BRISE: Energy-efficient Benchmark Reduction

Dmytro Pukhkaiev
Software Technology Group, TU Dresden, Germany
dmytro.pukhkaiev@tu-dresden.de

Sebastian Götz
Software Technology Group, TU Dresden, Germany
sebastian.goetz@acm.org

ABSTRACT

A considerable portion of research activities in computer science heavily relies on the process of benchmarking, e.g., to evaluate a hypothesis in an empirical study. The goal is to reveal how a set of independent variables (*factors*) influences one or more dependent variables. With a vast number of factors or a high amount of factors' values (*levels*), this process becomes time- and energy-consuming. Current approaches to lower the benchmarking effort suffer from two deficiencies: (1) they focus on reducing the number of factors and, hence, are inapplicable to experiments with only two factors, but a vast number of levels and (2) being adopted from, e.g., combinatorial optimization they are designed for a different search space structure and, thus, can be very wasteful. This paper provides an approach for benchmark reduction, based on adaptive instance selection and multiple linear regression. We evaluate our approach using four empirical studies, which investigate the effect made by dynamic voltage and frequency scaling in combination with dynamic concurrency throttling on the energy consumption of a computing system (parallel compression, sorting, and encryption algorithms as well as database query processing). Our findings show the effectiveness of the approach. We can save 78% of benchmarking effort, while the result's quality decreases only by 3 pp, due to using only a near-optimal configuration.

CCS CONCEPTS

• **General and reference → Empirical studies**; **Experimentation**; • **Mathematics of computing → Regression analysis**; • **Software and its engineering → Extra-functional properties**;

KEYWORDS

benchmarking, adaptive instance selection, non-functional properties, fractional factorial design, active learning

ACM Reference Format:
Dmytro Pukhkaiev and Sebastian Götz. 2018. BRISE: Energy-efficient Benchmark Reduction. In *GREENS'18: GREENS'18:IEEE/ACM 6th International Workshop on Green And Sustainable Software , May 27–26, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3194078.3194082

1 INTRODUCTION

Benchmarking is an inevitable and widely used part of empirical studies such as, e.g., energy-efficiency studies [8, 9, 20]. By benchmarking we understand an experimental examination of multiple factors' influence on a single or more variables under investigation. The main disadvantage of benchmark-driven studies is the high effort (in both time and energy) required for their completion. Assume, we encounter an experiment with *n factors*. Moreover, each factor comprises different values, i.e., *m levels*. Furthermore, while benchmarking, we cannot neglect the variance. Therefore, we need to repeat each measurement k times. Thus, the final number of runs can be computed as: $k \times m^n$ executions. Naturally, this number can be very high.

A general approach to address this problem is benchmark-runsize reduction, i.e., fractional factorial design [4]. It transforms the runsize by identifying a set of q unimportant factors among all n; thus, reducing the number of combinations to be benchmarked: $k \times m^{n-q}$. The reduction of levels per factor is not studied that extensively. A typical approach is to transform a multi-level factor into several *pseudofactors* (1 factor with n levels is transformed into k factors with m levels each, $\sum_k m = n$), which can be reduced later. However, such a reduction is coarse-grained (the finest granularity equals $2^2 = 4$, see Section 3.2 for more details) and, thus, inefficient.

Another way to tackle this problem is a utilization of a local search (LS) approach such as, e.g., hill climbing or metaheuristics (simulated annealing, tabu search, etc.). These approaches were initially developed for handling combinatorial optimization problems such as, e.g., scheduling, bin packing, knapsack problem, etc. Therefore, LS approaches by design presume search of an optimal solution in a large neighborhood, where the steps are relatively cheap (e.g., one needs only to calculate the score of the solution). On contrary, in benchmark reduction, steps can be very expensive: 1 step (benchmarking of a next configuration) costs approx. 30 seconds [20], comparing to approx. 1000 steps per second by LS approaches[1], i.e., 4 orders of magnitude higher then by a typical LS problem. Hence, LS approaches are too wasteful for our use case. Moreover, some of them, such as hill climbing suffer from the local optima.

In this paper, we introduce an iterative, adaptive and fine-grained benchmark reduction approach, which is suitable for energy measurements. The key idea is to start with very few measurements and to identify the necessity for performing further benchmarking based on this growing subset of measured configurations by building a polynomial regression model and predicting the missing results. As a running example and for our evaluation, we use a series of software energy-efficiency studies. In these case studies, an optimal configuration implies the most energy-efficient combination

[1]https://docs.optaplanner.org/7.5.0.Final/optaplanner-docs/html_single/index.html#scoreCalculationSpeed

of a CPU frequency (dynamic voltage and frequency scaling) and a thread number (dynamic concurrency throttling). Our findings show a high effectiveness of the approach, saving 78% of benchmarking effort (83.7 MJ instead of 374.3 MJ) while causing only a 3 percentage points (pp) deterioration of energy savings as a trade-off to using only a near-optimal configuration (12.15% of energy savings instead of 15.14%).

2 RESEARCH QUESTIONS

The aim of this paper is to show how to minimize the effort being spent on empirical energy-efficiency researches. Decreasing this effort, i.e., performing fewer experiments, has a trade-off w.r.t. the quality of the results, which we intend to maximize. Thus, our research objective is to develop an approach that minimizes the amount of benchmarking for factorial experiments whilst maximizing the quality of results. For this, we will answer the following research questions:

- RQ1 (Benchmark reduction). Is it possible to reduce the benchmarking effort required to find an optimal result and what is the effect of such a reduction on the quality of the obtained result?
- RQ2 (Genericity). Can the same benchmark reduction technique be equally applied to various types of algorithms or experiments?

