m-1} f_{mn}\right) + \sum_{m=1}^k f_m \left(\frac{r_m^{\mathcal{C}_b}}{r_m^{\mathcal{C}_t}}\right) + \sum_{m=2}^k \sum_{n=1}^{m-1} f_{mn} \left(\frac{r_m^{\mathcal{C}_b} r_n^{\mathcal{C}_b}}{r_m^{\mathcal{C}_t} r_n^{\mathcal{C}_t}}\right)} \quad (4)$$

In equation 4, the denominator of the right hand side represents the serial and parallel fractions contributing to the overall speedup. Specifically,  $f_m$  represents the program fraction enhanced exclusively by the resource enhancement  $r_m^{\mathcal{C}_b}/r_m^{\mathcal{C}_t}$  and  $f_{mn}$  represents the program fraction enhanced by combined resource enhancement  $r_m^{\mathcal{C}_b} r_n^{\mathcal{C}_b}/r_m^{\mathcal{C}_t} r_n^{\mathcal{C}_t}$ . In this manner, up to two-factor interactions are captured in the speedup formulation.

### 2.3.2. Enhancements due to higher order effects

In some cases, it is desirable to include higher order interactions into the speedup equation. Consider the three term interaction  $\mathcal{I}_{mnp} = (r_m^{\mathcal{C}_b} r_n^{\mathcal{C}_b} r_p^{\mathcal{C}_b}/r_m^{\mathcal{C}_t} r_n^{\mathcal{C}_t} r_p^{\mathcal{C}_t})$  along with the corresponding parallelizable program fraction  $f_{mnp}$ . This effect can be incorporated by adding  $f_{mnp}(\mathcal{I}_{mnp} - 1)$  to the denominator. Similarly, one may also incorporate other nonlinearities into the speedup equation.

---

<sup>1</sup>In the context of experimental design, resource variables are known as factors. Thus, an interaction between  $p$  ( $\leq k$ ) resources is referred to as a *p-factor interaction*

For example, consider the resource term representing the maximum memory bandwidth available per core. This term represents a nonlinear higher order interaction effect consisting of several factors such as channel width, channel count, memory frequency, number of cores, and core frequency. In this manner, the overall speedup equation can accommodate scaling across multiple resources and relevant nonlinear interactions.

#### 2.4. Amdahl’s Law for data-driven modeling

The postulated speedup equations in Section 2.3.1 result in analytical models, which, while theoretically reasonable, require data for validation. The data offers insights into the contributions of the various resources and interactions for purposes of prediction. Therefore, Amdahl’s law helps in hypothesizing a predictive model which is validated based on experimental data. Depending the predictive power of a given model, its complexity may be increased or reduced via the resource terms to avoid underfitting or overfitting [15]. This methodology results in a hybrid approach where both architectural knowledge and statistical techniques codetermine a multivariable analytical model. Further discussion on analytical modeling is provided in sections 3.3 and 4.

### 3. Experiment Setup

The efficacy of the analytical models in Section 2 is determined by using data from designed experiments. For this work, we conducted four experiments involving the *SPECCPU 2017* and *PCMark 10* benchmarks executed on Intel and AMD platforms. Let these experiments be represented by  $\mathbf{E}_q \forall q \in \{1, 2, 3, 4\}$ . For each experiment, the benchmark and hardware platform combinations are shown in Table 1.

Experiment ID	Benchmark	Hardware	Operating System	Compiler
$\mathbf{E}_1$	SPECCPU 2017	Intel Xeon 8180M	RHEL 7.3	ICC 19u4
$\mathbf{E}_2$	SPECCPU 2017	AMD EPYC 7702P	Ubuntu 19.04	GCC 8.2
$\mathbf{E}_3$	PCMark 10	Intel CoffeeLake 8700K	Win 10 Enterprise 1252	N/A (Pre-compiled)
$\mathbf{E}_4$	PCMark 10	AMD Ryzen 3900X	Win 10 Pro 1903	N/A (Pre-compiled)

Table 1: Overview of Experiments

#### 3.1. Workload settings

For each benchmark, the settings employed during experimentation are described below:

### 3.1.1. SPEC CPU 2017

The SPEC CPU 2017 benchmark consists of industry-standardized, CPU intensive suites for determining compute performance by stressing a system’s processor, memory subsystem, and compiler [20]. Among these suites, the *SPECrate 2017 Integer* and *SPECrate 2017 Floating Point* suites were both used to evaluate the performance of the *Intel Xeon 8180M* and *AMD EPYC 7702P* systems. The operational settings for both systems are depicted in the **Operating System** and **Compiler** columns of Table 1.

### 3.1.2. PCMark 10

The PCMark 10 benchmark represents a wide range of office activities from everyday productivity tasks to taxing work with digital media content [21]. This benchmark is divided into groups each of which consists of several tests. In this work, the *Essentials*, *Productivity*, and *Digital Content Creation* groups were all used to evaluate the perform of *Intel CoffeeLake 8700K* and *AMD Ryzen 3900X* systems. The operational settings for both systems are depicted in the **Operating System** and **Compiler** columns of Table 1.

