Optimizing Scientific Workflow: A Case Study of Mining and Tuning TCP Signatures Identification D

Huy Tu[a,*], George Papadimitriou[b], Mariam Kiran[c] and Tim Menzies[a,*]

[a]North Carolina State University, Raleigh, NC, USA
[b]University of Southern California, Los Angeles, CA, USA
[c]Lawrence Berkeley National Laboratory Energy Sciences Network, Berkeley, CA, USA

ARTICLE INFO

Keywords:
Scientific Workflow
TCP Signatures
Anomalies Detection
Hyper-Parameter Tuning
Sequential Optimization

ABSTRACT

Modern scientific workflows are often operated and maintained on distributed, dynamic, and high-performance cloud infrastructure. Any faults and anomalies mean loss of money and time (from running the current workflow and having to redo it). Such faults can forestall scientists from their attention and progress on their research endeavors. To address this issue, we developed X-FLASH, a scientific workflow anomalies detection, a data mining tool to automatically detect and accurately classify the anomolous TCP packets. Such information can guide the adaptation of application and infrastructure to reduce the likelihood of anomalies.
X-FLASH consists of (a) gradient boosting, XGBOOST; and (b) sequential optimization (to learn optimal settings for the model), FLASH.

1. Introduction

Scientific workflows have emerged as a flexible representation to declaratively express complex applications with data and control dependencies. Advanced virtualization technologies now enable scientific workflow to be executed in a reliable and scalable high-performance computing infrastructure. Not only are applications growing in scale and complexity, the distributed and high-performance computing infrastructure required to support science applications is increasingly diverse in scale and complexity. Examples of these infrastructures include the Department of Defense Leadership Computing Facilities; Open Science Grid [52]; XSEDE [1], cloud infrastructures (CloudLab[2]; Exogeni [13]) and national and regional network transit providers like ESnet[3] and Internet2[4].

This whole structure is threatened by undetected failures across the network. Such failures and anomalies at all levels of the system (hardware infrastructure, system software, middleware, application and workflows) make detecting, analyzing and acting on anomalous events challenging. Such anomalies are damaging to reliability of the cloud infrastructure, but more importantly, to the loss of money and resource from the clients to run the workflows. Particularly, in computational science, it adds extra overheads to scientists that forestall or even completely obstruct their research endeavors or scientific breakthrough. At the time of this writing, these problems are particular acute (the COVID-19 pandemic has stretched the resources used to monitor, maintain and repair this infrastructure).

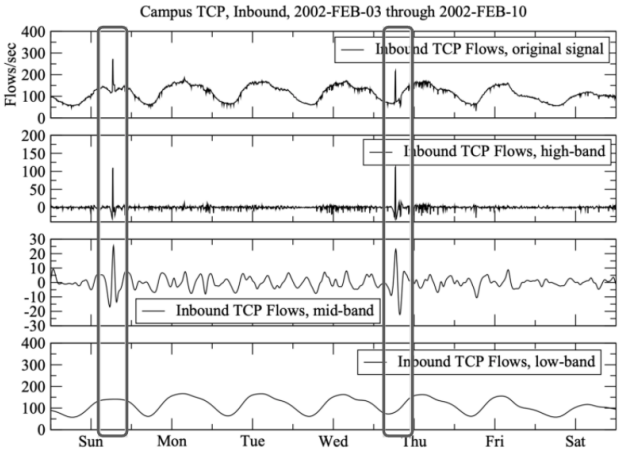


Figure 1: Example of packet flow signals for a one week period highlighting two anomalies plus high/mid/low decompositions.

This work seeks ways to find and detect those workflow anomalies, thus alerting the users and maintainers of these systems that there is a problem. We explore one specific kind of anomalies: faulty Transmission Control Protocol (TCP) file transfers similarly to those shown in grey boxes of Figure 1. Such anomalies represent a troubling class of problems, according to our subject matter experts [49].

Due to the mission-critical nature of the network's anomalies, the project's goal is to seek accurate and automatic anomalies detection method in order to proactively take appropriate and automatic action. This paper uses machine learning to automatically build those anomaly detectors. Much research has shown that better learning can be achieved by tuning the control parameters of those tools [5, 6, 41, 7, 1]. However, tuning has its own limitations:

- A standard hyperparamter optimization runs needs N eval-

*Corresponding author
huyqtu7@gmail.com (H. Tu); georgpap@isi.edu (G. Papadimitriou); mkiran@es.net (M. Kiran); timm@ieee.org (T. Menzies)
www.kentu.us (H. Tu); www.menzies.us (T. Menzies)
ORCID(s):

[1]"Extreme Science and Engineering Discovery Environment", http://www.xsede.org.
[2]"CloudLab", http://cloudlab.us.
[3]Lawrence Berkeley National Laboratory, ESnet: http://www.es.net.
[4]"Internet2", https://www.internet2.edu/

Table 1
Hyperparameter tuning options explored in this paper. Options in learners from recent SE papers on hyperparameter optimization [29, 25, 8, 6] then consulting the documentation of a widely-used data mining library (Scikit-learn [51]). Randint, randuniform and randchoice are all random functions to choose either integer, float, or a choice among the parameter ranges.

DATA PRE-PROCESSORS:

- StandardScaler
- MinMaxScaler
- KernelCenterer
- Normalizer(norm=a): a = randchoice(['l1', 'l2','max'])
- MaxAbsScaler
- RobustScaler(quantile_range=(a, b)): a, b = randint(0,50), randint(51,100)
- QuantileTransformer(n_quantiles=a, output_distribution=c, subsample=b). a, b = randint(100, 1000), randint(1000, 1e5); c=randchoice(['normal','uniform']).
- Binarizer(threshold=a): a = randuniform(0,100)
- SMOTE(a=n_neighbors, b=n_synthetics, c=Minkowski_exponent). a,b = randit(1,20),randchoice(50,100,200,400). c = randuniform(0.1,5)

LEARNERS:

- DecisionTreeClassifier(criterion=b, splitter=c, min_samples_split=a). a, b, c= randuniform(0.0,1.0), randchoice(['gini','entropy']), randchoice(['best','random']).
- RandomForestClassifier(n_estimators=a,criterion=b, min_samples_split=c). a,b,c = randint(50, 150), randchoice(['gini', 'entropy']), randuniform(0.0, 1.0)
- MultinomialNB(alpha=a): a = randuniform(0.0,0.1)
- KNeighborsClassifier(n_neighbors=a, weights=b, p=d, metric=c). a, b,c = randint(2, 25), randchoice(['uniform', 'distance']), randchoice(['minkowski','chebyshev']). If c=='minkowski': d= randint(1,15) else: d=2

uations which can increase learner times by many orders of magnitude. Based on the default parameters suggested by Goldberg [30], Storn [59], and using some empirical results from Fu et al. [1], we can compute the number of times a hyperparameter optimizer would have to call a data miner.

- Assuming that, $R = 25$ times[5] we are analyzing D data sets with say L learners and H hyperparameters with each H has continuous or discrete V values to pick, then with these settings, hyperparameter optimization with grid search needs to repeat the experiment millions of times ($RxLxD\ xHxV$). For instance, Table 1 is a sample of the options to explore in this space for L, H, and V, which approximately reach billion of choices (assuming each numeric range divides into, say, 10 options).

Recent work from the Software Engineering literature suggests that there are much better ways to perform hyperparameter optimization methods at very little computational cost [5, 6, 41, 7, 1]:

- In research on the DODGE(ϵ) algorithm [5], Argawal et al. reports that with only 30 evaluations by navigating through the output (result) space of the sample of the learners, the preprocessors, and their corresponding parameters choices to outperform traditional evolutionary approaches.
- In research on the FLASH algorithm [47], Nair et al. reports that sequential model optimization method can be utilized for software configuration optimization (and possibly hyperparameter tuning).

To date, these two approaches have not been compared against each other in the space of hyperparameter tuning. More importantly, the state-of-the-art anomalies detection through TCP file transfer results are based on off-the-shelf machine learning model. Accordingly, this paper asks it is better to optimize the data miners with FLASH or navigate thoroughly the whole data mining pipeline with DODGE(ϵ) for faulty Transmission Control Protocol.

Specifically, we ask:

- **RQ1: Does tuning improve the performance of anomalies detection?** We will show that relative improvement can be achieved up to 10% on accuracy, 10% on precision, and 40% on recall. Tuning the XGBoost model, an ensemble model (boosting), with FLASH helps to achieve the highest performance. DODGE(ϵ) shall not recommended for this task.
- **RQ2: Is tuning anomalies detection impractically slow?** In a result consistent with early work by Nair and Agrawal et al. [5, 47] we achieved statistically significant improvement in the performance scores for our data miners in less

[5] Why 25? In a 5x5 cross-val experiment, the data set order is randomized five times. Each time, the data is divided into five bins. Then, for each bin, that bin becomes a test set of a model learned from the other bins

than 50 evaluations. Accordingly, we say that hyperparameter optimization need not be impractically slow.