3 CASE STUDIES

We use a study on the exploration of the effect made by dynamic voltage and frequency scaling in combination with dynamic concurrency throttling on the energy consumption of a computing system running four different types of algorithms (compression, database queries, sorting and encryption) [20] as a running example.

It was performed on a machine with two Intel Xeon E5-2690 processors, each with 8 physical cores and up to 16 threads (Hyper-Threading). All cores of one socket use the same frequency and voltage configuration. In our experiments we have used the same frequency for both sockets. The system uses an Intel 520 series SSD. The total AC real power consumption of the system was measured with a calibrated ZES Zimmer LMG450 power analyzer [24]. A detailed description of the test system is presented in [10]. Dynamic Voltage and Frequency Scaling (DVFS) allows using one of sixteen CPU frequencies (from 1.2 to 2.9 GHz except 1.5, 2.1 and 2.6 GHz with a turbo mode of 3.8 GHz). Thus, there are two changing variables: CPU frequency and the number of threads.

The following parameters further increase the amount of benchmarking: algorithms (6 compression, 21 DB queries, 4 encryption, 3 sorting), actions (de-/compression, de-/encryption), data types (6 data formats for compression, 4 for encryption, 3 for sorting) and repetitions (8 times by DB queries and 10 times by remaining). More information can be found in Section 5.

3.1 Full factorial design

Full factorial design is an experiment, which consists of two or more *factors*, i.e., variables that can influence the investigated characteristic, each having a number of *levels*, i.e., values the variable can have. The word *full* denotes that all possible combinations of levels for all the factors are to be benchmarked. The number of experiments may extend even more by multiplying the overall number of combinations by a number of repetitions for each configuration, e.g., to decrease variance. Therefore, we can identify the main drawback of full factorial design: significant effort on its conduction.

For our example full factorial design requires 567.296 test runs (16 frequencies $\times$ 32 numbers of threads $\times$ 10 repetitions $\times$ 6 algorithms $\times$ 6 data types $\times$ 2 de/compression = 368.640 compression runs + 86.016 queries + 56.320 encryption + 56.320 sorting), which, for an average execution time of 30 seconds, takes up to 197 days and uses more than 1.92 GJ of energy, i.e., a German private household's average energy demand for housing per 3.3 month [6]. Note, that these numbers are extrapolated based on average values, practical conduction of a full factorial design was infeasible.

3.2 Fractional factorial design

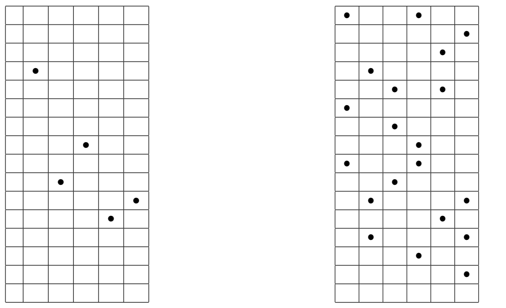
Fractional factorial design is an experiment, in which only a fraction of possible combinations is observed [3, 14, 15].

A common use case for both regular and non-regular design looks as follows: s^k, where s is the number of levels and k is the number of factors (e.g., 32 possible thread counts and 16 CPU frequencies). The idea is to decrease the number of factors, i.e., decrease the runsize up to s^{k-q}, where q = 1, 2, ... k-1. We can already see the deficiency of these approaches for the example we have discussed so far: with a small number of factors (2 factors) and a high number of levels (32 levels for one factor and 16 for another one) the runsize is considerably large ($32 \times 16 = 512$) and a removal of a factor makes an experiment completely irrelevant. Thus, we observe the need for a partial reduction of configurations, whilst preserving the number of factors.

In the context of a two-factor multi-level experiment the utilization of pseudofactors [4] is worth mentioning. Pseudofactors are used for asymmetric designs (when the number of levels varies for different factors). For example, one four-level factor can be substituted by two two-level factors. Then, one of these pseudofactors can be taken away by the reduction technique. In our example, we have a run size of $16 \times 32 = 512$, the utilization of pseudofactors offers four possibilities to represent the run size: a) single split $16 \times 16^2 = 16^3$, b) two times split $8^2 \times 8^2 \times 8^2 = 8^6$, c) three times split 4^{12} and d) four times split 2^{24}.

Intuitively, 24 factors may seem to be a decent choice to perform a reduction. However, this approach is very course-grained. For the experiment's form of 2^k, the finest level of granularity consists of 4 configurations (e.g., 2 CPU frequencies $\times$ 2 thread numbers). Coarse-grained subsets of configurations imply a lower probability of getting a result with a minimum effort, what is our primary goal. Moreover, the fact worth taking into consideration is that the described scenario is the best case. Unlike it, a considerable portion of designs cannot be transformed into a 2^k design, therefore, introducing even coarser granularity.

Summarizing the requirements for a two-factor multi-level experiment such as energy-efficiency studies, we come to the conclusion that a precise generic process of configuration selection needs to be 1) **iterative**, 2) **adaptive** and 3) **fine-grained**. Conventional methods of runsize reduction fail to satisfy these prerequisites and, thus, are not applicable. As the result, we need an unconventional fractional factorial design.



