AutoML from Service Provider's Perspective: Multi-device, Multi-tenant Model Selection with GP-EI

Chen Yu
Department of Computer Science, University of Rochester

Bojan Karlaš
Deparment of Computer Science, ETH Zurich

Jie Zhong
Deparment of Mathematics, California State University Los Angeles

Ce Zhang
Deparment of Computer Science, ETH Zurich

Ji Liu
Department of Computer Science, University of Rochester

Abstract

AutoML has become a popular service that is provided by most leading cloud service providers today. In this paper, we focus on the AutoML problem from the *service provider's perspective*, motivated by the following practical consideration: When an AutoML service needs to serve *multiple users* with *multiple devices* at the same time, how can we allocate these devices to users in an efficient way? We focus on GP-EI, one of the most popular algorithms for automatic model selection and hyperparameter tuning, used by systems such as Google Vizer. The technical contribution of this paper is the first multi-device, multi-tenant algorithm for GP-EI that is aware of *multiple* computation devices and multiple users sharing the same set of computation devices. Theoretically, given N users and M devices, we obtain a regret bound of $O((\mathbf{MIU}(T,K)+M)\frac{N^2}{M})$, where $\mathbf{MIU}(T,K)$ refers to the maximal incremental uncertainty up to time T for the covariance matrix K. Empirically, we evaluate our algorithm on two applications of automatic model selection, and show that our algorithm significantly outperforms the strategy of serving users independently. Moreover, when multiple computation devices are available, we achieve near-linear speedup when the number of users is much larger than the number of devices.

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

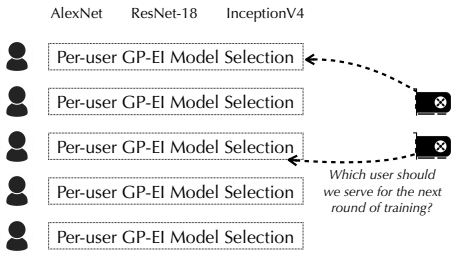


Figure 1: Multi-device, Multi-tenant Model Selection

1 INTRODUCTION

One of the next frontiers of machine learning research is its accessibility — How can we make machine learning systems easy to use such that users do not need to worry about decisions such as model selection and hyperparameter tuning as much as today? The industry's answer to this question seems to be making AutoML services available on the cloud, and prominent examples include Google Cloud AutoML and Microsoft Cognitive Services. In these services, users are provided with a single interface for uploading the data, automatically dealing with hyperparameter tuning and/or model selection, and returning a model directly without any user intervention as shown in Figure 1. Bayesian optimization is one of the core techniques that make AutoML services possible by strategically planning a series of models and hyperparameter configurations to tune and try.

From the service provider point of view, resource allocation is the problem coming together with the emerging popularity of such a service — When an online AutoML service needs to serve multiple users with limited number of devices, what is the most cost efficient way of allocating these devices to different users? During our conversation with multiple cloud service providers, many of them believe that an effective and cost efficient resource sharing could be of great practical interests

and is a natural technical question to ask.

In this paper, we focus on GP-EI, one of the most popular algorithms for AutoML that is used in systems such as Google Vizier Golovin et al. [2017] and Spearmint Snoek et al. [2012]. Specifically, we are interested in the scenarios that each user runs her own GP-EI instance on a different machine learning task, and there are multiple devices, each of which can only serve one user at the same time. *How to allocate resources? What are the theoretical properties of such an algorithm?*

The result of this paper is the first *multi-device*, *multi-tenant*, *cost sensitive* GP-EI algorithm that aims at optimizing for the "global happiness" of all users given limited resources. In order to analyze its performance, we introduce a new notation of Maximum Incremental Uncertainty (**MIU**) to measure dependence among all models. Given N users and M devices, the upper bound of cumulative regret is $O((\mathbf{MIU}(T,K)+M)\frac{N^2}{M})$, where $\mathbf{MIU}(T,K)$ will be specified in Section **5.2**. This bound converges to optimum and is nearly linear speedup when more devices are employed, which will be discussed also in Section **5.2**.

We evaluate our algorithms on two real-world datasets: (1) model selection for image classification tasks that contains 22 datasets (users) and 8 different neural network architectures; and (2) model selection for models available on Microsoft Azure Machine Learning Studio that contains 17 datasets and 8 different machine learning models. We find that, multi-tenant GP-EI outperforms standard GP-EI serving users randomly or in a round robin fashion, sometimes by up to 5× in terms of the time it needs to reach the same "global happiness" of all users. When multiple devices are available, it can provide near linear speedups to the performance of our algorithm.