For each experiment  $\mathbf{E}_q$ , let the number of runs of the benchmark be denoted by  $m_q$ . Each run results in a score corresponding to a system with a specific configuration. Throughout each experiment, the benchmark and operational settings were fixed while the system configurations were varied.

## 3.2. System configuration

The resources provided by the system enable the execution of the benchmark. However, not all resources are varied in the course of the experiment. In this work, we employ the term *resource variables* to refer only to the resources varied during the experiment. These variables are used to model the relationship between the system configuration and the benchmark performance. In general, any resource may be varied during the experiment to determine benchmark sensitivity. In the conventional sense of the Amdahl’s law, the benchmark sensitivity is quantified by the speedup due to the single resource enhancement. Based on sections 2.2 and 2.3, we attempt to quantify benchmark sensitivity as the speedup due to enhancement of mutually exclusive resources. Here, we focus on exploring the speedup due to enhancing the CPU and memory subsystems and accordingly the resource variables include one or more aspects of the CPU such as the number of cores, the number of threads, core frequency, uncore frequency, and last level cache size; and

one or more aspects of the memory subsystem such as the memory frequency and number of memory channels. Since the same resource variables support the execution of different workloads across various systems, the chosen variables generalize across systems and workloads. For each of the experiments undertaken, Table 2 depicts the range of configurations<sup>2</sup> utilized.

Resource attribute	$E_1$	$E_2$	$E_3$	$E_4$
#DataPoints ( $m_q$ )	58	20	177	38
#Cores (Min)	1	64	1	1
#Cores (Max)	28	64	6	12
Core Freq. in MHz (Min)	1800	1500	1200	2200
Core Freq. in MHz (Max)	2500	2000	3200	3800
Uncore Freq. in MHz (Min)	2200	1500	1200	2200
Uncore Freq. in MHz (Max)	2200	2000	3200	3800
LLC MB (Min)	7	128	3	64
LLC MB (Max)	38	256	12	64
Mem Freq. in MHz (Min)	2133	2666	1300	1333
Mem Freq. in MHz (Max)	2667	3200	2667	1600
#MemCH (Min)	6	8	1	4
#MemCH (Max)	6	8	2	4

Table 2: Range of the Experimental Design Spaces

In addition to the above resource variables, the simultaneous multithreading state (ON or OFF) was varied during the experiment.

### 3.3. Relation between experimental design and modeling

The relation between experiment design and modeling is crucial in developing analytical models with appropriate modeling assumptions. The number or set of points required to be collected is driven by the underlying purpose of the experiment. In most cases, the purpose is to accurately model the predictive relationship between the system resources and the benchmark output (score).

---

<sup>2</sup>In order to test the adaptability of the model across diverse design spaces, each experiment involved a subset of the possible variables.

*Explanatory and predictive modeling:* It is important to note that predictive models do not necessarily reflect the true underlying relationship. For example, the relation between system resources and benchmark scores is dynamic in nature and hence simulated in time whereas most analytical models are static. Therefore, analytical modeling is not meant to provide explanatory or "true" models of the underlying relationship but only to generate predictions based on simplified models. However, this does not necessarily imply analytical or predictive models are less accurate as they have shown to even outperform explanatory models in some cases [22]. This counter-intuitive behavior is possible due to reasons such as small magnitude of model parameters, noisy data, regressor correlations, or small sample sizes [22].

*Experiment design for analytical modeling:* The experiment design involves selecting the design space used for analytical modeling. The functional form of the analytical model is selected to represent the underlying data-generating process. Upon specifying the analytical model, the design space spanned by the resource variables, along with the corresponding benchmark scores, are used for the estimation procedure. The estimation procedure has a bearing on the set of points required [23]. In this work, we employ least squares estimation. In general, the set of points required to obtain acceptable accuracy varies based on the efficiency of the estimator. While the investigation of the optimal design space [24] is beyond the scope of the present work, we nevertheless examine diverse design spaces to assess the efficacy of the extended Amdahl's law (Equation 4) per the design scope shown in Table 2. While any random sampling or systematic design of experiments-based approaches could be employed within the feasible set of configurations, we used random sampling based on a One-Factor-At-a-Time (OFAT) approach to generate diverse design spaces. The idea behind employing diverse design spaces is to demonstrate the generalizability of the proposed extension. In what follows, we describe the procedure for model design, training, and validation.

#### 4. Model design, training, and validation

The data obtained from an experiment  $\mathbf{E}_q$  consists of  $m_q$  configurations and scores for fixed values of the benchmark and operational settings. From Section 2, it may be noted that each configuration  $\mathcal{C}$  is made up of one or more resource variables  $r_i^{\mathcal{C}}$  which are varied across runs to obtain the corresponding scores.

#### 4.1. Analytical modeling

Analytical models represent a predictive relationship between the independent system resources and the dependent target variable. Accordingly, the target and resource variables need to be specified and an underlying predictive relationship needs to be determined and validated using experimental data.

##### 4.1.1. Target variable

The target variable is the response such as the benchmark score associated with the model. If higher scores are considered better they are equivalent to the speedup. Similarly, if lower scores are considered better, they are equivalent to the inverse speedup. Here, the target variable is equivalent to the inverse speedup as detailed in Section 4.1.3. This is because *SPECCPU 2017* SpecRate and PCMark 10 output scores proportional to the speedup [20] and hence their inverse scores is the target variable.