(a) 5 configurations **(b) 20 configurations**

Figure 1: Various numbers of picked configurations with help of Sobol sampling

4 CONCEPT AND IMPLEMENTATION

The main idea of our approach is to treat configurations as data and to use data mining techniques to identify important configurations. The idea behind runsize reduction is an *iterative* addition of configurations, their measurement and a subsequent evaluation of the necessity in further benchmarking by predicting the measurements for the remaining configurations. We utilize a subset of selected configurations as training and testing datasets for a polynomial regression model that, if considered appropriate, builds a decision function, i.e., a matrix containing a full runsize with actual values for already benchmarked configurations and predicted values for omitted ones. Afterwards, we select a *possibly-optimal* configuration based on the results provided by the decision function. Furthermore, we provide a series of checks to determine whether the possibly-optimal configuration qualifies as being near-optimal.

The data mining technique, which represents the aforementioned process, is called adaptive instance selection (sampling) [17]. The name of our approach is derived from it: Benchmark Reduction via adaptive Instance SElection (BRISE). It comprises a selection algorithm and an evaluation of correctness (stop function). In order to pick configurations we utilize the Sobol sequence [22]. It provides a uniform, quasi-random sequence in multidimensional space. Each newly picked configuration preserves uniformity of a sequence; thus, covering the search space and satisfying our requirements. Moreover, such a selection technique enables us to scale configuration selection to more dimensions.

Figure 1 represents different amounts of benchmarked configurations for a search space of 96 configurations (16 × 6) selected by BRISE utilizing Sobol sampling: 5, 20 configurations respectively.

After measuring every new configuration (10+ times), we build a polynomial multiple Ridge regression model (VI degree). CPU frequency and thread count are used as independent variables, while energy consumption as a dependent. Ridge regression is a simple regression model from both computational and interpretational points of view and, therefore, is widely used in predicting runtimes of various algorithms [13]. Prediction of an algorithm's energy consumption instead of runtime does not have fundamental differences (both are floating-point variables); thus, we have considered this regression method as a decent choice. We split the measured data between training and testing datasets. Then, we re-balance these datasets so that the size of the training set is between 10 and 70% of the measured data and the accuracy of the model is very high ($R^2 \geq 0.99$). By R^2 or accuracy we understand a coefficient of determination of a prediction:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{samples}-1} (y_i - \bar{y}_i)^2}, \tag{1}$$

where $\bar{y} = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} y_i$, $\hat{y}_i$ is a predicted value of the i-th sample, y_i is the true value of the respective sample and $n_{samples}$ is the overall number of samples[2].

If we cannot satisfy the aforementioned condition for current datasets, we decrease the minimum acceptable accuracy so that $R^2 \geq 0.98$ and re-balance the datasets again. Finally, we receive the regression model with its highest possible accuracy.

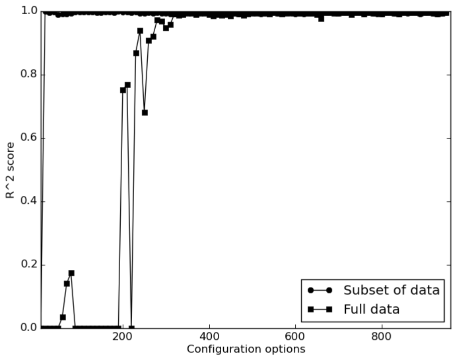
To illustrate the process of accuracy transformation, we discuss it for different experiments in the following. Figures 2a and 4 describe dependencies between the size of a measured subset and the accuracy of a regression model for different use cases: data compression and decompression, respectively. Figures 2b, 3a and 3b depict the dependencies for sorting algorithms, database queries and encryption algorithms, respectively. These figures show a comparison of the regression model's accuracy (circles) and its actual R^2 based on the full data (boxes).

Figures 2 and 3 show the most common cases: 69% and 28% of all experiments we conducted (for all algorithms, data types, actions, total 113), respectively. In both scenarios, we can see a stable high accuracy for a subset of configurations, while the actual accuracy changes from being insignificant to excellent. We can also observe a certain threshold at which the accuracy changes dramatically. Figure 4 shows a scenario (3% of experiments) for which even a full benchmarking process results in a weak regression model and, thus, BRISE is inapplicable. In such scenarios the line with circles goes down rapidly from the start. Therefore, we have determined a minimum acceptable accuracy of 0.85. Getting a lower R^2 implies an inapplicability of the benchmarking process in general and can be briefly identified; thus, significantly reducing effort savings.

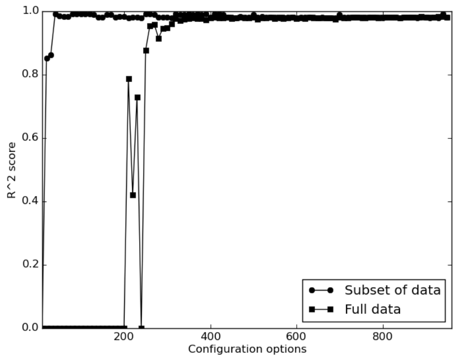
The experiments demonstrate how the selection algorithm without a stop condition works. A naive decision would have been to define a threshold for R^2 to be reached. However, a high accuracy can be deceitful as it is being checked only on a subset of configurations; therefore, it may differ for a full dataset as can be seen in Figures 2, 3 and 4. To address this issue, as a first step, we introduce a test of adequacy for the regression model, which implies the relative precision of the model. This test investigates if an experiment-specific rule is fulfilled. For instance, if the observed variable in an experiment is energy consumption, which is typically intended to be minimized, the experiment-specific rule is its non-negativeness. Otherwise, we consider the model irrelevant and start a new iteration by measuring the next configuration.