2 RELATED WORK

Multi-armed bandit problem There are lots of research on the multi-armed bandit problem. Some early work such as Lai and Robbins [1985] provided the lower bound of stochastic multi-armed bandit scenario and designed an algorithm to attain this lower bound. Then many research focused on improving constants of the bound and designing distribution-free-bound algorithms such as upper confidence bound (UCB) [Auer et al., 2002] and Minimax Optimal Strategy in the Stochastic case (MOSS) [Audibert and Bubeck, 2009]. UCB is now becoming a very important algorithm in bandit problem. Lots of algorithms are based on UCB, such as LUCB [Kalyanakrishnan et al., 2012], GP-UCB [Srinivas et al., 2012]. The UCB algorithm constructs the upper confident bound for each arm in every iteration, and chooses the arm with largest bound as the next arm to observe. UCB is very efficient by effectively balancing exploration and exploitation, admitting the regret upper bound $O(\sqrt{T|\mathcal{L}|\log T})$ [Bubeck et al., 2012], where T is running time, $\mathcal{L}$ is the set of arms. There also exist some variations of the bandit problem other than stochastic bandit problem, such as contextual bandit problem [Deshmukh et al., 2017, Langford and Zhang, 2008] and bandit optimization problem [Arora et al., 2012, Hazan et al., 2016]. We recommend readers to refer to the book by [Bubeck et al., 2012]. Regret is a common metric of algorithms above, while another important metric is the sample complexity [Gabillon et al., 2012, Kalyanakrishnan et al., 2012]. Recently there are also some research to combine bandit algorithms and Monte Carlo tree methods to find out an optimal path in a tree with random leaf nodes, that corresponds to finding the optimal strategy in a game. Representative algorithms include UCT [Kocsis et al., 2006], UGapE-MCTS [Kaufmann and Koolen, 2017], and LUCB-micro [Huang et al., 2017].

Expected improvement methods The expected improvement method dates back to 1970s, when J. Mockus and Zilinskas [1978] proposed to use the expected improvement function to decide which arm to choose in each iteration, that is, the arm is chosen with the maximal expected improvement. The advantage of this method is that the expected improvement can be computed analytically [J. Mockus and Zilinskas, 1978, Jones et al., 1998]. Recently, Snoek et al. [2012] extended the idea of expected improvement to the time sensitive case by evaluating the expected improvement per second to make selection. We also adopt this concept in this paper, namely, **EIrate**. There are also some works on analyzing the asymptotically convergence of EI method. Ryzhov [2016] analyzed the hit number of each non-optimal arm and the work by Bull [2011] provides the lower and upper bound of instantaneous regret when values of all arms are in Reproducing-Kernel Hilbert Space. Expected improvement methods have many application scenarios, see [Malkomes et al., 2016].

GP-UCB GP-UCB is a type of approaches considering the correlation among all arms, while the standard UCB [Auer et al., 2002] does not consider the correlation. GP-UCB chooses the arm with the largest UCB value in each iteration where the UCB value uses the correlation information. The proven regret achieves the rate $O(\sqrt{T\log T\log|\mathcal{L}|\gamma_T})$ where T is running time, $\mathcal{L}$ is the set of arms, and γ_T is the maximum information gain at time T. Some variants of GP-UCB include Branch and Bound algorithm [de Freitas et al., 2012], EGP algorithm [Rana et al., 2017], distributed

batch GP-UCB [Daxberger and Low, 2017], MF-GP-UCB [Kandasamy et al., 2016], BOCA [Kandasamy et al., 2017], GP-(UCB/EST)-DPP-(MAX/SAMPLE) [Kathuria et al., 2016], to name a few.

Parallelization bandit algorithms To improve the efficiency of bandit algorithms, multiple agents can be employed and they can perform simultaneous investigation. Zhong et al. [2017] designed an asynchronous parallel bandit algorithm that allows multiple agents working in parallel without waiting for each other. Both theoretical analysis and empirical studies validate that the nearly linear speedup can be achieved. Kandasamy et al. [2018] designed an asynchronous version of Thompson sampling algorithm to solve parallelization Bayesian bandit optimization problem. While they consider the single user scenario, our work extends the setup to the multi-user case, which leads to our new notation to reflect the *global happiness* and needs some new technique in theoretical analysis.

AutoML Closely related to this work is the emerging trend of AutoML system and services. Model selection and hyperparameter tuning is the key technique behind such services. Prominent systems include Spark TuPAQ [Sparks et al.], Auto-WEKA [Kotthoff et al., Thornton et al.], Google Vizier [Golovin et al.], Spearmint [Snoek et al., 2012], GPyOpt [GPyOpt, 2016], and Auto scikit-learn [Feurer et al.] and prominent online services include Google Cloud AutoML and Microsoft Cognitive Services. Most of these systems focus on a single-tenant setting. Recently, Li et al. [2018] describes a system for what the authors call "multi-tenant model selection". Our paper is motivated by this multi-tenant setting, however, we focus on a much more realistic choice of algorithm, GP-EI, that is actually used by real-world AutoML services.