##### 4.1.2. Resource variables

The value of the target variable is dependent on the resource variables. This dependency is specified by the model. The set of resource variables identified in Section 3.2 are incorporated into the model as model features<sup>3</sup>. We note that each feature may either be generic as depicted in equations 3, 4 or engineered using knowledge of the architecture and/or the benchmark execution. In this work, we use both types of features in the model. The generic features represent the independent variables such as frequencies and core counts. The engineered features used represent meaningful resources such as cache size available per core or the maximum bandwidth per core noted in Section 2.3.2.

##### 4.1.3. Designing Amdahl's law-based regression models

Given the target and resource variable specifications, the explanatory relationship between these variables is provided for by the extended Amdahl's law-based speedup equations similar to the forms shown in equations 3 or 4. Each of these equations offer varying degrees of predictive ability quantifiable by experimental data. As noted in Section 2.4, the explanatory relationship offered by equations such as 3, 4 is leveraged to build analytical models.

---

<sup>3</sup> $X_{k,i}$  represents the features of the model described in Equation 6.

Consider equation 4, wherein we rewrite the denominator after making the following substitutions. Let  $\alpha_0 := (1 - \sum_{m=1}^k f_m - \sum_{m=2}^k \sum_{n=1}^{m-1} f_{mn})$ ,  $\alpha_p := f_p \forall p \in \{1, \dots, k\}$ , and  $\alpha_q := f_{mn} \forall m \in \{1, \dots, k\}$ ,  $n \in \{1, \dots, m-1\}$ ,  $q \in \{k+1, \dots, \mathcal{K}\}$ , where  $\mathcal{K} = {}^k C_1 + {}^k C_2$  denotes the total number of single factors and two-factor interactions in the model. Further, let  $\tilde{X}_0 := 1$ ,  $\tilde{X}_p := (r_p^{c_t}/r_p^{c_b}) \forall p \in \{1, \dots, k\}$  and  $\tilde{X}_q := (r_m^{c_t} r_n^{c_t}/r_m^{c_b} r_n^{c_b}) \forall m \in \{1, \dots, k\}$ ,  $n \in \{1, \dots, m-1\}$ ,  $q \in \{k+1, \dots, \mathcal{K}\}$ . Accordingly, the denominator can be rewritten as  $\sum_{k=0}^{\mathcal{K}} \alpha_k / \tilde{X}_k$ , encapsulating the resource variables, interactions, and parallel fraction-based coefficients. The left side represents the performance speedup  $P^{c_t}/P^{c_b}$ . We note that the speedup equation applies to all runs of an experiment. Extending the above convention for each run  $i$ , let the speedup equivalent be denoted by  $\tilde{Y}_i$ , the  $k^{th}$  resource variable/interaction term by  $\tilde{X}_{k,i}$ , and  $\tilde{X}_0$  by  $\tilde{X}_{0,i}$ . For convenience, we will refer to all the terms  $\tilde{X}_{k,i}$  including  $\tilde{X}_{0,i}$  as resource terms. Thus, the speedup equations in Section 2.3.1 corresponding to the  $i^{th}$  run can be represented as:

$$\tilde{Y}_i = \frac{1}{\sum_{k=0}^{\mathcal{K}} \alpha_k \frac{1}{\tilde{X}_{k,i}}} \quad (5)$$

where,  $\alpha_k$  is the refactored coefficient of the  $k^{th}$  resource term and  $\mathcal{K}$  denotes the number of resource terms considered for modeling. It is easy to see that the above equation may be readily recast into a linear regression model suitable for learning:

$$Y_i = \sum_{k=0}^{\mathcal{K}} \alpha_k X_{k,i} \quad (6)$$

where,  $Y_i = \frac{1}{\tilde{Y}_i}$  and  $X_i = \frac{1}{\tilde{X}_i}$  are the reciprocal transformations used to recast equation 5. In this manner, analytical models may be designed based on Amdahl's law and regression techniques.

#### 4.1.4. Model training and cross-validation

Given a model design, its feature weights (coefficients) are determined by training the model. These coefficients specify a trained model, also known as a model instance. In this work, the models are trained using ordinary least squares implemented by the python library *scikit-learn*. The features of the model were scaled via normalization during preprocessing to improve model training.

The performance of a model needs to be evaluated to determine its validity for prediction purposes. We use *Mean Absolute Percentage Error (MAPE)* as the model evaluation metric. The validation procedure consists of evaluating the model on data not used for training. In this work, we perform model validation by *five-fold cross-validation* as it provides an accurate estimate of the expected generalization error and uses the data efficiently for validation [15]. The measurements dataset consisting of  $m_q$  observations is split into five parts called folds. Each split uses four parts for training and one part for validation as shown in Figure 1. Thus, each split reserves 80% of the data for training and 20% for validation testing. The validation errors across all folds are averaged to determine the expected generalization error of the model.