Next, we can start final checks, which require benchmarking and, thus, are performed only after the test of adequacy. First, we benchmark the identified possibly-optimal configuration to find its actual value. Secondly, we compare this actual value with the one of the naive choice. By this, we ensure the satisfaction of an

[2]http://scikit-learn.org/stable/modules/model_evaluation.html#r2-score

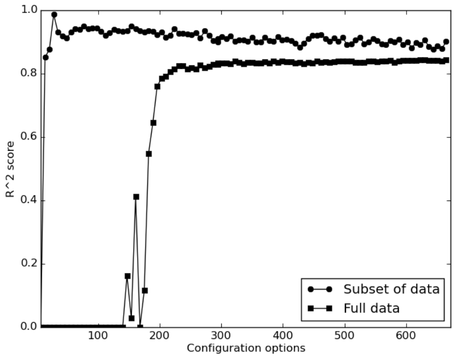


(a) **Pbzip2 compression of application data (representative for 63% of compression experiments)**

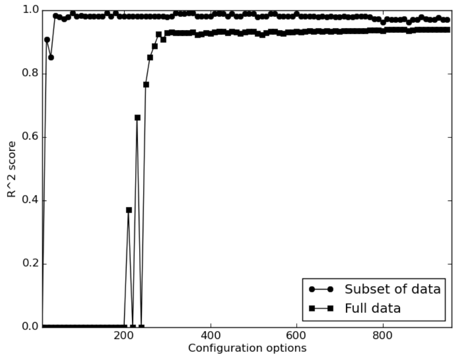


(b) **Radix sort of 750 millions of integers (representative for 83% of sorting experiments)**

Figure 2: R^2 score of a regression model from a number of test runs (best case)



(a) **TPC-H Query #6 on MonetDB (representative for 24% of database queries experiments)**



(b) **AES encryption of game data (representative for 9% of encryption experiments)**

Figure 3: R^2 score of a regression model from a number of test runs (acceptable case)

initial goal (e.g., energy savings for sweet-spot research). If the actual value is worse than that of the naive choice, the possibly-optimal configuration is added to the training/testing dataset and the process repeats. From this point, the configurations BRISE picks are not the same for each experiment, it *adapts* to it individually. Thus, the final set of configurations for even the same search space is different for each experiment.

Finally, if all the aforementioned conditions are satisfied, we compare all the measured configurations and pick the optimal among them. The reason for such a comparison is the possibility of slight impreciseness of the regression model. Assume, the possibly-optimal configuration has the best result based on its predicted value. Moreover, after the measurement it is better than the naive choice. However, it is possible that the actual value slightly differs from the predicted one and at the same time is worse than some other configuration among those already measured. In this case, we substitute the possibly-optimal configuration by the better one.

The afore-described approach enables us to answer the research questions RQ1 and RQ2:

(1) RQ1 (Benchmark reduction). After a certain threshold we receive a highly accurate regression model and, thus, the chance of predicting the optimal configuration is very high. Otherwise, we can determine that the benchmarking process is inapplicable in general and stop benchmarking early.
(2) RQ2 (Genericity). The approach is generic, i.e., applicable to different computational processes.

5 EVALUATION

We evaluate the approach by applying it to research focusing on the search for energy-efficient configurations, which comprises the investigation of the energy consumption for multiple algorithms [20]. Due to the space limitation we do not present the detailed results for all types of algorithms in this paper, but show only the overall

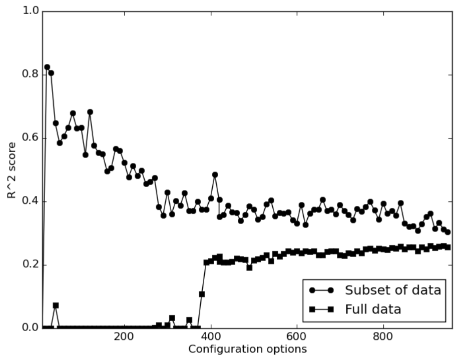
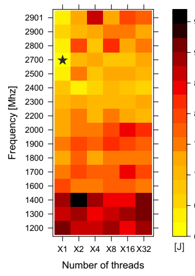
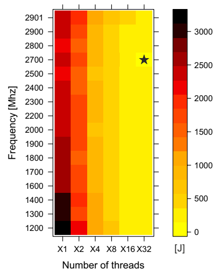


Figure 4: R^2 score of a regression model from a number of test runs (worst case). Pigz decompression of FLAC audio (representative for 1% of compression experiments)



(a) Energy consumption of pigz decompressing FLAC audio



(b) Energy consumption of a counting sort over 200 millions of integers

Figure 5: Heatmaps for exemplary processes (optimal configuration in terms of energy is marked with a star)

results for all algorithms (see Table 1). The full detailed results as well as the source code can be accessed online[3].