3 MATHEMATICAL PROBLEM STATEMENT

In this section, we give the mathematical expression of the problem. We first introduce the multi-device, multi-tenant (MDMT) AutoML problem, which is a more general scenario of single-device, single-tenant (SDST) AutoML problem. Then in Section 3.1, we propose a new notion – time sensitive hierarchical bandit (TSHB) problem to abstract the MDMT AutoML problem. At last in Section 3.2, we define a new metric to quantify the goal.

Single-device, single-tenant (SDST) AutoML AutoML often refers to the single-device, single-tenant scenario, in which a single user aims finding the best model (or hyper parameter) for his or her individual dataset as soon as possible. Here the device is an abstract concept, it can refer to a server, or a CPU, GPU. The problem is usually formulated into a Bayesian optimization problem or a bandit problem using Gaussian process to characterize the connection among different models (or hyper parameters).

Multi-device, multi-tenant (MDMT) AutoML The MDMT AutoML considers the scenario where there are multiple devices available and multiple tenants who seek the best model for each individual dataset (one tenant corresponds to one dataset). While the objective of the SDST AutoML is purely from the perspective of a customer, the objective of MDMT AutoML is from the *service provider*, because it is generally impossible to optimize performance for each single one, given the limited computing resource. There are two fundamental challenges: 1) When there are multiple devices are available, how to coordinate all computing resources to maximize the efficiency? Simply extending the algorithms in SDST AutoML often let multiple devices to run the same model on the same dataset, which apparently wastes the computing resource; 2) To serve multiple customer, we need to find a global metric to guide us to specify the most appropriate customer to serve besides of choosing the most promising the model for his / her dataset for each time. The overall problem is how to utilize all devices to achieve a certain global happiness. Each device is considered to be atomic, that is, each device can only run one algorithm (model) on one dataset at the same time.

3.1 Time sensitive hierarchical bandit (TSHB) problem

To find a systematic solution to the MDMT AutoML problem, we develop a new notion – time sensitive hierarchical bandit (TSHB) problem to formulate the MDMT AutoML problem.

Definition of TSHB problem Now we formally propose the *ime sensitive hierarchical bandit problem* to abstract the multi-device, multi-tenant autoML framework. Suppose that there are N *users* (or datasets in autoML framework) and M *devices*. Each user has a candidate set of *models* (or algorithms in autoML framework) he or she is interested, specifically, $\mathcal{L}_i$ is the candidate model set for user $i \in [N]$. Here we consider a more general situation, that is we do not assume that $\mathcal{L}_i$ and $\mathcal{L}_j$ are disjoint for $i, j \in [N]$. Denote the set of all models by $\mathcal{L} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \cdots \cup \mathcal{L}_N$. Running a model $x \in \mathcal{L}$ on a device will take $c(x)$ units of time. One model can only be run on one device at the same time and one device can only run one model at the same time. Since one model has been assigned to an idle device, $c(x)$ units time later, the performance of

model x will be observed, denoted by $z(x)$. For example, the performance could be the accuracy of the model. W.L.O.G., we assume that the larger the value of $z(x)$, the better. Roughly saying, the overall goal is to utilize M devices to help N users to find out each individual optimal model from the corresponding candidate set as soon as possible. More technically, the goal (from the perspective of service provider) is to maximize the *cumulative global happiness* over all users. We will define a regret to reflect this metric in the next subsection.

Remark 1. *Here we simply assume $c(x)$ to be known beforehand. Although it is usually unknown beforehand, it is easy to estimate an approximate (but high accurate) value by giving the dataset set size, the computational hardware parameters, historical data, and other information. Therefore, for simplicity in analysis, we just use estimated value so that we can assume the runtime of each model to be exactly known beforehand. In our empirical study, this approximation does not degrade the performance of our algorithm.*

3.2 Regret definition for cumulative global happiness

To quantify the goal – cumulative global happiness, we define the corresponding regret. Let us first introduce some more notations and definitions:

- $\mathcal{L}(t)$: the set of models whose performances have been **observed** up to time t;
- x_i^*: the best model for user i, that is, $x_i^* := \arg\max_{x\in\mathcal{L}_i} z(x)$;
- $x_i^*(t)$: the best model for user i observed up to time t, that is,

$$x_i^*(t) := \arg\max_{x\in\mathcal{L}(t)\cap\mathcal{L}_i} z(x). \quad (1)$$

We define the individual regret (or negative individual happiness) of user i at time t by the gap between the currently best and the optimal, i.e., $z(x_i^*) - z(x_i^*(t))$.