- **RQ3: Does tuning change conclusions about what factors are most important in anomalies detection?** Our results show that any such "most important feature" list is altered by the optimization process described here. Also, after optimization, these lists tend to be very data set specific. Hence, we advise against advocating such lists.

From the empirical results of this study, we shall design a scientific workflow anomalies detection, called X-FLASH, with (1) an ensemble model, XGBoost; and (2) FLASH as a sequential optimization (to learn the optimal settings for the model).

The rest of this paper is structured as follows. Background work and this study case are discussed in the next two sections. §4 & §5 describes our empirical methodology and experimental design. This is followed by the details of the experiment used to answer our research questions in §6. Further discussion and threats to validity future work from this research are explored in §7. Finally, the conclusions of this work are given in §8.

2. Background

2.1. Why Studying Scientific Workflow?

Modern computational and data science often involve processing and analyzing vast amounts of data through large scale simulations of underlying science phenomena. With advantage in flexible representation to express express complex applications with data and control dependencies, scientific workflows have become prevalent centerpiece for such data-intensive science.

Today's high-performance networked systems are essential to people's daily lives, from mobile connectivity to intercloud communications, from checking emails to running complex scientific workflows on distributed cloud infrastructures. According to International Data Corporation and CISCO Internet Report in 2020 [2], 25 percent of global IT leaders surveyed say network automation will have the most impact on networking over the next five years. Figure 2 summarized the trends of growth for network needs from Cisco Annual Internet Reports from 2014 [53] and 2018 [58, 2] in 4 attributes including number of users (in billions), number of connected devices to the network (in billions), number of workloads per physical server, and Data Center (DC) Traffic (in zetabytes). Most growths are double in the span of 8 years, but the amount of data center traffic is approximately 5.7 times more. As the scientific workflow and the infrastructure support them keep increasing both in resource demands and complexity, there is an urgent need to guarantee the network reliable, high throughput connectivity, secure, and 99.9% available. However, there are bounds to be anomalies in such large scale systems and applications. Therefore, it is essential to identify and understand these anomalies early to allow the network administrator to reduce the likelihood of the anomalies and response time to forthcoming incidents.

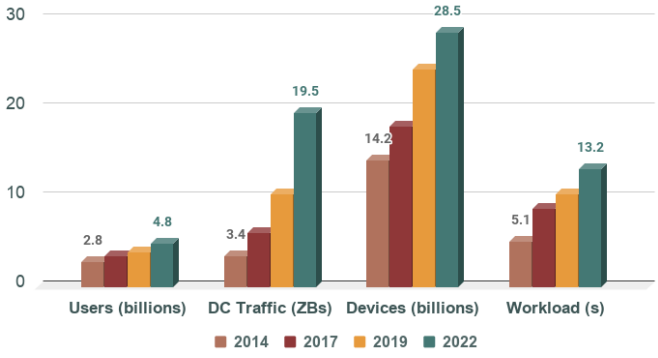


Figure 2: Network trends for users, data center traffic, connected devices to the network, and workloads per server from year 2014 to speculated future in 2022 ([53, 58, 2]).

In the particular case of scientific workflows, there are extra challenges. Unlike commercial systems (like Amazon Web Services) the teams maintaining this software do not work 24/7 on these systems. Rather, they split their time between system maintenance and all the other tasks they need to complete their own research. Hence, when a node in a scientific workflow crashes, it is highly possible that there would be some delay in finding the system's supervisor and requesting an appropriate fix.

For all the above reasons, we seek better ways to offer early warnings about these workflows. Such anomalies can be damaging to the scientific research community including:

- Severe real-life implications if contaminated results are transitions into practice;
- Staining the reputation of the researchers and institutions involved.

By proactively reducing the possibilities of having anomalies, we can reduce the mentioned risks and damages. Moreover, these shall improve scientific productivity. Such productivity have facilitated breakthroughs in several domains such as astronomy, physics, climate science, earthquake science, biology, among many others [62].

2.2. Anomalies Detection in Scientific Workflow

Various techniques could be used to anticipate workflow anomalies. It is typical to ensure reliability and quality of large-scale scientific workflows execution. This can seem very trivial regarding the scale or the frequency of the anomalies. However, these scientific workflows can take days to weeks to complete because of it's complexity comprising of dozens steps including data acquisition/transformation/preprocessing and model simulating/computing. Therefore, such anomaly can be detrimental to both cloud infrastructure and also the clients/scientific research groups. Domain knowledge could be applied, e.g., "Execution E_i has failed if it takes longer than t seconds". However this approach is brittle and non-portable between applications and resource types.

In 2013, Samak et al. [56] employed Naive Bayes classifier to predict the failure probability of tasks for scientific

workflows on the cloud using task performance data. They found that in some cases, a job destined for failure can potentially be executed successfully on a different resource. Others [12] have compared logistic regression, artificial neural nets (ANN), Random Forest (RF) and Naive Bayes for failure prediction of workflow tasks in the cloud and concluded that the Naive Bayes' approach provided the maximum accuracy. In Buneci et al.'s work [16], the authors have used a k-nearest neighbors (k-NN) classifier to classify workflow tasks into "Expected" and "Unexpected" categories using feature vectors constructed from temporal signatures of task performance data. Recently, Herath et al. [24] developed RAMP which is based on using adaptive uncertainty function to dynamically adjustest to avoid repetitive alarms while incorporating user feedback on repeated anomaly detection. These and several existing work [42, 27, 54, 24, 55] in end-to-end monitoring of workflow applications and systems are essential building blocks to detect such problems, current techniques for anomaly detection are often based on thresholds and simple statistics (e.g. moving averages) (Jinka et al. [37]) that has several limitations: (1) fail to understand longitudinal patterns, (2) miss anomaly detection, (3) seldom be used for identifying the root cause of the anomalies, and (4) limited to solely binary classification work.

Hence, it is not really applicable to our problem when we trying to understand different types of anomalies. Moreover, we believe it is more applicable to employ multivariate technique in order to (1) capture the interaction and relationship between features (as Deelman et. al [23] recommended); and also (2) detect the right type of anomalies in large-scale workflow executions on complex system.

2.3. Models Optimization

All previous anomalies detection work in scientific workflow lack (1) model optimization and (2) a tuning study. Specifically, this paper's work is based on the study done by Deelman et al. [49] that applied solely off-the-shelf RF to study faulty TCP file transfers in scientific workflow. It is essential for this study to develop such anomaly detection with tuning as the backbone.

Many previous studies have advised that using data miners without parameter optimizer is not recommended because: (1) such optimization can dramatically improve performance scores; and (2) any conclusions from unoptimized data miner can be changed by new results from the tuned algorith,. For example, Agrawal et al. [9] showed how optimizers can improve recall dramatically by more than 40%. Moreover, Fu et al. [25] showed how optimized data miners can generate different features importance for software defect prediction task. Hence, it is necessary to use data miner using/used by optimizers.

Configuration in the analysis pipeline have numerous problems in their nature:

- *A daunting number of options*: Table 1 presented billion of tuning options. Agrawal et al. [5] reported billion of options in both data pre-processors, data miners, and parameter settings in both processors and leaners. Meta heuristic methods such as Genetic Algorithm would need 10,000 evaluations (100 generations with 100 candidates per generation) or more.
- *Slow convergence*: Fu et al.'s replication work [26] of Tantithamthavorn et al. [61] experiment require 109 days of CPU. Worse case, decades of CPU time were needed by Treude et al. [64] to achieve a 12% improvement over the default settings.
- *Poorly chosen default configuration*: In ASE 2019's Keynote speech, Zhou [70] remarked that 30% of errors in cloud environments are due to configuration errors. Jamshidi et al. [36] reported for text mining applications on Apache Storm, the throughput achieved using the worst configuration is 480 times slower than the throughput achieved by the best configuration.

Solving these configurations is not limited to software systems and hyperparameter optimization in machine learning but also for cloud computing and software security. In cloud computing, different analytic jobs have diverse behaviors and resource requirements, choosing the correct combination of virtual machine type and cloud environment can be critical to optimize the performance of a system while minimizing cost [11, 66, 72, 20, 69]. In security of cloud computing, such problems of how to maximize conversions on landing pages or click-through rates on search-engine result pages [67, 71, 34] has gathered interest.

3. Faulty TCP Case Study

3.1. Overview

In this work, we analyze anomalous network transfers by utilizing data collected using TCP statistics (Tstat) [63], which is a tool to collect TCP traces for transfers. How TCP works can be demonstrated in Figure 3 as follows: the server sends a packet to a client (with the load in bytes), when the client receives the packet, it sends an "acknowledgment" (ACK) signal back. The total time from sending the package to receiving the ACK is recorded as the round trip time (RTT). Time windows is a measure that used by the TCP protocol to allow servers to wait for the ACK, before deciding to resend the packet again. The window size grows over

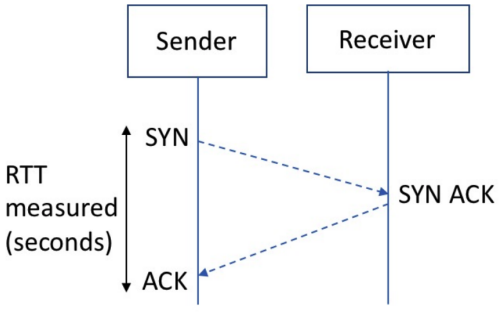


Figure 3: Round Trip Time (RTT) measured in the three way communication in TCP.