The compression algorithm case study comprises six algorithms: pbzip2[4], plzip[5], pigz[6] and three variations of NanoZip[7] (512 MB, 5 GB and 50 GB of memory); six types of data: application and game data, audio data in WAV and FLAC formats, XML text data extracted from Wikipedia: 100 MB and its extended version of 1 GB; two actions: compression and decompression; and ten repetitions to decrease the variance. Moreover, all possible configurations of the above described search space were repeated for all combinations of two factors: 16 CPU frequencies and 6 degrees of parallelism (thread numbers). Thus, we conducted: $16 \times 6 \times 10 \times 6 \times 6 \times 2 = 69.120$ test runs. The benchmarking took 28 days and 272,5 MJ of energy, excluding recomputations of broken results.

After applying BRISE to this case study, we were able to considerably reduce the amount of benchmarking: up to 78%, while preserving the optimal results in 5 cases out of 12. For those experiments, where BRISE picked a near-optimal configuration instead of the optimal one, we can observe only a slight decrease in energy savings (0.28 - 1.37 pp).

The compression case study contains one outlier: decompression of FLAC audio. This experiment is an example of an indistinguishable optimal configuration: the difference in energy consumption at different configurations is so small that it is overlapped by the variance (see Figure 5a). Hence, the regression model for such a case cannot accurately predict the energy consumption even with a high number of measured configurations (see Figure 4). Thus, we exclude this setting from the evaluation.

In summary, for the compression case study, by using near-optimal configurations instead of the optimal ones, we were able to save 9.08% of energy instead of 9.53% (i.e., 0.45 pp less). At the same time, we diminished the benchmarking effort by 77.6% in terms of energy and 78.2% w.r.t. time, respectively (see Table 1). In absolute numbers, such a reduction means that the benchmarking process required 211.6 MJ less energy and instead of 28 only took 7.8 days.

Table 1 provides our findings for all investigated types of algorithms. Here, `NFFD` implies the benchmarking effort made on the respective algorithm type with an NFFD, `BRISE` means the reduced effort for a corresponding type, `ES` - effort savings (percentage of the benchmarking reduction), `SF` - energy savings using the optimal configurations obtained by an NFFD, `SB` - savings using near-optimal configurations provided by BRISE and `TO` - the trade-off of using BRISE instead of an NFFD (i.e., `TO` = `SB` – `SF`)). For each type of algorithm the first row represents the information on energy while the second shows the time information. The `Total` row, shows aggregated results that are calculated as a sum for columns `NFFD` and `BRISE`, a weighted average for columns `SF`, `SB` and `TO` and for column `ES` the percentage is calculated from the columns `NFFD` and `BRISE`. The main reason for using the sum function for the physical entities such as days and Joules is to emphasize the absolute amount of effort necessary to perform the aforementioned benchmarking with and without BRISE. We use the weighted average based on the number of use cases (optimal configurations) because the studies have different sizes of output (see under the name of each study). For example, for compression algorithms the average energy reduction is 9.53×11 and for encryption algorithms it is only 23.87×4. Thus, each found (near-)optimal configuration has the same weight and the same impact on the overall results.

In our study on database queries [9], we assessed the energy consumption for 21 TPC-H queries, an industry standard benchmark. Our findings show that BRISE can provide decent effort savings for this case study: approximately 83% of energy and time savings for benchmarking, while diminishing the energy savings of the approach itself by 3.8 pp, from 16.7% to 12.9%.

Furthermore, we studied energy-efficiency of three parallel sorting algorithms (radix-, counting- and quicksort) for three different array sizes (200, 500 and 750 million of integers). Using BRISE, we decreased the benchmark effort by 77.6% and 78.1% of energy and time, respectively. Simultaneously, the resulting energy savings of the approach itself decreased by 4 pp to 10%.

For encryption algorithms, we investigated Advanced Encryption Standard (AES) [18] and 3-Data Encryption Standard (3DES) [1].

[3]https://github.com/dpukhkaiev/BRISE/tree/master/
[4]http://compression.ca/pbzip2
[5]http://www.nongnu.org/lzip/plzip.html
[6]http://zlib.net/pigz
[7]http://nanozip.net

	NFFD	BRISE	ES [%]	SF [%]	SB [%]	TO [pp]
Compression	272.5 MJ	60.9 MJ	77.6	9.53	9.08	-0.45
[11 out.]	27.99 days	7.8 days	78.2	-9.17	4.63	4.54
DBs	28.47 MJ	4.99 MJ	82.5	16.69	12.86	-3.83
[21 out].	2.77 days	0.47 days	82.9	2.18	2.02	-0.16
Sorting	31.4 MJ	6.99 MJ	77.6	13.97	9.94	-4.03
[5 out.]	3.16 days	0.69 days	78.1	-33.2	-40.2	-7.0
Encryption	41.9 MJ	10.8 MJ	74.3	23.87	19.61	-4.26
[4 out.]	4.45 days	1.09 days	75.3	-27.71	-21.55	6.16
Total	374.27 MJ	83.68 MJ	77.6	15.14	12.15	-2.99
	38.37 days	10.05 days	73.8	-8.09	-4.71	3.38

Table 1: Overall results. `NFFD` - benchmarking effort made with an NFFD, `BRISE` - reduced effort, `ES` - effort savings (percentage of the benchmarking reduction), `SF` - energy savings by an NFFD, `SB` - savings by BRISE and `TO` - the trade-off between BRISE and an NFFD (`TO` = `SB` − `SF`)), `[X out.]` - a quantity of optimal configurations.