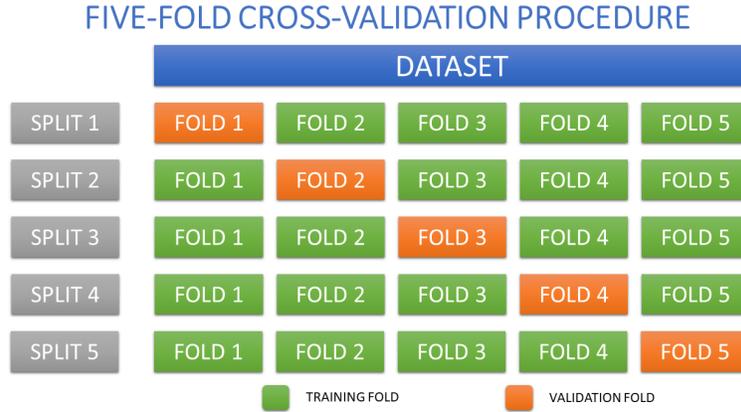


Figure 1: Five-fold cross-validation depicting training and validation folds

Once the model is cross-validated and the expected generalization errors are deemed satisfactory, it is used for prediction. The end-to-end analytical modeling flow consisting of the training, validation, and prediction phases are depicted in Figure 2.

## 5. Results and discussion

For each of the four experiments  $\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3,$  and  $\mathbf{E}_4$ , multi-variable regression models based on extended Amdahl’s law were developed and cross-validated as described in Section 4. The cross-validation results evaluate the effectiveness of the model for purposes of prediction. As stated in Section 4.1.4 we use the *Mean Absolute Percentage Error (MAPE)* as the evaluation

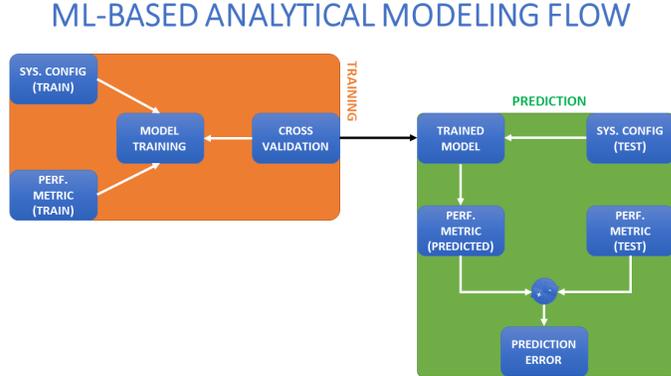


Figure 2: Analytical modeling within a Machine Learning (ML) framework

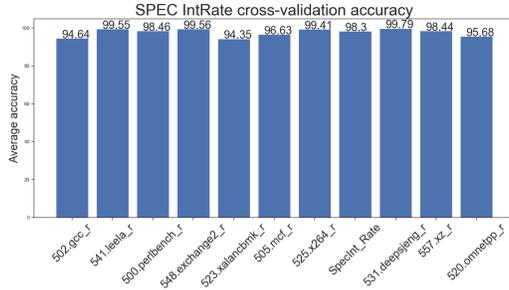
metric. For depicting the results below, we convert MAPE into the the equivalent accuracy by calculating  $100-MAPE$ . The average of the *Mean Absolute Percentage Accuracy* for all validation folds is shown in the accuracy results below for each benchmark and hardware platform.

### 5.1. Experiments involving SPEC CPU 2017

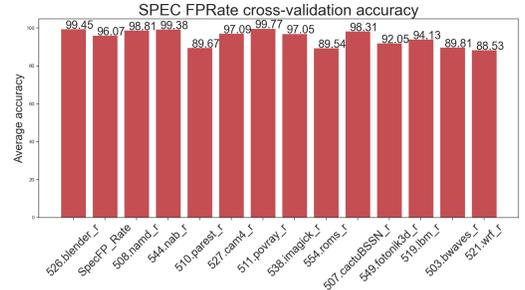
These experiments involved executing the SpecRate Integer and Floating point suites within the SPEC CPU2017 benchmark. The Integer suite consists of 11 tests and the Floating Point suite consists of 14 tests. Twenty five models were constructed and cross-validated corresponding to each of the 25 tests spanning both suites.

#### 5.1.1. $E_1$ : SPEC CPU 2017 on Intel Xeon 8180M

Figures 3a and 3b depict the cross-validation accuracy of the tests in the Integer and Floating Point suites benchmarked on Intel Xeon 8180M. In case of the Integer suite, the underlying models result in an accuracy of  $\approx 95\%$  or higher. However, in case of the Floating point tests, the *wrf-r* model resulted in the minimum accuracy of  $\approx 88\%$  and some models for other tests resulted in the maximum accuracy of  $> 99\%$ . The overall models for the Integer and Floating point suites resulted in average accuracies of 98% and 96%, respectively.