Table 2
Top 20% important TCP Statistics features collected by the Tstat tool identified by the state-of-the-art model [49] across the data. (S2C: Server to Client, C2S: Client to Server)

Attribute	Types	Description
c_bytes_all	C2S	Number of bytes transmitted in the payload, including retransmissions
c_pkts_retx	C2S	Number of retransmitted segments
c_bytes_retx	C2S	Number of retransmitted bytes
s_ack_cnt_p	S2C	Number of segments with ACK field set to 1 and no data
durat	—	Flow duration since first packet to last packet
c/s_first	Both	Client/Server first segment with payload since the first flow segment
c/s_last	Both	Client/Server last segment with payload since the first flow segment
c/s_first_ack	Both	Client/Server first ACK segment (without SYN) since the first flow segment
c/s_rtt_avg	Both	Average round trip time (RTT) computed measuring the time elapsed between the data segment and the corresponding ACK
c/s_rtt_min	Both	Minimum RTT observed during connection lifetime
c/s_rtt_max	Both	Maximum RTT observed during connection lifetime
s_rtt_std	S2C	Standard deviation of the RTT
c_cwin_max	C2S	Maximum in-flight-size computed as the difference between the largest sequence number so far, and the corresponding last ACK message on the reverse path. It is an estimate of the congestion window
c_pkts_reor	C2S	Number of packet reordering observed
c_pkts_dup	C2S	Number of network duplicates observed
c_pkts_unk	C2S	Number of segments not in sequence or duplicate which are not classified as specific events
s_sack_cnt	S2C	Number of SACK messages sent
s_win_max	S2C	Maximum segment size observed
c_pkts_push	C2S	Number of push separated messages from Client to Server
c/s_last_handshakeT	Both	For Transport Layer Security (TLS) flows, time of Client/Server last packet seen before first Application Data (relative)
c_appdataT	C2S	For TLS flows, time between the Client first Application Data message and the first flow segment

time respectively to the wait time to allows the protocol to adjust its waiting time. This affects the total recorded RTT while building up retransmissions and overflowing flows in both TCP directions.

The TCP protocol is designed to ensure all packets are delivered reliably. It does this by tracking various packet information, for example window timeouts, packet numbers, RTT and more, to calculate if loss has occurred. If yes, it triggers the host to resend the packets again. Collectively, Tstat traces contain about 133 variables per packet on both server and client sides. These features are listed in details on Tstat's documentation [6]. Table 2 reported the top 20% features ranked by their importance shared across the data by applying the state-of-the-art work by Papadimitriou et al. [49]. It is mission-critical to understand what attributes are essential to the model to make decisions on classifying the right type of anomalies. The proposed X-FLASH solution include tuning the solution learner, the attributes that are deemed important before tuning and after tuning change significantly in our experiment. This indicates that previous anomalies detection work reported misleading key features to the system managers or scientists which can cost them extra time and effort to debug. Accordingly, our proposed solution does inform the right key features to the scientists and network maintainers in order to take appropriate action and prevent future network anomalies.

[6] http://tstat.polito.it/

3.2. Variants

TCP provides reliable and error-checked delivery of a data stream between senders and receivers. Research efforts have been focusing on TCP extensions as variants to allow improvement of various network anomalies and enable congestion control. Jacobson et al. [35] established implementations of the modern TCP. Since the seminal work of Jacobson et al., some TCP variants are introduced to prioritize throughput over loss prevention. In this paper scope, we specifically focus on four of these variants:

1. TCP Cubic: the current default TCP algorithm in Linux [31]. Cubic aggressively increases the window size to allow throughput utilization to rise quickly, and then slowly as it reaches the saturation point. Window growth is independent of the RTT recorded, allowing Cubic to achieve equitable allocations between multiple flows on a link.

2. TCP Reno: is known as the dominant congestion control algorithm in Internet applications [48] which incorporate fast retransmit and fast recovery. The principle is cutting the window size by half when loss is detected. If multiple ACKs are received, Reno performs fast retransmits, skipping the slow start phase by halving the congestion window.

3. TCP Hamilton: aggressively reduce congestion window whenever loss occurs, allowing it to recover quickly. New

flows are seen to converge faster to optimal throughput under Hamilton than other algorithms [44, 39].

4. TCP BBR: attempts to create a balance between fast retransmits and fairness among flows on the same link. BBR reacts on actual congestion and network models of the available bandwidth. Google introduced BBR to improve download speeds on internal networks [17]. It calculates RTT and bottleneck capacity and uses this to estimate its delivery rate. The values of RTT and bottleneck capacity are independently managed and BBR attempts to stay within this range.

3.3. Types of anomalies

Network anomalies can create bogus traffic on the link causing network congestion and utilization in a router. Other than loss, there is significant effort to recognize congestion conditions such as overflowing buffers [4, 50]. However, there are commonly occurring network anomalies which can seriously impact user experience and also affecting the clients' work negatively as mentioned in §2.1. Described by [28, 10, 43] there are three common network anomalies:

- Packet Loss happens when one or more packets fail to reach their destination. These could either be caused by errors in transmission or too much congestion on link, causing routers to randomly drop packets.
- Packet Duplication happens when the sender re- transmits packets, thinking that the previous packets have not reached their destination. This can be commonly observed when packet losses happen and retransmits increase.
- Packet Reordering happens when arrival order of packets or sequence number is completely out-of- order. In the case of real-time media streaming application, it is particularly relevant to show network instability.

4. Software Configuration Optimization

The case was made above that (a) anomalies detection in scientific workflow is mission-critical, (b) previous studies lack optimization for their analytics pipelinem, and (c) tuning is a daunting task that requires careful attention per domain. Therefore, X-FLASH is designed to include tuning in the data mining pipeline for anomalies detection in scientific workflow. This section described how FLASH and DODGE(ϵ) can tune the data mining pipeline for a better scientific workflow anomalies detection.

4.1. Core Problem

The problem can be described starting with a configurable data miner has a set X of configurations $x \in X$. Let x_i represent the ith configuration of a data miner method. $x_{i,j}$ represent the jth configuration option of the configuration x_i. In general, $x_{i,j}$ indicates either an (i) integer variable or a (ii) Boolean variable. The configuration space (X) represents all the valid configurations of a data miner tool. The configurations are also referred to as independent variables (x_i) where $1 \leq i \leq |X|$, has one (single-objective) corresponding performance measures $y_{i,k} \in Y$ indicating the $1 \leq kth \leq m$ objective (dependent variable). In our setting, the cost of optimization technique is the total number of iteration required to find the best configuration settings.

We consider the problem of finding a good configuration, x^*, such that $f(x^*)$ is less than other configurations in X. Our objective is to find x^* while minimizing the number of iterations/measurements.

4.2. Overview

The heart of this problem is how do we optimize the analytical results (performance) with the knowledge at hand while minimizing the iterations (time) for the model to converge? In software engineering literatures, the solution can be seen in three schools of thought including evolutionary optimization, Sequential Model-Based optimization, and ϵ-dominance.

4.2.1. Evolutionary Optimization

Evolutionary algorithms (EAs) have been used as black-box optimization for improving current results [32] by mutating the existing configurations. *Differential evolution* (DE) [59] is a popular candidate in this approach. The premise of DE is that the best way to mutate the existing tunings is to extrapolate between current solutions. Three solutions a, b, c are selected at random. For each tuning parameter k, at some probability cr, we replace the old tuning x_k with y_k where $y_k = a_k + f \times (b_k - c_k)$ where f is a parameter controlling differential weight. The main loop of DE runs over the population of size np, replacing old items with new candidates (if new candidate is better). This means that, as the loop progresses, the population is full of increasingly more valuable solutions (which, in turn, helps extrapolation).

DE is similar to Genetic Algorithm (GA) except that the candidate solutions are not considered as binary strings (chromosome) but (usually) as real vectors. One key aspect of DE is that the mutation step size (see step 1 for the mutation) is dynamic, that is, it adapts to the configuration of your population and will tend to zero when it converges. This makes DE less vulnerable to genetic drift than GA. However, both of this methods in the this evolutionary approach suffered similar following limitations:

- *Cost Inefficient*: Genetic algorithms (in SE) often start with a population of $N = 10^2$ individuals, run for possible of $G = 10^2$ generations where useful variations of individuals in generation i are used to seed generation $i + 1$, evaluate all N individuals in all generations.
- *Slow Convergence*: the performance delta across these generations may be very slow and take a long time to stabilize (similar to GA being stuck in local-minima).
- *Poor Performance*: Agrawal et al. [5] GA and DE were shown to be inferior to DODGE(ϵ) (state of the art ϵ-dominance method for software engineering research).