We also assessed the energy-efficiency of hash functions (in form of hash trees for parallelization): SHA-2 (512-bit digest) [19] and Whirlpool [2]. We applied block ciphers (encryption and decryption processes) to application, game and text data. With BRISE, we reduced time- and energy effort by 74.3% and 75.3% respectively, while the energy savings decreased to 19.6% (4.3 pp less). However, the time loss improved to 21.5% (6.2 pp faster).

The overall weighted average of energy savings for all four case studies using optimal configurations was 15.14%. Using NFFD took 374.27 MJ of energy and 38.37 days. BRISE reduced the benchmarking effort by 77.6% of energy and 73.8% time, resulting in only 83.68 MJ and 10.05 days. The overall trade-off of using a near-optimal configuration instead of an optimal one totaled in 2.99 pp, i.e., implying energy savings of 12.15%. Our findings vividly demonstrate the effectiveness of the approach.

6 RELATED WORK

By this point, we have compared BRISE only with an NFFD, which it considerably outperforms in terms of the effort savings (runsize reduction). To make the evaluation extensive we discuss contemporary benchmarking reduction approaches and select appropriate ones for a comparison.

SPLConqueror is an approach for configuring software systems with help of performance-influence models [21]. These models represent influences of configurable parameters not only on the system but on other parameters as well. The approach aims at minimizing the sample size to obtain these models while maximizing their accuracy. The authors utilize step-wise linear regression to learn from a sample of configurations.

Metaheuristics for combinatorial optimization problems such as Hill Climbing (HC) [7], Simulated annealing (SA) [16] are also directly comparable with BRISE as they start from an empty set and continue exploring the search space to find an optimum; thus, reducing the number of configuration to be benchmarked.

BRISE can easily interchange its selection algorithm. To illustrate this statement, we swap Sobol sampling by Fedorov's exchange algorithm (FEA) [5]. It is a general fractional factorial design that focuses on runsizes with a big number of factors.

The group of approaches developed by Hutter et al. is also worth mentioning: SMAC [11], ParamILS [12]. They follow the same goal: solving algorithm configuration problems with minimal effort. However, there is a key difference that makes these approaches incomparable to BRISE. Namely, the termination criteria: both ParamILS and SMAC operate in either a predefined time budget or amount of configurations to be measured. Hence, our approaches are not directly comparable.

As can be seen from Table 2, which depicts the results, FEA shows its superiority for our use case. Energy savings are comparable: 11.88% by FEA versus 12.15% by Sobol sequence, while effort savings are significantly better: 93.2% versus 77.6% by FEA and Sobol sequence, respectively. FEA picks configurations starting from the edges of a search space. It starts from corners, then covers the edges and follows to the configurations in the middle. This works ideally when the optimal configuration is on the edge of a search space that is sometimes the case for energy consumption studies: at highest thread number. However, generally, it needs more measurements to create an accurate model. This statement is supported by Table 3, which contains a full factorial design for a single experiment (512 configurations). We can see that FEA scales slower with a growing number of levels even for a luckily picked study. Therefore, we recommend using the Sobol sequence.

Next, we compare BRISE to other approaches:

The approach provided by Siegmund et al. has a deficiency: its reliance on the accuracy as a quality measure. In Section 4 we pointed out the high accuracy based only on a subset of configurations can be misleading, subsequently leading to a weak result. We applied SPLConqueror to our use cases utilizing the set up the authors stated to be the most efficient: Placket-Burman design with 49 factors and 7 levels: PB(49,7). It showed surprisingly better results over more sensible set ups for our case such as: PB(2,16) or PB(24,2). The results from Table 2 support our claim on the unreliability of accuracy or mean error as a success metric. Despite a high accuracy,

	ES [%]	TS [%]	EE [%]	ET [%]
BRISE	12.15	-4.71	77.6	73.8
FEA	11.88	-8.69	93.2	94.1
SPLConqueror	6.8	-	56	-
HC(1)	6.94	-21.18	81.5	81.6
HC(3)	13.44	-8.31	59.2	59.0
HC(5)	14.24	-9.18	45.2	45.4
HC(7)	14.73	-7.06	33.3	33.4
SA(10%)	13.88	-6.81	42.46	45.44
SA(50%)	-4.98	-45.46	82.65	83.41
NFFD	15.14	-8.09	0	0

Table 2: Overall savings by the reduction approaches. ES and TS - the energy and time savings by a respective algorithm. EE and ET - the effort savings of energy and time. HC(X) - Hill climbing with X random starting points, SA(X%) - Simulated annealing with a temperature that cools down by X% comparing to the current one.

which caused SPLConqueror's stop condition, the resulting energy savings reach only 6.8% compared to 12.15% provided by BRISE. At the same time, the effort savings achieved by this approach were only 56%, which is significantly less than 78% of BRISE.

Table 2 also shows that HC with few starting points provides worse results compared to BRISE, but bigger savings of effort. After adding more starting points (e.g., $\geq$ 3) it outperforms BRISE in terms of the variable under investigation, but requires more effort (up to 66.7%).