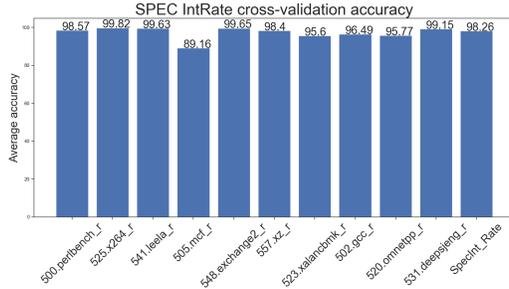


(a) SpecRate Integer

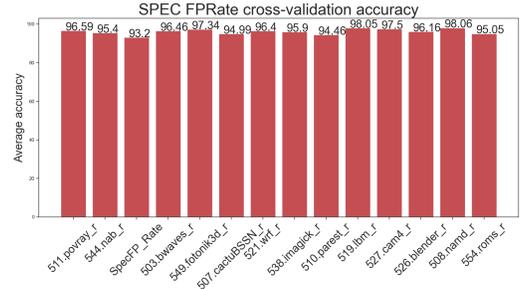


(b) SpecRate Floating Point

Figure 3: SPEC CPU 2017 cross-validation results on Intel Xeon 8180M



(a) SpecRate Integer



(b) SpecRate Floating Point

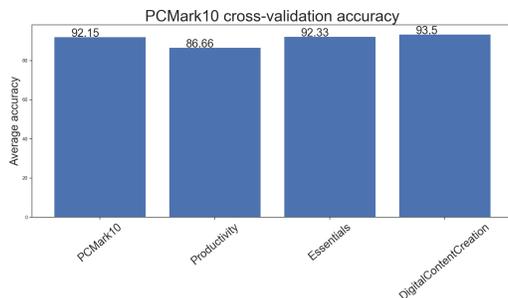
Figure 4: SPEC CPU 2017 cross-validation results on AMD EPYC 7702P

### 5.1.2. $E_2$ : SPEC CPU 2017 on AMD EPYC 7702P

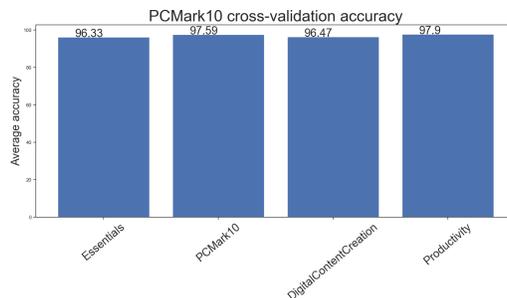
Figures 4a and 4b depict the cross-validation accuracy of the tests in the Integer and Floating Point suites benchmarked on AMD EPYC 7702P. In case of the Integer tests, model accuracies are greater than 95% with the exception of *mcf\_r* whose model accuracy was  $\approx 89\%$ . In the Floating point tests, most tests resulted in  $\approx 95\%$  accuracy or greater. The overall models for Integer and Floating point suites resulted in 98% and 93%, respectively.

### 5.2. Experiments involving PCMark 10

The experiments involved executing the PCMark 10 benchmark on each of the platforms below. The benchmark consists of three test groups namely, Essentials, Productivity, and Digital Content Creation. Each of test groups have several tests. For demonstrating model efficacy, we examine the cross-validated accuracies of four models each corresponding to either one of the



(a) PCMark10 on Intel CoffeeLake 8700K



(b) PCMark10 on AMD Ryzen 3900X

Figure 5: PCMark10 on Intel CFL 8700K & AMD Ryzen 3900X: Cross-validation results

three test groups or to the aggregate output of the PCMark 10 benchmark.

### 5.3. $\mathbf{E}_3$ : PCMark 10 on Intel CoffeeLake 8700K

Figure 5a depicts the cross-validation accuracy of the models corresponding to each of the three test groups in PCMark 10. The accuracies of the Essentials, Productivity, and Digital Content Creation models are found to be 92%, 88%, and 92%, respectively. The overall PCMark 10 model results in an average accuracy of 92%.

### 5.4. $\mathbf{E}_4$ : PCMark 10 on AMD Ryzen 3900X

Figure 5b depicts the cross-validation accuracy of the models corresponding to each of the three test groups in PCMark 10. The accuracies of the Essentials, Productivity, and Digital Content Creation models are found to be 96%, 98%, and 96%, respectively. The overall PCMark 10 model results in an accuracy of 98%. We note that the model accuracies on the Ryzen 3900X are found to be higher than the respective model accuracies on the Coffee-Lake 8700K. This indicates model learning varies according to the platform architecture for a given benchmark.

### 5.5. Results of cross-validation

The models corresponding to the fifty-eight (58) tests were cross-validated per the five-fold procedure outlined in Section 4.1.4. The cross-validation accuracies for a few tests are provided in Table 3 below.

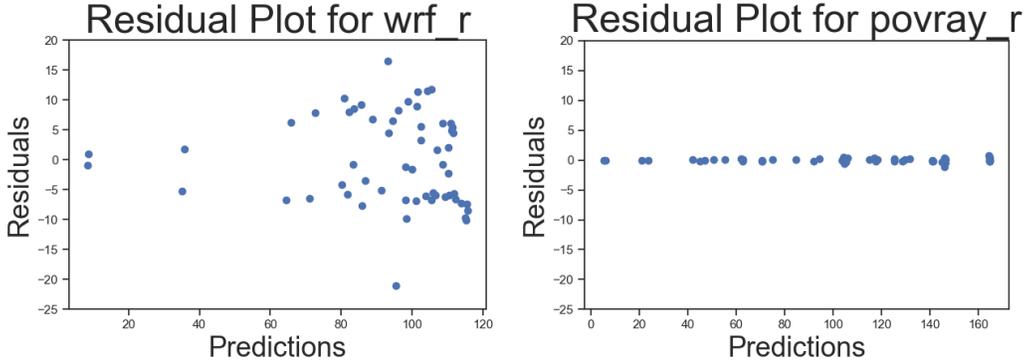
<b>Experiment</b>	<b>Test</b>	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
<b>E<sub>1</sub></b>	SpecInt_Rate	96.68%	98.85%	99.01%	98.40%	98.61%
<b>E<sub>2</sub></b>	SpecFP_Rate	79.74%	94.63%	97.60%	98.40%	95.61%
<b>E<sub>3</sub></b>	Productivity	92.78%	83.94%	82.36%	91.09%	87.83%
<b>E<sub>4</sub></b>	PCMark10 (Overall)	97.83%	99.22%	99.59%	91.44%	99.33%