In most AutoML systems, the user experience goes beyond the regret at a single time point for a single user. Instead, the regret is defined by the integration over time and the sum over all users' regrets. It is worth noting that the regret for each user is not the same as the one in the SDST scenario since even a user is not served currently, he or she still receives the penalty (measured by the gap between the optimal model's performance and the currently best performance). More formally, the regret at time T is defined by

$$\mathbf{Regret}_T = \sum_{i=1}^{N} \int_0^T \Big(z(x_i^*) - z\big(x_i^*(t)\big)\Big)\mathrm{d}t. \quad (2)$$

Our goal is to utilize all devices to minimize this regret.

Discussion: Why is Multi-Device Important? Having multiple devices in the pool of computation resources does not necessarily mean that we need to have a multi-device GP-EI algorithm to do the scheduling. One naive solution, which is adopted by ease.ml Li et al. [2018] is to treat all devices as a single device to do *distributed* training for each training task. In fact, If the training process can scale up linearly to all devices in the computation pool, such a baseline strategy might not be too bad. However, the availability of resources provided on a modern cloud is far larger than the current limitation of distributed training — Whenever the scalability becomes sublinear, some resources could be used to serve other users instead. Given the growth rate of online machine learning services, we believe the multi-device setting will only become more important.

4 ALGORITHM

The proposed Multi-device Multi-tenant GP-EI (MM-GP-EI) algorithm follows a simple philosophy – as long as there is a device available, select a model to run on this device. To minimize the regret (or equivalently maximize the cumulative global happiness), the key is to select a promising model to run whenever there is an available device. We use the expected improvement rate (**EIrate**) to measure the quality of each model in the set of models that have not yet been selected before (selected models include the ones whose performance have been observed or that are under test currently). The **EIrate** value for model x depends on two factors: the running time of model x and its expected improvement (**EI**) value (defined as Expected Improvement Function in Section 4.1)

$$\mathbf{EIrate}(x) := \mathbf{EI}(x)/c(x).$$

This measures the averaged expected improvement. This concept also appeared in Snoek et al. [2012].

4.1 Expected Improvement Function

Every Bayesian-based optimization algorithm has a function called acquisition function [Brochu et al., 2010] that guides the search for the next model to test. In our algorithm, we will call it expected improvement function (EI function).

Suppose at time t there is a device free, we first compute posterior distributions for all models given all current observation and then use these posterior distributions to construct EI function for every model.

First, for each model x and any user who has this model (notice that different users can share the same model), we use $\mathbf{EI}_{i,t}(x)$ to denote expected improvement of user i's best performance if model x is observed. Formally,

we have

$$\mathbf{EI}_{i,t}(x) = \mathbb{E}\Big[\max\big\{z(x) - z\big(x_i^*(t)\big), 0\big\}\Big]. \quad (3)$$

Here $\mathbb{E}$ means taking expectation of posterior distribution of $z(x)$ at time t.

Then, we sum this value over all users who have model x to represent the total expected improvement $\mathbf{EI}_t(x)$ if model x is observed. Formally, we have

$$\mathbf{EI}_t(x) = \sum_{i=1}^{N} \mathbb{1}(x \in \mathcal{L}_i)\mathbf{EI}_{i,t}(x), \quad (4)$$

where $\mathbb{1}(A) = 1$ if A happens, and $\mathbb{1}(A) = 0$ if A does not happen. At last, we define the **EIrate** value of x at time t as follows:

$$\mathbf{EIrate}_t(x) = \frac{\mathbf{EI}_t(x)}{c(x)}. \quad (5)$$

Now, we can choose the model with the max value of EIrate as the next one to run at time t:

$$x_{\text{next to run at time } t} = \arg\max_{x \in \mathcal{L}\setminus\mathcal{L}(t)} \mathbf{EIrate}_t(x). \quad (6)$$

Algorithm 1 MM-GP-EI Algorithm

Input: $\mu(x)$, $k(x,x')$, $c(x)$, $\mathcal{L}$, $\{\mathcal{L}_i\}_{i=1}^N$ and the total time budget T.