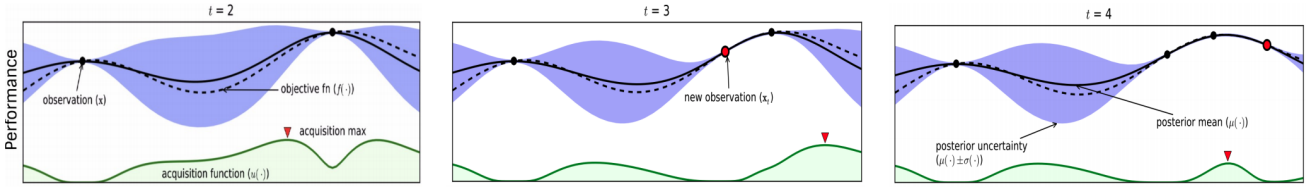


Figure 4: An example of Sequential Model-based method's working process from [15]. The figures show a Gaussian process model (GPM) of an objective function over four iterations of sampled values. Green shaded plots represent acquisition function. The value of the acquisition function is high where the GPM predicts larger objective and where the prediction uncertainty (confidence) is high such points (configurations in our case) is sampled first. Note that the area on the far left is never sampled even when it has high uncertainty (low confidence) associated.

For those reasons, research in this area in the last decade has explored non-EA methods for software configuration.

4.2.2. Sequential Model-Based optimization

FLASH is variant of Sequential Model-based Optimization (SMBO) which's core concept is "given what we know about the problem, what should we do next?". To illustrate this, consider Figure 4 of SMBO. The bold black line represents the actual performance function (f, which is unknown in our setting) and the dotted black line represents the estimated objective function (in the language of SMBO, this is the *prior*). The optimization starts with two points (t=2). At each iteration, the acquisition function is maximized to determine A model is built on the points and these evaluated measurements as the prior belief. This model can then learn where to sample next and find extremes of an unknown objectives. A posterior is then defined and captured as our updated belief in the objective function or surrogate model. The purple regions represent the configuration or uncertainty of estimation in a region—the thicker that region, the higher the uncertainty. The green line in that figure represents the acquisition function. The acquisition function is a user-defined strategy, which takes into account the estimated performance measures (mean and variance) associated with each configuration. The chosen sample (or configuration) maximizes the acquisition function ($argmax$). This process terminates when a predefined stopping condition is reached which is related to the budget associated with the optimization process.

Gaussian process models are often the surrogate model of choice in the literature. Yet, building GPM can be very challenging due to (1) GPM can be very fragile and sensitive to the parameters settings and (2) GPM do not scale to high dimensional data as well as large data set (i.e. large option space). Therefore, Nair et al. [47] proposed FLASH to improve over traditional SMBO by:

- FLASH models each objective as a separate Classification and Regression Tree (CART) model. Nair et al. report that the CART algorithm can scale much better than other model constructors (e.g. Gaussian Process Models).
- FLASH replaces the actual evaluation of all combinations of parameters(which can be a very slow process) with a *surrogate evaluation*, where the CART decision trees are used to guess the objective scores (which is a very fast process). Such guesses may be inaccurate but, as shown by Nair et al. [47], such guesses can rank guesses in (approximately) the same order as that generated by other, much slower, methods [46].

FLASH can be executed as follow:

Step 1 **Initial Sampling:** A predefined of configurations from the option space is sampled and evaluated. The evaluated configurations are removed from the unevaluated pool.

Step 2 **Surrogate Modeling:** The evaluated configurations and the corresponding performance measures are then used to build CART models.

Step 3 **Acquisition Modeling**: The acquisition function accepts the generated surrogate model (or models) and the pool of unevaluated configurations (uneval configs) to choose the next configuration to measure.

For multi-objective problem, for each configuration x_i, N (random projections) vectors V of length o (objectives) is generated with:

- Guess it's performance score $y_{i,j}$ using CART.
- Compute its mean weight as:

$$mean_i = \frac{1}{N} \sum_n^N \sum_j^o (V_{n,j} \cdot y_{i,j})$$

- if $mean_i > max_{current}$ then $best = x_i$.

Step 4 **Evaluating**: The configuration chosen by the acquisition function is evaluated and removed from the configuration candidates pool.

Step 5 **Terminating**: The method terminates once it runs out of predefined budget.

FLASH was invented to solve software configuration problem and it performed faster than more traditional optimizers such as Differential Evolution [60] or NSGA-II [22]. FLASH is an improvement of SMBO and was chosen as the optimizer component for X-FLASH in a few technical areas including:

- For surrogate modeling, CART is replaced with GPM. GPM would have taken $O(M^3)$ time whereas CART is a bifurcating algorithm that would only take $O(M \cdot N^2)$ where M is the size of the training dataset and N is the number of attributes.
- FLASH's acquisition function uses Maximum Mean. By assuming that the greatest mean might contain the values that most extend to the desired maximal (or minimal) goals. It cut the runtime from $O(M^2)$ to only $O(M)$.
- GPM assumed smoothness that configurations that are close to each other have similar performance. However, CART makes no assumption that neighboring regions have the same properties (i.e. the smoothness of configuration options space).

4.2.3. DODGE and ϵ-Dominance

In 2005, Deb et al. [21] proposed partitioning the output space of an optimizer into ϵ-sized grids. In multi-objective optimization (MOO), a solution is said to *dominate* the other solutions if and only if it is better in at least one objective, and no worse in other objectives. A set of optimal solutions that are not dominated by any other feasible solutions form the *Pareto frontier*. Figure 5 is an example of the output space based on ϵ-dominance. The yellow dots in the figure form the Pareto frontier.

Deb's principle of ϵ-dominance is that if there exists some ϵ value below which is useless or impossible to distinguish results, then it is superfluous to explore anything less than ϵ [21]. Specifically, consider distinguishing the type of anomalies discussed in this paper, if the performances of two learners (or a learner with various parameters) differ in less than some ϵ value, then we cannot statistically distinguish them. For the learners which do not significantly improve the performance, we can further reduce the attention on them.

Agrawal et al. [5] successfully applied ϵ-dominance to some SE tasks such as software defect prediction and SE text mining. Their proposed approach, named DODGE(ϵ), was a tabu search; i.e., if some settings resulted in some performance within ϵ of any older result, then DODGE(ϵ) marked that option as "to be avoided".

A tool for software analytics, DODGE(ϵ) needed just a few dozen evaluations to explore billions of configuration options for (a) choice of learner, for (b) choice of pre-processor, and for (c) control parameters for the learner and pre-processor. These configurations, when combined together, make up billions of options that are reported in Table 1 of [5]. DODGE(ϵ) executed by:

1. Assign weights $w = 0$ to configuration options.
2. $N = 30$ times repeat:
   (a) Randomly pick options, favoring those with most weight;
   (b) Configuring and executing data pre-processors and learners using those options;
   (c) Dividing output scores into regions of size $\epsilon = 0.2$;
   (d) if some new configuration has scores with ϵ of prior configurations then...
   - ...reduce the weight of those configuration options $w = w - 1$;
   - Else, add to their weight with $w = w + 1$.
3. Return the best option found in the above.

Note that after Step 2d, then the choices made in subsequent Step 2a will avoid options that arrive within ϵ of other observed scores.

Experiments with DODGE(ϵ) found that best learner performance plateau after just $N = 30$ repeats of Steps 2-5. To explain this result, Argrawal et al. [5] note that for a range of software analytics tasks, the outputs of a learner divide into only a handful of equivalent regions. For example, when an software analytics task is repeated 10 times, each time with 90% of the data, then the observed performance scores (e.g. recall, false alarm) can vary by 5 percent, or more. Assuming normality, then scores less than $\epsilon = 1.96 * 2 * 0.05 = 0.196$ are statistically indistinguishable. Hence, for learners evaluated on (say) $N = 2$ scores, those scores effectively divide into just $C = \left(\frac{1}{\epsilon=0.196}\right)^{N=2} = 26$ different regions. Hence, it is hardly surprising that a few dozen repeats of Steps 2-5 were enough to explore a seemingly very large space of options.

5. Methodologies

While the above DODGE(ϵ) and FLASH algorithms have been shown to work well for analytics tasks in software engineering (e.g. effort estimation, bug location etc). These algorithms have not been successfully deployed outside the realm of SE. Accordingly, the rest of this paper tests if DODGE(ϵ)

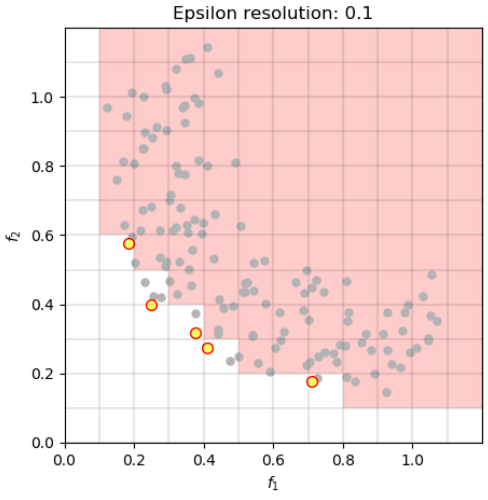


Figure 5: An example of Pareto frontier [57]. Objectives f_1 and f_2 are both to be minimized. ϵ value is set to 0.1. The gray dots represent the feasible solutions. The yellow dots represent the Pareto optimal solutions.