Table 3: Mean Absolute Percentage Accuracy across the cross-validation folds

The configurations used in the cross-validation procedure (Figure 1) corresponding to experiment **E<sub>1</sub>** are listed in Table 4.

While most of the above fifty eight (58) have an average cross-validated accuracy  $\geq 95\%$ , we find that the accuracies approximately range between 80% and 99%, thereby indicating the scope of the model performance. The prediction accuracy is influenced by the accuracy of the estimates of the model parameters, which are estimated in this work by the least squares technique. The precision of the estimator is the highest when the conditions of the Gauss-Markov theorem are satisfied, thereby the estimator becomes the Best Linear Unbiased Estimator (BLUE) [25]. In a linear framework, the conditions require non-correlation and homoscedasticity of the error terms. For example, consider the least and most accurate predictions from the floating point suite of experiment **E<sub>1</sub>**. From Figure 3b, they belong to the wrf\_r and povray\_r tests, respectively. The residual plots of these tests are provided in Figure 6. The increasing variance of residuals with increasing values of predictions for the case of wrf\_r (Figure 6a) is suggestive of heteroscedasticity, thereby justifying a lower prediction accuracy ( $\approx 88\%$ ). On the other hand, the residual behavior in the case of povray\_r (Figure 6b) does not exhibit any tangible change in variance across the axis of predictions and is suggestive of homoscedasticity, thereby justifying a higher prediction accuracy ( $> 99\%$ ). In this manner, residual analysis is recommended to validate the assumptions pertaining to the modeling and estimation (training) procedure. It is possible to address issues related to correlation and heteroscedasticity of errors by including nonlinear features within the Amdahl’s law formulation (similar to the mention in 2.3.2) and regularized estimation procedures. The reader is referred to Section 3.3.3 of [15] for details.

In general, the model performance is not only influenced by modeling aspects such as their assumptions, structure, and training methods but also by the data variation offered by the experiment sample. For example, consider the memory frequency resource variable. If all the data points in the sam-



(a) Residual plot: wrf\_r from  $\mathbf{E}_1$ , acc.  $\approx 88\%$       (b) Residual plot: povray\_r from  $\mathbf{E}_1$ , acc.  $> 99\%$

Figure 6: Residual plots for wrf\_r and povray\_r to demonstrate impact on model accuracy

ple have the same memory frequency, there is no variation in this resource variable to be able to relate it to the system performance. It follows that the model would be unable to learn the effect of the memory frequency on the system performance. In this manner, it is necessary to ensure variation in the resource variables to model their impact on the system performance. Further, the statistical assumptions involved in least squares training need to be validated in order to build generalizable models [26]. It is also important to note that the actual execution is dynamic in nature whereas these models offer static approximations of the dynamic process to statistically relate the resource variables to the performance metric. Despite the shortcomings of analytical modeling, it results in functionally simpler models to predict system performance with reasonable accuracy as suggested by the findings of our study.

### 5.6. Use cases for new designs

Once the models are deemed satisfactory, it is not only possible to predict performance of the benchmarks within the range of the system configurations (Table 2) but pose the inverse problem of finding the range of resource variables for which a performance target is feasible along with cost considerations. This analysis can help in the design of new systems with desirable performance and cost characteristics.

## 6. Conclusion

In this work, the problem of predicting multicore system performance based on benchmark measurements is studied. The conventional Amdahl's law formulation is examined in the context of multicore systems. Resulting speedup equations were found to be limited to single resource enhancements with respective assumptions and do not allow for studying the simultaneous effect of multiple resource enhancements. To mitigate these limitations, an extension of the Amdahl's law is proposed to incorporate the effect of multiple system resources simultaneously. A regression framework suitable for learning is derived from the extended Amdahl's law. Regression models are trained on data from experiments across multiple benchmarks and architectures. The expected prediction accuracy of these models was determined by cross-validation. Results indicate an average cross-validation accuracy of  $\approx 80\% - 99\%$  depending on the benchmark and hardware platform considered. The predictive ability of the regression models across different benchmarks and architectures demonstrates the generalizability of the proposed Amdahl's law extension in effectively learning computer performance due to the simultaneous enhancement of multiple resources. The proposed method generalizes across various benchmarks and compute architectures. Future work should investigate the role of optimizing experiment design in relation to estimator accuracy and efficiency, performance reachability analysis, feature learning for rapid generation of analytical models of various product architectures and benchmarks.