1: $x_{\text{initial}}^{(i)} = \arg\max_{x\in\mathcal{L}_i} \mu(x), \forall i \in [N]$.
2: $\mathcal{L}_{\text{ob}} = \big\{x_{\text{initial}}^{(i)}\big\}_{i=1}^N$
3: **while** there is a device available and the elapsed time t is less than T **do**
4: Refresh $\mathcal{L}_{\text{ob}}$ to include all observed models at present
5: Update posterior mean $\mu_t(\cdot)$, posterior covariance $k_t(\cdot,\cdot')$ of $z(x)$ given $\{z(x)\}_{x\in\mathcal{L}_{\text{ob}}}$
6: Update $x_i^*(t) = \arg\max_{x\in\mathcal{L}_{\text{ob}}\cap\mathcal{L}_i} z(x), \forall i \in [N]$
7: $\mathbf{EI}(x) = \sum_{i=1}^{N} \sum_{x\in\mathcal{L}_i\setminus\mathcal{L}_{\text{ob}}} \sigma_t(x)\tau\Big(\frac{\mu_t(x) - z\left(x_i^*(t)\right)}{\sigma_t(x)}\Big), \forall x \in \mathcal{L}$
8: Run $x_{\text{next}} = \arg\max_{x\in\mathcal{L}\setminus\mathcal{L}_{\text{ob}}} \frac{\mathbf{EI}(x)}{c(x)}$ on this free device
9: **end while**

Output: $x_1^*(T), x_2^*(T), \cdots, x_N^*(T)$.

4.2 Choosing Prior: Gaussian Process

Next, we must choose a suitable prior of $z(x)$ to estimate **EI** function in (3). Here, we choose Gaussian Process (GP) as the prior like many other Bayesian optimization algorithms [Bull, 2011, Srinivas et al., 2012], mainly because of its convenience of computing posterior distribution and **EI** function.

A Gaussian Process $GP(\mu(x), k(x,x'))$ is determined by its mean function $\mu(x)$ and covariance function $k(x,x')$. If $z(x)$ has a GP prior $GP(\mu(x), k(x,x'))$, then after observing models in $\mathcal{L}(t)$ at time t, the posterior distribution of $z(x)$ given $\{z(x)\}_{x\in\mathcal{L}(t)}$ is also a Gaussian Process $GP(\mu_t(x), k_t(x,x'))$. Here, posterior mean $\mu_t(x)$ and variance $k_t(x,x')$ can be computed analytically. We give the formulas in Supplemental Meterials (**Section A**).

EI function can also be computed analytically if $z(x)$ obeys Gaussian Process (whatever prior or posterior). The following lemma gives the expression.

Lemma 1. *Let $\Phi(x)$ denote cumulative distribution function (CDF) of standard normal distribution and $\phi(x)$ denote probability density function (PDF) of standard normal distribution. Also, let $\tau(x) = x\Phi(x) + \phi(x)$. Then, if $X \sim \mathcal{N}(\mu, \sigma^2)$, and $a \in \mathbb{R}$ is a constant, we have*

$$\mathbb{E}\Big[\max\big\{X - a, 0\big\}\Big] = \sigma\tau\Big(\frac{\mu - a}{\sigma}\Big).$$

This section ends by the detailed description of the proposed MM-GP-EI algorithm in Algorithm 1.

Discussion: How to Choose Prior Mean $\mu(x)$ and Prior Covariance $k(x,x')$ Prior mean $\mu(x)$ and prior covariance $k(x,x')$ are chosen according to the specific problem. They often characterize some properties of models in the problem, such as expected value of models and correlations among different models. In our multi-device multi-tenant example, the parameters of Gaussian process can be obtained from historical experiences and the correlation depends on two factors: the similarity of algorithms and the similarity of users' datasets. We following standard AutoML practice (used in Google Vizier or ease.ml) to construct the kernel matrix from historical runs.

5 MAIN RESULT

Before we introduce the main theoretical result, let us propose a new notation *Maximum Incremental Uncertainty* (**MIU**), which plays a key role in our theory.

5.1 Maximum Incremental Uncertainty

Suppose that K is the kernel matrix, that is, $K := [k(x,x')]_{(x,x'\in\mathcal{L})}$, where $k(x,x')$ is the kernel function and $\mathcal{L}$ is the set of all models as defined in Section **3.1**. So, K is an $|\mathcal{L}| \times |\mathcal{L}|$ positive semi-definite (covariance) matrix. Suppose S is a subset of $[|\mathcal{L}|] := \{1, 2, \cdots, |\mathcal{L}|\}$. Let K_S be a submatrix of K with columns and rows indexed by S.

We define the s-**MIU** score of matrix K ($1 \le s \le |\mathcal{L}|$) by

$$\mathbf{MIU}_s(K) := \max_{\substack{S'\subset S\subseteq[\mathcal{L}],\\ |S|=s,|S'|=s-1}} \begin{cases} \sqrt{\frac{\det(K_S)}{\det(K_{S'})}}, & \det(K_{S'})\neq 0;\\ 0, & \text{otherwise,}\end{cases}$$

where we define $\det(K_\varnothing) = 1$.