The results for SA confirmed the statement we have made previously. It heavily depends on its parameters such as temperature and cooling factor. Table 2 depicts two variants of SA: with cooling rate of 10 and 50% comparing to the current temperature. This means that the temperature after an iteration equals to $0.9 * T_{old}$ or $0.5 * T_{old}$, respectively. The results show us that for a slower cooling (10%) SA provides the energy savings of 13.88% while saving 45.2 and 45.4% of energy and time, respectively. That is comparable to HC with 5 restarts, i.e., these results are better than the ones of BRISE in terms of energy savings, but demand more configurations to be measured to obtain such a quality. SA with a faster cooling, in turn, needs a comparable to BRISE amount of configurations to find a solution it believes to be the best: 83.41%. However, the resulting configurations were mostly incorrect, resulting in even higher energy consumptions of tuned applications in comparison with naive configurations of the respective applications.

However, Table 2 shows the results for a relatively small runsize: $6 \times 16 = 96$ configurations. Moreover, it was luckily designed as it contained an optimal configuration for each experiment. However, for a different use case, with different factors and unpredictable behavior there would be no possibility to pre-reduce the runsize.

To compare the approaches for large runsizes, for a single experiment (pigz compression of WAV audio), we have performed a costly full factorial design ($16 \times 32 = 512$ configurations, 6.4 hours of execution time, 3.5 MJ of energy) and applied BRISE, FEA, HC, SA and SPLConqueror to it. In order to check both the stability and effectiveness of the approaches we repeated the execution 10 times. Table 3 describes our findings.

	Energy Savings [%]	Effort Savings [%]
HC	2.55-5.48	53-98
HC(10)	5.48	53
SA(10%)	-10.31	80-92
SA(50%)	-25.05	96-99
SPLConqueror	6.82	86.7
FEA	11.13	83.6
BRISE	11.13	87.3-88.9
FULL:	11.92	0

Table 3: Energy and effort savings for a bigger dataset (512 configurations)

Here, `Energy savings` are those provided by the optimal or near-optimal configuration. The maximum is provided by the full factorial design: 11.92%. BRISE made the same prediction 10 times and saves 0.8 pp less: 11.13%; however, at the same time, it saves up to 89% of effort. HC, on the other hand, provides much more modest savings by the near-optimal configurations: the best case is a decrease of the consumption by 5.48%, while the worst case is considerably less energy-efficient: 9.37 pp less savings. The overall effort savings for ten executions of HC are only 53%. We have used the same cooling rates as for the smaller search space and it resulted in even worser energy savings. Both cooling rates of 10 and 50% picked configurations that are less energy-efficient than the respective naive configurations. At the same time they needed to measure 8-20% and 1-4% of configurations to find these solutions. SPL Conqueror, in turn, showed its stability returning the same answer for all runs, but still had weaker predicted optimum: 6.82% of energy savings with 86.7% of effort savings.

BRISE using both selection algorithms has shown stability of results (the same near-optimal configuration identified) and a better applicability for the cases with a considerable amount of configurations. Hence, we can draw the conclusion that for an accurately picked runsize (e.g., without local optima and with a global one) HC can outperform BRISE. However, for more complex scenarios, BRISE guarantees savings, while HC shows very unreliable results. Simulated annealing, in turn, needs to be carefully tuned for each search space size to provide the appropriate results, which is infeasible in practice.

7 THREATS TO VALIDITY

In this section we would like to discuss the limitations of the approach. It also contains hints in applying BRISE to other case studies.

The main limitation is a utilization of a linear regression as an experiment model. Regressions as well as many other machine learning algorithms are very sensitive to hyperparameters [23]. Therefore, the model we use in this paper is manually tuned for a two-factor (CPU frequency, thread count) experiment. The degree of the model makes an enormous impact on the quality of results. To tune the model we have tested the R^2 of the model on a sample already conducted factorial experiment depending on a selected degree starting from 2. For the small values (< 5) we have observed extremely low scores even with sufficiently big training set (70%). That means that the model cannot properly represent the underlying data and cannot be used in practice. Starting from the 5th degree we have observed high scores (0.9+). Too high degree can introduce overfitting, i.e., the model is tuned exactly to specific training data and cannot correctly predict a dependent variable's value for testing data. Hence, the chosen degree should be as small as possible, but with an appropriate score. In practice we recommend to skip the very first degree with high scores (5th in our example) as it may still be unstable. This motivates our selection of the 6th degree.

With the growth of the parameters amount tuning of the regression model becomes another optimization problem: hyperparameter tuning. It can also be solved with a BRISE-like approach, thus, introducing an endless recursive problem. Therefore, a regression-based approach can be applied only to experiments with a small (2-3) amount of factors. For handling more dimensions one needs a parameter-less approach such as usage of Gaussian process, etc.

Another limitation is derived from a distinctive feature of BRISE, its test of adequacy. It is currently domain-specific and should be either substituted with a domain-independent one or designed manually for each distinct domain.

8 CONCLUSION AND FUTURE WORK

Two-factor multi-level factorial experiments are a special case of factorial experiments that are not applicable to conventional approaches. In this paper, we reduce this gap by providing a generic approach for benchmark reduction. We showed its effectiveness using four case studies, whereof all seek for the most energy-efficient configuration for a specific type of algorithm. We were able to diminish the benchmark effort by 78%, while observing an overall decrease in energy savings by only 3 pp. The selection algorithm can be interchanged and, if picked luckily, it can provide even more savings: 93% of effort with 4 pp trade-off. We were able to apply BRISE to different computational processes and, by this, showed empirical support for its genericity.