## Acknowledgement

The authors thank Intel Corporation for supporting this work.

## References

- [1] A. Gonzalez, F. Latorre, G. Magklis, Processor microarchitecture: An implementation perspective, *Synthesis Lectures on Computer Architecture* 5 (1) (2010) 1–116.
- [2] J.-Y. Le Boudec, *Performance Evaluation of Computer and Communication Systems*, EPFL Press, Lausanne, Switzerland, 2010.

- [3] R. Jain, *The Art Of Computer Systems Performance Analysis: Techniques For Experimental Measurement, Simulation, And Modeling*, John Wiley & Sons, 2008.
- [4] L. Eeckhout, Computer architecture performance evaluation methods, *Synthesis Lectures on Computer Architecture* 5 (1) (2010) 1–145.
- [5] A. Akram, L. Sawalha, A survey of computer architecture simulation techniques and tools, *Ieee Access* 7 (2019) 78120–78145.
- [6] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, V. S. Pai, Challenges in computer architecture evaluation, *Computer* 36 (8) (2003) 30–36.
- [7] S. Van den Steen, S. De Pestel, M. Mechri, S. Eyerman, T. Carlson, D. Black-Schaffer, E. Hagersten, L. Eeckhout, Micro-architecture independent analytical processor performance and power modeling, in: *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, IEEE, 2015, pp. 32–41.
- [8] T. Hoefler, R. Belli, Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results, in: *Proceedings of the international conference for high performance computing, networking, storage and analysis*, 2015, pp. 1–12.
- [9] E. İpek, S. A. McKee, R. Caruana, B. R. de Supinski, M. Schulz, Efficiently exploring architectural design spaces via predictive modeling, *ACM SIGOPS Operating Systems Review* 40 (5) (2006) 195–206.
- [10] G. M. Amdahl, Validity of the single processor approach to achieving large scale computing capabilities, in: *Proceedings of the April 18-20, 1967, spring joint computer conference*, 1967, pp. 483–485.
- [11] M. D. Hill, M. R. Marty, Amdahl’s law in the multicore era, *Computer* 41 (7) (2008) 33–38.
- [12] J. L. Gustafson, Reevaluating amdahl’s law, *Communications of the ACM* 31 (5) (1988) 532–533.
- [13] X.-H. Sun, Y. Chen, Reevaluating amdahl’s law in the multicore era, *Journal of Parallel and distributed Computing* 70 (2) (2010) 183–188.

- [14] M. A. N. Al-hayanni, F. Xia, A. Rafiev, A. Romanovsky, R. Shafik, A. Yakovlev, Amdahl's law in the context of heterogeneous many-core systems—a survey, *IET Computers & Digital Techniques* 14 (4) (2020) 133–148.
- [15] G. James, D. Witten, T. Hastie, R. Tibshirani, *An introduction to statistical learning*, Vol. 112, Springer, 2013.
- [16] B. C. Lee, D. M. Brooks, Accurate and efficient regression modeling for microarchitectural performance and power prediction, *ACM SIGOPS operating systems review* 40 (5) (2006) 185–194.
- [17] W. Jia, K. A. Shaw, M. Martonosi, Stargazer: Automated regression-based gpu design space exploration, in: *2012 IEEE International Symposium on Performance Analysis of Systems & Software*, IEEE, 2012, pp. 2–13.
- [18] P. Lotfi-Kamran, H. Sarbazi-Azad, Dark silicon and the history of computing, in: *Advances in Computers*, Vol. 110, Elsevier, 2018, pp. 1–33.
- [19] M. Hamada, J. Wu, *Experiments: planning, analysis, and parameter design optimization*, Wiley New York, 2000.
- [20] J. Bucek, K.-D. Lange, J. v. Kistowski, Spec cpu2017: Next-generation compute benchmark, in: *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, 2018, pp. 41–42.
- [21] UL, Pemark 10 - the complete benchmark for the modern office, <https://benchmarks.ul.com/pcmark10> (2017).
- [22] G. Shmueli, et al., To explain or to predict?, *Statistical science* 25 (3) (2010) 289–310.
- [23] W. Krämer, Finite sample efficiency of ordinary least squares in the linear regression model with autocorrelated errors, *Journal of the American Statistical Association* 75 (372) (1980) 1005–1009.
- [24] P. Goos, B. Jones, *Optimal design of experiments: a case study approach*, John Wiley & Sons, 2011.
- [25] H. Theil, *Principles of econometrics*, Wiley, 1971.
- [26] R. H. Myers, *Classical and modern regression with applications*, Vol. 2, Duxbury press Belmont, CA, 1990.