Table 3
Number of flows generated at Data Node under normal and anomalous conditions.

	TCP Congestion Algorithm			
Type	Cubic	Reno	Hamilton	BBR
Normal	257	265	258	221
Loss	839	811	839	225
Duplicate	530	530	530	442
Reordering	534	522	566	434

and/or FLASH work well for anomaly detection for faulty Transmission Control Protocol.

5.1. Data

5.1.1. 1000 Genome Workflow Transfers

This study case data are comprised of traffic trace from a real science workflow named 1000- Genome project workflow that reconstructs genomes of 2,504 individuals across 26 different populations [3]. This was used in the most prior state-of-the-art work for TCP anomalies detection [49]. Table 3 summarizes the data for each type of TCP data transfer flows (described in §4.1.3) across 4 datasets corresponding to TCP variants (described in §4.1.2).

5.2. Data Miners

Hyperparameter optimizers tune the control of data miners. This section describes the data miner candidates for X-FLASH.

5.2.1. CART and RF

CART and *RF*. CART recursively builds one decision tree by finding the feature whose ranges most reduce *entropy* (which is a measure of the division of class labels that call into each range) [38]. Using CART as a sub-routine, our Random Forests builds many trees, each time using different subsets of the data rows R and columns C[7]. Test data is then passed across all N trees and the conclusions are determined (say) a majority vote across all the trees [14]. Holistically, RF is based on bagging (bootstrap aggregation) which averages the results over many decision trees from sub-samples (reducing variance). Both are popular in the field of machine learning and implemented in popular open-source toolkit Scikit-learn [51].

CART is selected for it's explainability and efficiency, showed in [47, 68, 18]. RF was used in the previous study for this same problem for its advantages of performance. However, we will show later that data mining without optimization is not enough.

5.2.2. XGBoost

Gradient Boosting is chosen as a model for it's advantages of reducing both variance and bias. It is an ensemble model which involves:

- Boosting builds models from individual so called "weak learners" in an iterative way. In boosting, the individual models are not built on completely random subsets of data and features but sequentially by putting more weight on instances with wrong predictions and high errors (reducing biases).
- The gradient is a partial derivative of our loss function - so it describes the steepness of our error function in order to minimize error in the next iteration.

Gradient Boosting reduces the variances with multiple models (similar to bagging in RF) and also reduces bias with subsequently learning from previous step (boosting). XGBoost is an improved Gradient Boosting method by (1) computing second-order gradients, i.e. second partial derivatives of the loss function (instead of using CART as the loss function); and (2) advanced regularization (L1 & L2) [19].

[7] Specifically, using $\log_2 C$ of the columns, selected at random.

Table 4
Results Matrix of Multiclass Classification

	Actual			
Prediction	C1	C2	C3	C4
C1	TP_{11}	FN_{12}	FN_{13}	FN_{14}
C2	FP_{21}	TP_{22}	FN_{23}	FN_{24}
C3	FP_{31}	FP_{32}	TP_{33}	FN_{34}
C4	FP_{41}	FP_{42}	FP_{43}	TP_{44}

5.3. Evaluation Metrics

The problem studied in this paper is a multiclass classification task with 4 classes. The performance of such multiclass classifier can be assessed via a confusion matrix as shown in Table 4 where each class is denoted as C_i.

Further, "false" means the learner got it wrong and "true" means the learner correctly identified a positive or negative class. The four counts include True Positives (TP), False Positive (FP), False Negative (FN) and True Negative (TN).

Due to the multiclass and anomalies detection nature, we want to make sure all classes are treated fairly. A macro-average is preferred to compute each metric independently for each class C_i and then take the average. We used the following 3 macro-average measures that can be defined from this matrix as:

$$Accuracy = \frac{\sum_{i=1}^{L} TP_i}{N} \tag{1}$$

$$Precision = \frac{\sum_{i=1}^{L} \frac{TP_i}{TP_i+FP_i}}{L} \tag{2}$$

$$Recall = \frac{\sum_{i=1}^{L} \frac{TP_i}{TP_i+FN_i}}{L} \tag{3}$$

No evaluation criteria is "best" since different criteria are appropriate in different real-world contexts. No evaluation criteria is "best" since different criteria are appropriate in different real-world contexts. Therefore, it is recommended to evaluate such new tool on more than one solely criteria (i.e. accuracy, precision, and recall).

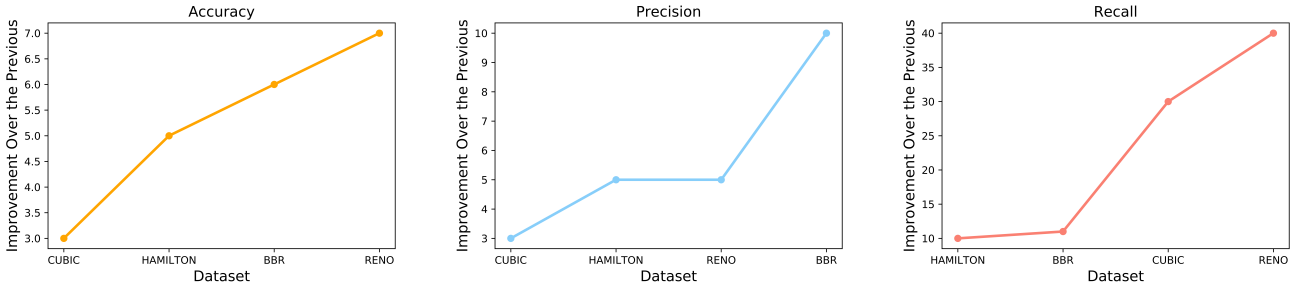


Figure 6: Sorted relative median delta for each performance across all datasets between X-FLASH and RF (previous work's choice of data miner).

5.4. Statistical Testing

We compared our results using Scott-Knott procedure. This method sorts results from different treatments, then splits them in order to maximize the expected value of differences in the observed performances before and after divisions. For lists l, m, n of size ls, ms, ns where $l = m \cup n$, the "best" division maximizes $E(\Delta)$; i.e. the delta in the expected mean value before and after the spit:

$$E(\Delta) = \frac{ms}{ls} abs(m.\mu - l.\mu)^2 + \frac{ns}{ls} abs(n.\mu - l.\mu)^2$$

Scott-Knott then checks if that "best" division is actually useful. To implement that check, Scott-Knott would apply some statistical hypothesis test H to check if m, n are significantly different (and if so, Scott-Knott then recurses on each half of the "best" division). For this study, our hypothesis test H was a conjunction of statistical significance test and an effect size test. Specifically, significance test here is non-parametric bootstrap sampling, which is useful for detecting if two populations differ merely by random noise, cliff's delta [45, 29]. Cliff's delta quantifies the number of difference between two lists of observations beyond p-values interpolation [40]. The division passes the hypothesis test if it is not a "small" effect ($Delta \geq 0.147$). The cliff's delta non-parametric effect size test explores two lists A and B with size $|A|$ and $|B|$:

$$Delta = \frac{\sum_{x \in A} \sum_{y \in B} \begin{cases} +1, & \text{if } x > y \\ -1, & \text{if } x < y \\ 0, & \text{if } x = y \end{cases}}{|A||B|} \quad (4)$$

In this expression, cliff's delta estimates the probability that a value in list A is greater than a value in list B, minus the reverse probability [40]. This hypothesis test and its effect size is supported by Hess and Kromery [33].

5.5. Test Rig

We applied k-fold cross-validation, with $k = 10$ to randomly partition the data into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k1$ subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once.

6. Results

RQ1: Does tuning improve the performance of anomalies detection?

For our first set of restuls, some untuned default learners (RF and CART and XGBOOST) are compared with data miners with optimization (FLASH and DODGE (ϵ)).

Table 5 shows those results, including the statistical ranking generated from Scott-Knott test in §4.4 for accuracy, precision, and recall metrics defined in §4.3. Across all 4 datasets (HAMILTON, BBR, RENO, and CUBIC), X-FLASH performed the best. Improvement with the previous work can be observed closely in Figure 6. X-FLASH improved up to 10% relatively for precision and accuracy but 40% relatively for recall. The benefit of tuning is even higher when comparing between XGBOOST and X-FLASH. In one extreme case, it improved 39% to 74% (which is 90% relative improvement). With a mission-critical task like anomalies detection, it is essential in order to optimize the solution at hand.