Let us understand the meaning of the notation **MIU**. Given $|\mathcal{L}|$ Gaussian random variables with covariance matrix $K \in \mathbb{R}^{|\mathcal{L}|\times|\mathcal{L}|}$, $\det(K_{S'})$ denotes the total quantity of uncertainty for all random variables in $S' \subset \mathcal{L}$. $\det(K_S)/\det(K_{S'})$ denotes the incremental quantity of uncertainty by adding one more random variable into S' to form S. If the added random variable can be linearly represented by random variables in S', the incremental uncertainty is zero. If the added random variable is independent to all variables in S, the incremental uncertainty is the variance of the added variable. Therefore, $\mathbf{MIU}_s(K)$ measures the *largest* incremental quantity of uncertainty from $s-1$ random variables to s random variables in $\mathcal{L}$.

Remark 2. Why do not use Information Gain? *People who are familiar with the concept of information gain (IG) may ask "IG is a commonly used metric to measure how much uncertainty reduced after observed a sample. Why not use it here?" Although IG and MIU essentially follow the same spirit, the IG metric is not suitable in our setup. Based on the mathematical definition of IG (see Lemma 5.3 in Srinivas et al. [2012]), it requires the observation noise of the sample to be* nonzero *to make it valid (otherwise it is infinity), which makes it inappropriate in the main target scenario in this paper. In our motivating example – cloud computing platform, the observation noise is usually considered to be zero, since no people run the same experiment twice. That motivates us to define a slightly different metric, (i.e., Maximum Incremental Uncertainty), to fix the non-observation-noise issue.*

5.2 Main Theorem

To simplify the analysis and result, we make the following assumption commonly used for analyzing EI [Bull, 2011] and GP-UCB [Srinivas et al., 2012].

Assumption 1. *Assume that*

- *there exists a constant R such that: $|z(x) - \mu_t(x)| \leq \sigma_t(x)R$, for any model $x \in \mathcal{L}$ and any $t \geq 0$;*
- $\sigma(x) \leq 1$.

Now we are ready to provide the upper bound for the regret defined in (2).

Theorem 2. *Let $\mathbf{MIU}(T,K) := \sum_{s=2}^{|\mathcal{L}(t)|} \mathbf{MIU}_s(K)$. Under Assumption 1, the regret of the output of Algorithm 1 up to time T admits the following upper bound*

$$\mathbf{Regret}_T \lesssim (\mathbf{MIU}(T,K) + M)\,\frac{N^2}{M}\bar{c}.$$

where $\bar{c} := \frac{1}{N}\sum_{i=1}^{N} c(x_i^*)$ is the average time cost of all optimal models, and $\lesssim$ means "less than equal to" up to a constant multiplier.

The proof of Theorem 2 can be found in the Supplemental Materials. To the best of our knowledge, this is the first bound for time sensitive regret. We offer the following general observations:

- (**convergence to optimum**) If the growth of $\mathbf{MIU}(T,K)$ with respect to T is $o(T)$, then average regret converges to zero, that is,

$$\frac{1}{T}\mathbf{Regret}_T \to 0.$$

In other words, the service provider will find the optimal model for each individual user.

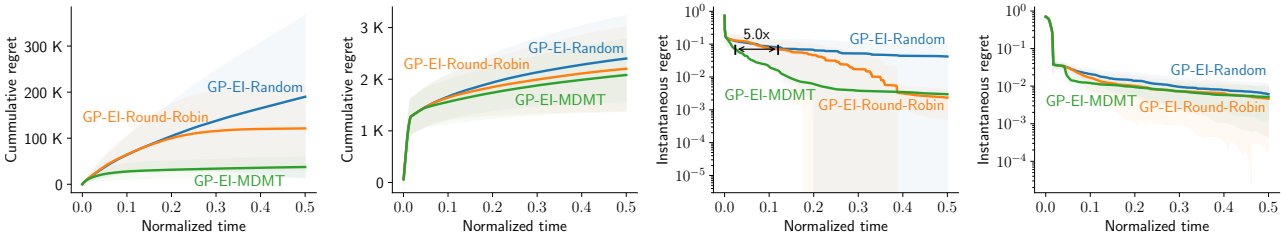
- (**nearly linear speedup**) When more and more devices are employed, that is, increasing M, then the regret will decrease roughly by a factor M as long as M is dominated by $\mathbf{MIU}(T,K)$.