Adoption of the proposed approach can considerably reduce effort on empirical energy-efficiency studies: several days instead of months. Moreover, it can be relatively easy tuned to different scenarios: studies of runtime, etc.

One of the main directions for the future work is substitution of a parameter-dependent regression model with a parameter-less model such as Gaussian process, etc. It can adapt BRISE to handle more dimensions. However, we also plan to investigate a trade-off between choosing the respective models and provide rules for picking an optimal model in terms of effort on its tuning and quality of the results. Another important task is to make BRISE domain-independent, thus, we are currently designing a generic stop condition.

ACKNOWLEDGMENTS

This work is supported (in part) by the German Research Foundation (DFG) within the Collaborative Research Center SFB 912–HAEC and the SAB funded project GUIDES.

REFERENCES

[1] William C. Barker and Elaine B. Barker. 2012. *SP 800-67 Rev. 1. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher.* Technical Report. Gaithersburg, MD, United States.
[2] PSLM Barreto, Vincent Rijmen, et al. 2000. The Whirlpool hashing function. In *First open NESSIE Workshop, Leuven, Belgium*, Vol. 13. 14.
[3] Yen-Sheng Chen and Ting-Yu Ku. 2015. Development of a Compact LTE Dual-Band Antenna Using Fractional Factorial Design. *Antennas and Wireless Propagation Letters, IEEE* 14 (2015), 1097–1100.
[4] A. Dean, M. Morris, J. Stufken, and D. Bingham. 2015. *Handbook of Design and Analysis of Experiments.* CRC Press.
[5] V. V. Fedorov. 1972. *Theory of optimal experiments.* New York: Academic Press.
[6] Federal Statistical Office Germany. 2015. *Economy and Use of Environmental Resources. Tables on Environmental-Economic Accounting. Part 2: Energy (Preliminary Report). Reporting period 2000 - 2013.* Technical Report. Wiesbaden, Germany.
[7] S. M Goldfeld, R. E Quandt, and H. F Trotter. 1966. Maximization by quadratic hill-climbing. *Econometrica: Journal of the Econometric Society* (1966), 541–551.
[8] Sebastian Götz, Thomas Ilsche, Jorge Cardoso, and Josef Spillner. 2014. Energy-Efficient Data Processing at Sweet Spot Frequencies. In *Proceedings of the 4th International Symposium on Cloud Computing, Trusted Computing and Secure Virtual Infrastructures.*
[9] Sebastian Götz, Thomas Ilsche, Jorge Cardoso, Josef Spillner, Thomas Kissinger, Uwe Aßmann, Wolfgang Lehner, Wolfgang E. Nagel, and Alexander Schill. 2014. Energy-Efficient Databases using Sweet Spot Frequencies. In *Proceedings of the International Workshop on Green Cloud Computing.*
[10] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop.* 896–904.
[11] F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers.* 507–523.
[12] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. 2009. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research* 36, 1 (2009), 267–306.
[13] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown. 2014. Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence* 206 (2014), 79 – 111.
[14] Jessica Jaynes, Xianting Ding, Hongquan Xu, Weng Kee Wong, and Chih-Ming Ho. 2013. Application of fractional factorial designs to study drug combinations. *Statistics in Medicine* 32, 2 (2013), 307–318.
[15] Kevin J. Kauffman, J. Robert Dorkin, Jung H. Yang, Michael W. Heartlein, Frank DeRosa, Faryal F. Mir, Owen S. Fenton, and Daniel G. Anderson. 2015. Optimization of Lipid Nanoparticle Formulations for mRNA Delivery in Vivo with Fractional Factorial and Definitive Screening Designs. *Nano Letters* 15, 11 (2015), 7300–7306. PMID: 26469188.
[16] AG Khachaturyan, SV Semenovskaya, and B Vainstein. 1979. A Statistical-Thermodynamic Approach to Determination of Structure Amplitude Phases. *Sov. Phys. Crystallogr* 24 (1979), 519–524.
[17] Huan Liu and Hiroshi Motoda. 2002. On Issues of Instance Selection. *Data Mining and Knowledge Discovery* 6, 2 (2002), 115–130.
[18] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. 2009. *Advanced Encryption Standard.* Alpha Press.
[19] National Institute of Standards and Technology. 2015. *FIPS PUB 180-4: Secure Hash Standard (SHS).* National Institute of Standards and Technology, Gaithersburg, MD, USA.
[20] Dmytro Pukhkaiev. 2016. *Energy-efficient Benchmarking for Energy-efficient Software.* Master's thesis. Technische Universität Dresden.
[21] Norbert Siegmund, Alexander Grebhahn, Sven Apel, and Christian Kästner. 2015. Performance-influence Models for Highly Configurable Systems. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015).* ACM, New York, NY, USA, 284–294.
[22] Il'ya Meerovich Sobol'. 1967. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki* 7, 4 (1967), 784–802.
[23] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. 2016. Bayesian Optimization with Robust Bayesian Neural Networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4134–4142.
[24] ZES ZIMMER Electronic Systems 2009. *4 Channel Power Meter LMG450, User manual.* ZES ZIMMER Electronic Systems.