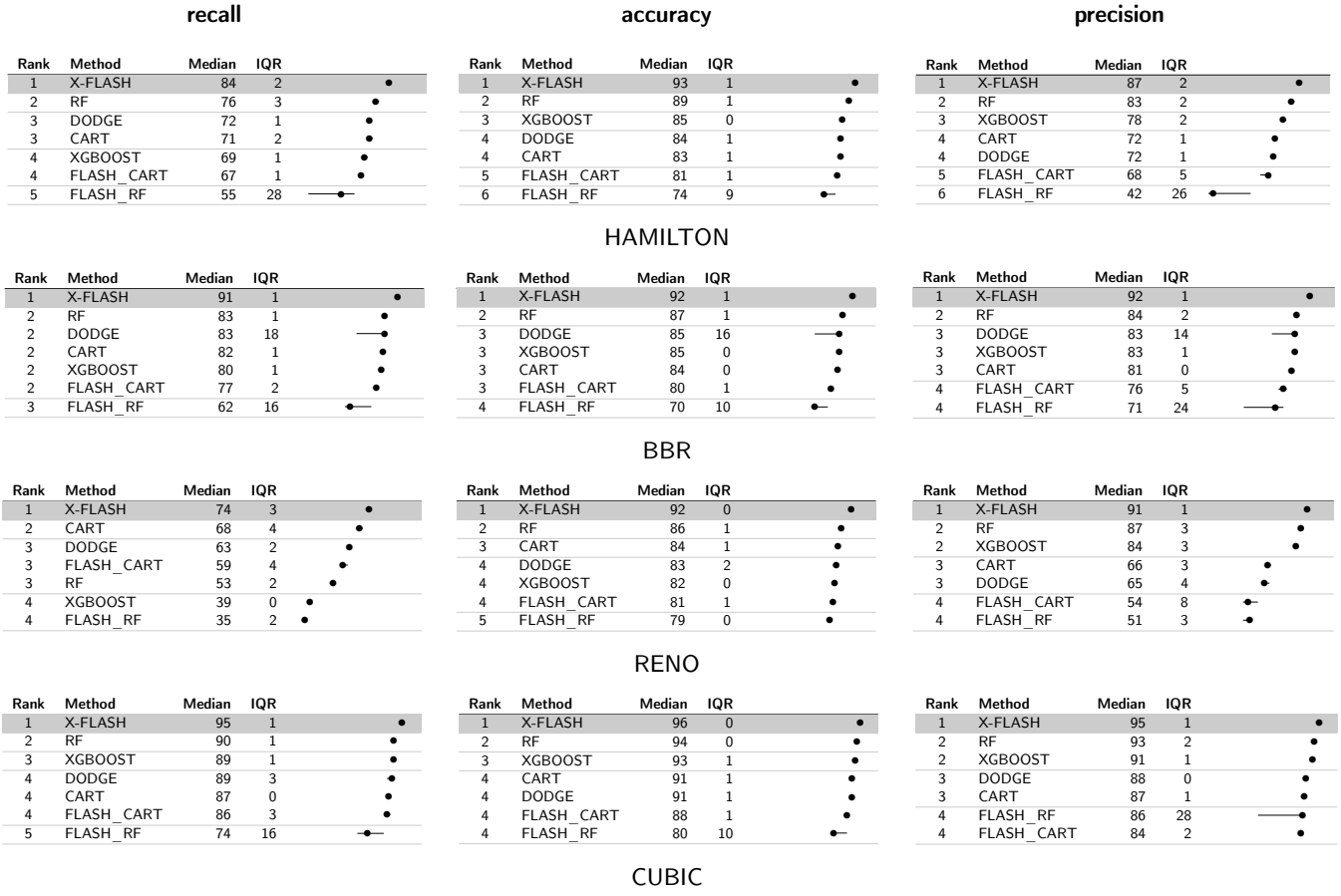
Moreover, X-FLASH outperformed DODGE(ϵ) is very notable, when DODGE(ϵ) is a state-of-the-art data mining with optimization method taken from Software Engineering literature [5] (bug reports classification, close-issues prediction, defect prediction, etc). This shows how one method that works well for a disciplinary field may not work well in a different field. It is a scientists' and researchers' responsibilities to revise and tune the method for their specialized study field.

In term of correctness...
X-FLASH was best at detecting anomalies.

RQ2: Is tuning anomalies detection impractically slow?

To our surprise, we achieved statistically significant improvement in the performance scores for our data miners in less than 30 evaluations. In this space, our proposed solution took the most time among default and the state-of-the-art optimizer DODGE(ϵ) (30 evaluations). However, considering the mission-critical nature of the problem and it still take

Table 5: Accuracy, Precision, and Recall results corresponded to first to third columns that are ranked by Scott-Knott statistical tests. The different rankings are separated by the gray line and also the numbers in the first column. The best performing data mining method is boxed in red line.



recall

Rank	Method	Median	IQR
1	X-FLASH	84	2
2	RF	76	3
3	DODGE	72	1
3	CART	71	2
4	XGBOOST	69	1
4	FLASH_CART	67	1
5	FLASH_RF	55	28

accuracy

Rank	Method	Median	IQR
1	X-FLASH	93	1
2	RF	89	1
3	XGBOOST	85	0
4	DODGE	84	1
4	CART	83	1
5	FLASH_CART	81	1
6	FLASH_RF	74	9

precision

Rank	Method	Median	IQR
1	X-FLASH	87	2
2	RF	83	2
3	XGBOOST	78	2
4	CART	72	1
4	DODGE	72	1
5	FLASH_CART	68	5
6	FLASH_RF	42	26

HAMILTON

Rank	Method	Median	IQR
1	X-FLASH	91	1
2	RF	83	1
2	DODGE	83	18
2	CART	82	1
2	XGBOOST	80	1
2	FLASH_CART	77	2
3	FLASH_RF	62	16

Rank	Method	Median	IQR
1	X-FLASH	92	1
2	RF	87	1
3	DODGE	85	16
3	XGBOOST	85	0
3	CART	84	0
3	FLASH_CART	80	1
4	FLASH_RF	70	10

Rank	Method	Median	IQR
1	X-FLASH	92	1
2	RF	84	2
3	DODGE	83	14
3	XGBOOST	83	1
3	CART	81	0
4	FLASH_CART	76	5
4	FLASH_RF	71	24

BBR

Rank	Method	Median	IQR
1	X-FLASH	74	3
2	CART	68	4
3	DODGE	63	2
3	FLASH_CART	59	4
3	RF	53	2
4	XGBOOST	39	0
4	FLASH_RF	35	2

Rank	Method	Median	IQR
1	X-FLASH	92	0
2	RF	86	1
3	CART	84	1
4	DODGE	83	2
4	XGBOOST	82	0
4	FLASH_CART	81	1
5	FLASH_RF	79	0

Rank	Method	Median	IQR
1	X-FLASH	91	1
2	RF	87	3
2	XGBOOST	84	3
3	CART	66	3
3	DODGE	65	4
4	FLASH_CART	54	8
4	FLASH_RF	51	3

RENO

Rank	Method	Median	IQR
1	X-FLASH	95	1
2	RF	90	1
3	XGBOOST	89	1
4	DODGE	89	3
4	CART	87	0
4	FLASH_CART	86	3
5	FLASH_RF	74	16

Rank	Method	Median	IQR
1	X-FLASH	96	0
2	RF	94	0
3	XGBOOST	93	1
4	CART	91	1
4	DODGE	91	1
4	FLASH_CART	88	1
4	FLASH_RF	80	10

Rank	Method	Median	IQR
1	X-FLASH	95	1
2	RF	93	2
2	XGBOOST	91	1
3	DODGE	88	0
3	CART	87	1
4	FLASH_RF	86	28
4	FLASH_CART	84	2

CUBIC

Table 6
Time performance (in seconds) on median for each dataset. The format reported is $X(Y)$ where X is the median time (s) for the method finished executing and Y is the Interquartile Range of X.

Data	DEFAULT	DODGE(ϵ)	X-FLASH
HAMILTON	10 (0.2)	103 (53)	283 (67)
BBR	4 (0.1)	49 (129)	133 (29)
RENO	14 (0.3)	200 (30)	538 (192)
CUBIC	5 (0.3)	53 (5)	88 (44)

less than 10 minutes at most with standard hardware (CPU). Modern hardware choices (e.g. GPUs) and parallel computation can be introduced and configured to improve the time to be more practical for the industry.

In term of efficiency...
X-FLASH was worst but still converged in less than 10 minutes at max.

RQ3: Does tuning change conclusions about what factors are most important in anomalies detection?

It is important to understand which attribute(s) associated more with differentiating characteristics between different types of anomalies and normal scientific flows. Scientists and network managers can then inspect the flagged ones with high likelihood of anomolies. From Table 7, among the top 10 features, only 6 to 7 features are commonly chosen as decisive factors for the anomalies detectors (while the rest 30-40% top features are not the same, non-overallped features). It demonstrated how previous study and conclusion from default learner can be untrustworthy. Previous study [49] did not even go in-depth about the features importance analysis of the anomalies detection for these TCP traces.