Convergence Rate of the Average Regret. We consider the scenario where $M \ll \mathbf{MIU}(T,K)$ and $|\mathcal{L}(t)|$ increases linearly with respect to T. Then the growth of $\mathbf{MIU}(T,K)$ will dominate the convergence rate of the average regret. Note that $\mathbf{MIU}(T,K)$ is bounded by

$$\mathbf{MIU}(T,K) \leq \sum_{i\in\text{top } |\mathcal{L}(t)| \text{ elements in diag}(K)} \sqrt{K(i,i)}.$$

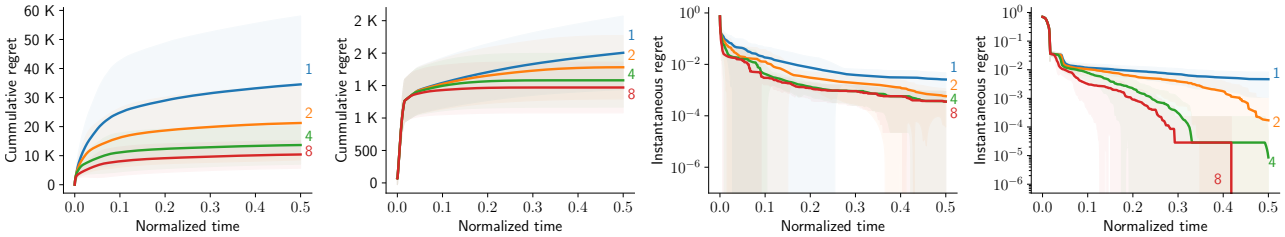
Discussion: Special Cases. Consider the following special cases:

- ($O(1/T)$ **rate**) The convergence rate for $\frac{1}{T}\mathbf{Regret}_T$ achieves $O(1/T)$, if $\mathbf{MIU}(T,K)$ is bounded, for example, all random variables (models) are linearly combination of a finite number of hidden Gaussian random variables.
- (**not converge**) If all models are independent, then K is a diagonal matrix, and $\mathbf{MIU}_s(K)$ is a constant, which means $\mathbf{MIU}(T,K)$ is linearly increased of T. In such a case, the regret is of order T, which implies no convergence for the average regret. This is plausible in that the algorithm gains no information from previous observations to decide next because of independence.
- ($O(1/T^{(1-\alpha)})$ **rate with** $\alpha \in (0,1)$) This rate can be achieved if $\mathbf{MIU}(T,K)$ grows with the rate $O(T^\alpha)$.



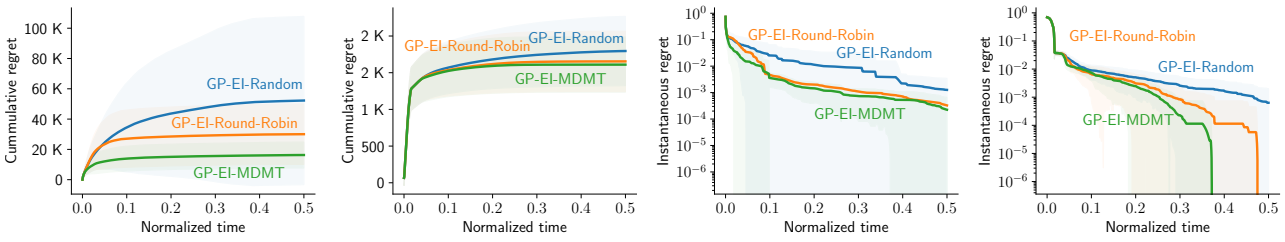
(a) Azure - 1 device (b) DeepLearning - 1 device (c) Azure - 1 device (d) DeepLearning - 1 device

Figure 2: Performance of Different Model Selection Algorithms with a Single Computation Device.



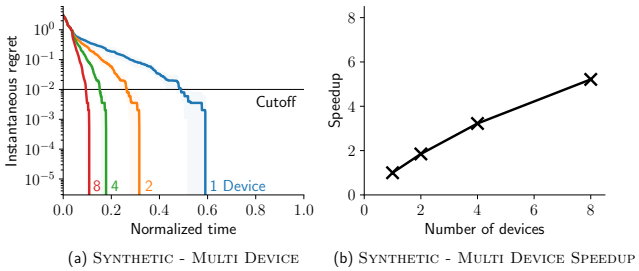
(a) Azure (b) DeepLearning (c) Azure (d) DeepLearning

Figure 3: The Impact of Multiple Devices on Our Approach.



(e) Azure - 4 devicess (f) DeepLearning - 4 devicess (g) Azure - 4 devicess (h) DeepLearning - 4 devicess

Figure 4: Performance of Different Model Selection Algorithms with Four Computation Devices.