Fold sequence	LLC (MB)	NumCores	Mem. Freq. (MHz)	NumMemCH	Core Freq. (MHz)	Uncore Freq. (MHz)	NumThreadsPerCore	LLC.per.core (derived)	B/W available per Core-Clk (derived)
VTTTT	38	8	2667	6	2500	2200	2	4.75	0.0064008000000000
VTTTT	38	12	2667	6	1800	2200	1	3.17	0.0059266666666667
VTTTT	38	12	2667	6	2200	2200	1	3.17	0.00484909090909091
VTTTT	38	12	2667	6	2500	2200	1	3.17	0.0042672000000000
VTTTT	38	12	2667	6	1800	2200	2	3.17	0.0059266666666667
VTTTT	38	12	2667	6	2200	2200	2	3.17	0.00484909090909091
VTTTT	38	12	2667	6	2500	2200	2	3.17	0.0042672000000000
VTTTT	38	18	2667	6	2500	2200	1	2.11	0.0028448000000000
VTTTT	38	18	2667	6	2500	2200	2	2.11	0.0028448000000000
VTTTT	38	1	2667	6	2500	2200	1	38.00	0.0512064000000000
VTTTT	38	1	2667	6	2500	2200	2	38.00	0.0512064000000000
VTTTT	38	20	2667	6	1800	2200	1	1.90	0.0035560000000000
TVTTT	38	20	2667	6	2200	2200	1	1.90	0.00290945454545454
TVTTT	38	20	2667	6	2500	2200	1	1.90	0.0025603200000000
TVTTT	38	20	2667	6	1800	2200	2	1.90	0.0035560000000000
TVTTT	38	20	2667	6	2200	2200	2	1.90	0.00290945454545454
TVTTT	38	20	2667	6	2500	2200	2	1.90	0.0025603200000000
TVTTT	38	22	2667	6	2500	2200	1	1.73	0.00232756363636364
TVTTT	38	22	2667	6	2500	2200	2	1.73	0.00232756363636364
TVTTT	38	24	2667	6	2500	2200	1	1.58	0.0021336000000000
TVTTT	38	24	2667	6	2500	2200	2	1.58	0.0021336000000000
TVTTT	38	28	2667	6	1800	2200	1	1.36	0.0025400000000000
TVTTT	38	28	2667	6	2000	2200	1	1.36	0.0022860000000000
TVTTT	38	28	2667	6	2200	2200	1	1.36	0.00207818181818182
TVTTT	38	28	2667	6	2500	2200	1	1.36	0.0018288000000000
TVTTT	38	28	2667	6	1800	2200	2	1.36	0.0025400000000000
TVTTT	38	28	2667	6	2000	2200	2	1.36	0.0022860000000000
TVTTT	38	28	2667	6	2200	2200	2	1.36	0.00207818181818182
TVTTT	38	28	2667	6	2500	2200	2	1.36	0.0018288000000000
TVTTT	38	4	2667	6	2500	2200	1	9.50	0.0128016000000000
TVTTT	38	4	2667	6	2500	2200	2	9.50	0.0128016000000000
TVTTT	38	8	2667	6	2500	2200	1	4.75	0.0064008000000000
TVTTT	14	28	2667	6	2500	2200	1	0.50	0.0018288000000000
TVTTT	21	28	2667	6	2500	2200	1	0.75	0.0018288000000000
TVTTT	28	28	2667	6	2500	2200	1	1.00	0.0018288000000000
TVTTT	35	28	2667	6	2500	2200	1	1.25	0.0018288000000000
TTTVT	7	28	2667	6	2500	2200	1	0.25	0.0018288000000000
TTTVT	14	28	2667	6	2500	2200	2	0.50	0.0018288000000000
TTTVT	21	28	2667	6	2500	2200	2	0.75	0.0018288000000000
TTTVT	28	28	2667	6	2500	2200	2	1.00	0.0018288000000000
TTTVT	35	28	2667	6	2500	2200	2	1.25	0.0018288000000000
TTTVT	7	28	2667	6	2500	2200	2	0.25	0.0018288000000000
TTTVT	38	12	2400	6	2500	2200	1	3.17	0.0038400000000000
TTTVT	38	12	2400	6	2500	2200	2	3.17	0.0038400000000000
TTTVT	38	20	2400	6	2500	2200	1	1.90	0.0023040000000000
TTTVT	38	20	2400	6	2500	2200	2	1.90	0.0023040000000000
TTTVT	38	24	2400	6	2500	2200	1	1.58	0.0019200000000000
TTTIV	38	24	2400	6	2500	2200	2	1.58	0.0019200000000000
TTTIV	38	28	2400	6	2500	2200	1	1.36	0.00164571428571429
TTTIV	38	28	2400	6	2500	2200	2	1.36	0.00164571428571429
TTTIV	38	12	2133	6	2500	2200	1	3.17	0.0034128000000000
TTTIV	38	12	2133	6	2500	2200	2	3.17	0.0034128000000000
TTTIV	38	20	2133	6	2500	2200	1	1.90	0.0020476800000000
TTTIV	38	20	2133	6	2500	2200	2	1.90	0.0020476800000000
TTTIV	38	24	2133	6	2500	2200	1	1.58	0.0017064000000000
TTTIV	38	24	2133	6	2500	2200	2	1.58	0.0017064000000000
TTTIV	38	28	2133	6	2500	2200	1	1.36	0.00146262857142857
TTTIV	38	28	2133	6	2500	2200	2	1.36	0.00146262857142857

Table 4: The list of configurations used in  $E_1$ . The *Fold sequence* column shows whether the configuration was a part of the training (T) or validation (V) set during each cross-validation fold stated in Figure 1. For example, the sequence TTTTV indicates configuration was part of the training set during the first four folds and a part of the validation set in the fifth fold.