Moreover, interestingly, the learners rankings were also changed slightly with tuning. Default CART performed similarly or better than default XGBoost in 3/4, 2/4, and 1/4 for recall, accuracy, and precision respectively. However, after tuning through FLASH, XGBoost always better across 4 datasets for each metric.

In term of real-world applicability...
Tuning has shown how the features are considered important differently with and without it which can affect the real-world

Table 7
Non-overlapped Features selected by default learners versus after tuned by FLASH on average.

Data	DEFAULT	Tuned by FLASH
HAMILTON	c_mss_max, c_pkts_unfs, s_last_handshakeT	s_rtt_min, durat, c_appdataB
BBR	s_first_ack, c_appdataT, s_rtt_std	c_pkts_retx, s_win_min, c_pkts_reor
RENO	c_first_ack, c_appdataT, s_ack_cnt_p, s_win_max	s_rtt_cnt, s_rtt_min, c_ack_cnt_p, c_rtt_min
CUBIC	s_rtt_min, c_first, s_rtt_avg, c_pkts_rto	c_bytes_retx, c_ack_cnt_p, c_pkts_fs, c_pkts_all

7. Threats of Validity

As with any large scale empirical study, biases can affect the final results. Therefore, any conclusions made from this work must be considered with the following issues in mind:

7.1. Evaluation Bias

This paper employed accuracy, precision, and recall as defined in Equation (1)-(3). There are other evaluation scores that could be applied to this kind of analysis (e.g. G-score, harmonic mean of recall and false-alarm reported in [65]) and, in the future, it would be useful to test in the central claim of this paper holds for more than just accuracy, precision, and recall.

To increase the validity of our results, we applied two statistical tests: bootstrap significant test and the cliff-delta effect sizet. Hence, anytime in this paper we reported that "X was different from Y" then that report was based on both an effect size and a statistical significance test.

7.2. Learner Bias

This study utilized X-FLASH (XGBoost + FLASH) as a proposed data mining method and compared it with other data mining methods (e.g. DODGE (ϵ)). Nevertheless, it might be useful in future work to test if the central claim of this paper (that data miner + optimization, called X-FLASH, is a good way to detect and classify anomalies) hold across multiple tasks associated with scientific workflows.

7.3. Sampling Bias

Like any data mining paper, our work is threatened by sampling bias; i.e. what holds for the data we studied here may not hold for other kinds of data. However, what researchers can do is make all their scripts and data available such that other researchers can test their conclusions whenever new data becomes available. To that end, we have made all our scripts and data available at https://github.com/ai-se/tuning_workflow/.

7.4. External Validity

The approach, DODGE(ϵ), that SE literature have established as "standard tool" may not be "general" at all. Rather, they may be tools that are powerful in their home domain but need to be used with care if applied to new domains such as scientific workflow (and specifically, anomalies detection). Some of the data quirks that essential to the success of DODGE(ϵ) include: (1) the prediction is binary (e.g. 0 or 1, faulty or non-faulty, etc); and (2) the target class is infrequent. Those data quirks lead to:

- It is harder to find the target;
- The larger the observed ϵ in the results;
- The greater the number of redundant tunings;

That is, since software engineering often deals with relatively infrequent target classes, then we should expect to see a large ϵ uncertainty in our conclusions which is more likely that DODGE(ϵ) will work. However, for the faulty TCP file transfers detection in this paper, it is a multiclass classification and the distribution is only infrequent for the normal flows instead of the targets (i.e. anomalies flows). Therefore, for such a more dynamic problem with multiple target classes and the distribution is diverse, FLASH is recommended as a more general approach for optimization.

In summary, and in support of the general theme of this paper, this external validity demonstrates the danger of treating all data with the state-of-the-art method, especially when switching domain (e.g. DODGE(ϵ) from SE literatures).

8. Conclusion

Our exploration shows that when learning anomalies detectors for faulty TCP file transfers without tuning are considered *harmful* and *misleading* to the reliability of networked infrastructures. Among the cyberinfrastrucutre, such anomalies may change the experimental results that lead to scientific progress can be forestalled and discovery results can be even refuted.

Our proposed solution X-FLASH combined an ensemble model (XGBoost) and a sequential multiobjective optimizer (FLASH) from Software Engineering to detect and classify the correct malicious activity or attacks, before it contaminates downstream scientific process:

- Tuning default learners will improve the relative performance up to 10%, 10%, and 40% for accuracy, precision, and recall (see Table 5).
- Tuning changes conclusions on what factors are most important in detecting for anomalies by 30%-40% (see Table 7).

Moreover, the RQ1 also showed how X-FLASH out-performed state-of-the-art data mining in SE literature, DODGE(ϵ) [5]. This result is suggestive (but not conclusive) evidence that (a) prior work on analytics has *over-fitted* methods (to systems like Apache); and that (b) there is *no better time* than now to develop new case studies (like scientific workflows).

As to future work, it is now important to explore the implications of these conclusions to other kinds of scientific workflow analytics. Specifically, old papers for anomalies detection for scientific workflow [42, 27, 54, 24, 55, 24, 16, 12, 56] that not based on TCP data transfers should be reinvestigated as none have done tuning study.

9. Acknowledgements

We thank the CS community from the Pegasus Research group and Renaissance Computing Institute at UNC (RENCI) for their assistance with this work.

This work was partially funded by an NSF CISE Grant #1826574 and #1931425.

References

[1] , 2016. Tuning for software analytics: Is it really necessary? Information and Software Technology .

[2] , 2020. Cisco annual internet report (2018–2023) white paper. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[3] A, A., Abecasis, G., DM, A., RM, D., DR, B., A, C., AG, C., P, D., EE, E., P, F., SB, G., Gibbs, R., ED, G., ME, H., Knoppers, B., JO, K., ES, L., Lee, C., Lehrach, H., JA, S., 2015. A global reference for human genetic variation. Nature .

[4] Afanasyev, A., Tilley, N., Reiher, P., Kleinrock, L., 2010. Host-to-host congestion control for tcp. IEEE Communications Surveys Tutorials .

[5] Agrawal, A., Fu, W., Chen, D., Shen, X., Menzies, T., 2019. How to "dodge" complex software analytics. Transactions on Software Engineering .

[6] Agrawal, A., Fu, W., Menzies, T., 2018a. What is wrong with topic modeling? and how to fix it using search-based software engineering. Information and Software Technology .

[7] Agrawal, A., Fu, W., Menzies, T., 2018b. What is wrong with topic modeling? and how to fix it using search-based software engineering. IST .

[8] Agrawal, A., Menzies, T., 2018a. Is “better data" better than “better data miners" (benefits of tuning smote for defect prediction). International Conference on Software Engineering .

[9] Agrawal, A., Menzies, T., 2018b. Is “better data” better than “better data miners”? on the benefits of tuning smote for defect prediction, in: ICSE.

[10] Ahmed, M., Naser Mahmood, A., Hu, J., 2016. A survey of network anomaly detection techniques. J. Netw. Comput. Appl. .

[11] Alipourfard, O., Liu, H.H., Chen, J., Venkataraman, S., Yu, M., Zhang, M., 2017. Cherrypick: Adaptively unearthing the best cloud configurations for big data analytics, in: NSDI.

[12] Bala, A., Chana, I., 2015. Intelligent failure prediction models for scientific workflows. Expert Systems with Applications URL: http://www.sciencedirect.com/science/article/pii/S0957417414005533, doi:https://doi.org/10.1016/j.eswa.2014.09.014.

[13] Baldine, I., Xin, Y., Mandal, A., Ruth, P., Heerman, C., Chase, J., 2012. Exogeni: A multi-domain infrastructure-as-a-service testbed, in: Korakis, T., Zink, M., Ott, M. (Eds.), Testbeds and Research Infrastructure. Development of Networks and Communities.

[14] Breiman, L., 2001. Random forests. Machine Learning URL: https://doi.org/10.1023/A:1010933404324, doi:10.1023/A:1010933404324.

[15] Brochu, E., Cora, V.M., de Freitas, N., 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. CoRR .

[16] Buneci, E.S., Reed, D.A., 2008. Analysis of application heartbeats: Learning structural and temporal features in time series data for identification of performance problems, in: SC.

[17] Cardwell, N., Cheng, Y., Gunn, C.S., Yeganeh, S.H., Jacobson, V., 2016. Bbr: Congestion-based congestion control. Queue .

[18] Chen, J., Chakraborty, J., Clark, P., Haverlock, K., Cherian, S., Menzies, T., 2019. Predicting breakdowns in cloud services (with spike), in: ESEC/FSE.

[19] Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system, in: SIGKDD.

[20] Dalibard, V., Schaarschmidt, M., Yoneki, E., 2017. Boat: Building auto-tuners with structured bayesian optimization, in: WWW.

[21] Deb, K., Mohan, M., Mishra, S., 2005. Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evolutionary computation 13, 501–525.

[22] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6, 182–197. doi:10.1109/4235.996017.

[23] Deelman, E., Mandal, A., Jiang, M., Sakellariou, R., 2019. The role of machine learning in scientific workflows. The International Journal of High Performance Computing Applications .

[24] Dinal Herath, J., Bai, C., Yan, G., Yang, P., Lu, S., 2019. Ramp: Real-time anomaly detection in scientific workflows, in: Big Data.

[25] Fu, W., Menzies, T., Shen, X., 2016a. Tuning for software analytics: Is it really necessary? Information and Software Technology 76, 135–146.

[26] Fu, W., Nair, V., Menzies, T., 2016b. Why is differential evolution better than grid search for tuning defect predictors? CoRR abs/1609.02613.

[27] Gaikwad, P., Mandal, A., Ruth, P., Juve, G., Król, D., Deelman, E., 2016. Anomaly detection for scientific workflow applications on networked clouds, in: HPCS.

[28] Ghasemi, M., Benson, T., Rexford, J., 2017. Dapper: Data plane performance diagnosis of tcp.

[29] Ghotra, B., McIntosh, S., Hassan, A.E., 2015. Revisiting the impact of classification techniques on the performance of defect prediction models, in: Proceedings of the 37th International Conference on Software Engineering-Volume 1, IEEE Press. pp. 789–800.

[30] Goldberg, D.E., Holland, J.H., 1988. Genetic algorithms and machine learning. Mach. Learn. .

[31] Ha, S., Rhee, I., Xu, L., 2008. Cubic: A new tcp-friendly high-speed tcp variant. SIGOPS Oper. Syst. Rev. .

[32] Harman, M., Mansouri, S.A., Zhang, Y., 2012. Search-based software engineering: Trends, techniques and applications. ACM Comput. Surv. .

[33] Hess, M.R., Kromrey, J.D., 2004. Robust confidence intervals for effect sizes: A comparative study of cohen’sd and cliff’s delta under non-normality and heterogeneous variances, in: annual meeting of the American Educational Research Association, pp. 1–30.

[34] Hill, D., Nassif, H., Liu, Y., Iyer, A., Vishwanathan, S., 2017. An efficient bandit algorithm for realtime multivariate optimization, in: SIGKDD.

[35] Jacobson, V., 1988. Congestion avoidance and control.

[36] Jamshidi, P., Casale, G., 2016. An uncertainty-aware approach to optimal configuration of stream processing systems.

[37] Jinka, P., 2015. Anomaly Detection for Monitoring: A Statistical Approach to Time Series Anomaly Detection. O’Reilly Media. URL: https://books.google.com/books?id=DnnuuQEACAAJ.

[38] L. Breiman, J. H. Friedman, R.A.O., Stone, C.J., 1987. Classification and regression trees. Cytometry .

[39] Lukaseder, T., Bradatsch, L., Erb, B., Van Der Heijden, R.W., Kargl, F., 2016. A comparison of tcp congestion control algorithms in 10g networks, in: LCN.

[40] Macbeth, G., Razumiejczyk, E., Ledesma, R.D., 2011. Cliff’s delta calculator: A non-parametric effect size program for two groups of observations. Universitas Psychologica 10, 545–555.

[41] Majumder, S., Balaji, N., Brey, K., Fu, W., Menzies, T., 2018. 500+ times faster than deep learning (a case study exploring faster methods for text mining stackoverflow), in: Mining Software Repositories (MSR), 2018 IEEE/ACM 15th International Conference on, ACM.

[42] Mandal, A., Ruth, P., Baldin, I., Król, D., Juve, G., Mayani, R., Silva, R.F.D., Deelman, E., Meredith, J., Vetter, J., Lynch, V., Mayer, B., Wynne, J., Blanco, M., Carothers, C., Lapre, J., Tierney, B., 2016. Toward an end-to-end framework for modeling, monitoring and anomaly detection for scientific workflows, in: IPDPSW.

[43] Mellia, M., Meo, M., Muscariello, L., Rossi, D., 2008. Passive analysis of tcp anomalies. Comput. Netw. .

[44] Miras, D., Bateman, M., Bhatti, S., 2008. Fairness of high-speed tcp stacks, in: 22nd International Conference on Advanced Information Networking and Applications (aina 2008).

[45] Mittas, N., Angelis, L., 2013. Ranking and clustering software cost estimation models through a multiple comparisons algorithm. IEEE Transactions on software engineering 39, 537–551.

[46] Nair, V., Menzies, T., Siegmund, N., Apel, S., 2017. Using bad learners to find good configurations, in: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, pp. 257–267.

[47] Nair, V., Yu, Z., Menzies, T., Siegmund, N., Apel, S., 2018. Finding faster configurations using flash. TSE , 1–1doi:10.1109/TSE.2018.2870895.

[48] Padhye, J., Firoiu, V., Towsley, D.F., Kurose, J.F., 2000. Modeling tcp reno performance: a simple model and its empirical validation. Transactions on Networking .

[49] Papadimitriou, G., Kiran, M., Wang, C., Mandal, A., Deelman, E., 2019. Training classifiers to identify tcp signatures in scientific workflows, in: INDIS.

[50] Parichehreh, A., Alfredsson, S., Brunstrom, A., 2018. Measurement analysis of tcp congestion control algorithms in lte uplink, in: TMA.

[51] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al., 2011. Scikit-learn: Machine learning in python. Journal of Machine Learning Research 12, 2825–2830.

[52] Pordes, R., Petravick, D., Kramer, B., Olson, D., Livny, M., Roy, A., Avery, P., Blackburn, K., Wenaus, T., Würthwein, F., Foster, I., Gardner, R., Wilde, M., Blatecky, A., McGee, J., Quick, R., 2007. The open science grid. Journal of Physics: Conference Series doi:10.1088/1742-6596/78/1/012057.

[53] Prieto, R., Mora, M., Vittore, V., 2015. Cisco global cloud index projects cloud traffic to quadruple by 2019. URL: https://newsroom.cisco.com/press-release-content?articleId=1724918.

[54] Rodriguez, M.A., Kotagiri, R., Buyya, R., 2018. Detecting performance anomalies in scientific workflows using hierarchical temporal memory. Future Generation Computer Systems .

[55] Samak, T., Gunter, D., Goode, M., Deelman, E., Juve, G., Mehta, G., Silva, F., Vahi, K., 2011. Online fault and anomaly detection for large-scale scientific workflows, in: HPCC.

[56] Samak, T., Gunter, D., Goode, M., Deelman, E., Juve, G., Silva, F., Vahi, K., 2012. Failure analysis of distributed scientific workflows executing in the cloud, in: CNSM.

[57] Shu, R., Xia, T., Chen, J., Williams, L., Menzies, T., 2019. Improved recognition of security bugs via dual hyperparameter optimization. arXiv:1911.02476.

[58] Sommerhauser, C., Mora, M., Faulkner, M., 2018. Global cloud index projects cloud traffic to represent 95 percent of total data center traffic by 2021. URL: https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1908858.

[59] Storn, R., Price, K., 1997a. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11.

[60] Storn, R., Price, K.V., 1997b. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359.

[61] Tantithamthavorn, C., McIntosh, S., Hassan, A.E., Matsumoto, K., 2016. Automated parameter optimization of classification techniques for defect prediction models, in: ICSE.

[62] Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M., 2014. Workflows for E-Science: Scientific Workflows for Grids.

[63] di Torino, T.N.G.P., 2016. Tstat: Log tcp complete.

[64] Treude, C., Wagner, M., 2019. Predicting good configurations for github and stack overflow topic models, in: MSR.

[65] Tu, H., Yu, Z., Menzies, T., 2020. Better data labelling with emblem (and how that impacts defect prediction). IEEE Transactions on Software Engineering doi:10.1109/TSE.2020.2986415.

[66] Venkataraman, S., Yang, Z., Franklin, M., Recht, B., Stoica, I., 2016. Ernest: Efficient performance prediction for large-scale advanced analytics, in: NSDI.

[67] Wang, Y., Yin, D., Jie, L., Wang, P., Yamada, M., Chang, Y., Mei, Q., 2018. Optimizing whole-page presentation for web search. ACM Trans. Web .

[68] Xia, T., Shu, R., Shen, X., Menzies, T., 2019. Sequential model optimization for software process control.

[69] Yadwadkar, N.J., Hariharan, B., Gonzalez, J.E., Smith, B., Katz, R.H., 2017. Selecting the best vm across multiple public clouds: A data-driven performance modeling approach, in: SoCC.

[70] Zhou, Y., 2019. The human dimension of cloud computing, ase.

[71] Zhu, H., Jin, J., Tan, C., Pan, F., Zeng, Y., Li, H., Gai, K., 2017a. Optimized cost per click in taobao display advertising, in: SIGKDD.

[72] Zhu, Y., Liu, J., Guo, M., Bao, Y., Ma, W., Liu, Z., Song, K., Yang, Y., 2017b. Bestconfig: Tapping the performance potential of systems via automatic configuration tuning, in: SoCC.