(a) Synthetic - Multi Device (b) Synthetic - Multi Device Speedup

Figure 5: Speedup of using Multiple Devices for Our Approach on Synthetic Data

6 EXPERIMENTS

We validate the effectiveness of the multi-device, multi-tenant GP-EI algorithm.

6.1 Data Sets and Protocol

We use two datasets for our experiments, namely (1) DeepLearning and (2) Azure. Both datasets are from the ease.ml paper Li et al. [2018] in which the authors evaluate their single-device, multi-tenant GP-UCB algorithm. The DeepLearning dataset is collected from 22 users, each runs an image classification task. The system needs to select from 8 deep learning models, including NIN, GoogLeNet, ResNet-50, AlexNet, BNAlexNet, ResNet-18, VGG-16, and SqueezeNet. The Azure dataset is collected from 17 users, each runs a Kaggle competition. The system needs to select from 8 binary classifiers, including Averaged Perceptron, Bayes Point Machine, Boosted De-

cision Tree, Decision Forests, Decision Jungle, Logistic Regression, Neural Network, and SVM.

Protocol We run all experiments with the following protocol. In each run we randomly select 8 users which we will isolate and use to estimate the mean and the covariance matrix of the prior for the Gaussian process. We test different model selection algorithms using the remaining users.

For all model selection strategies, we warm start by training two fastest models for each user, and then switching to each specific automatic model selection algorithm:

- **GP-EI-Random:** Each user runs their own GP-EI model selection algorithm; the system chooses the next user to serve uniformly at random and trains one model for the selected user;
- **GP-EI-Round-Robin:** Each user runs their own GP-EI model selection algorithm; the system picks the next user to serve in a round robin manner;
- **GP-EI-MDMT:** Our proposed approach in which each user runs their own GP-EI model selection algorithm; the system picks the next user to serve using the MM-GP-EI algorithm we proposed.

Metrics We measure the performance of a model selection system in two ways: (1) Cumulative Regret $\mathbf{Regret}_T$ and (2) Instantaneous Regret: at time T, we calculate the average among all users of the difference between the best possible accuracy for each user and the current best accuracy the user gets. Intuitively, this measures the global "unhappiness" among all users at time T.

6.2 Single device experiments

We validate the hypothesis that, given a single device, multi-tenant GP-EI outperforms both round robin and random strategies for picking the next user to serve.

Figure 2 shows the result on both datasets. The colored region around the curve shows the 1σ confidence interval. On AZURE, our approach outperforms both round robin and random significantly — we reach the same instantaneous regret up to $5\times$ faster than round robin. This is because, by prioritizing different users with respect to their expected improvement, the global happiness of all users can increase faster than treating all users equally. On the other hand, for DEEPLEARNING, we do not observe a significant speedup for our approach. This is because the first two trials of models already give a reasonable quality. If we measure the standard deviation of the accuracy of models for each user, the average for AZURE is 0.12 while for DEEPLEARNING it is 0.04. This means that, once the system trains the first two initial models for each user in AZURE, there could be more potential performance gain still undiscovered among models that have not yet been sampled.

6.3 Multiple device experiments

We validate the hypothesis that using multiple devices speeds up our multi-device, multi-tenant model selection algorithm. Figure 3 shows the result of using multiple devices for our algorithm. We see that the more devices we use, the faster the instantaneous regret drops. In terms of speedup, since DEEPLEARNING has more users than AZURE (14 vs. 9), we see that the speedup is larger on DEEPLEARNING. The significant speedup of reaching instantaneous regret of 0.03 for AZURE is most probably due to the small number of users compared to the number of devices (9 vs. 8).

We now compare our approach against GP-EI-Round-Robin and GP-EI-Random when there are multiple devices available. Figure 4 shows the result. We see that, up to 4 devices (9 users in total), our approach outperforms round robin significantly on AZURE. When we use 8 devices for Azure, because there are only 9 users, both our approach and round robin achieve almost the same performance.

We also conduct an experiment using a synthetic dataset with 50 users and 50 models (Figure 5). We model the performance as a Gaussian Process and generate random samples independently for each user. The Gaussian Process has zero mean and a covariance matrix derived from the Matérn kernel with $\nu = 5/2$. Each generated sample is upwards in order to be non-negative. We run our approach on the same dataset while varying the number of devices. For each device count we repeat the experiment 5 times. To quantify speed gains we measure the average time it takes the instantaneous regret to hit a cutoff point of 0.01. We can observe that adding more devices makes the convergence time drop at a near-linear rate.

7 CONCLUSION

In this paper, we introduced a novel multi-device, multi-tenant algorithm using GP-EI to maximize the "global happiness" for all users, who share the same set of computing resources. We formulated the "global happiness" in terms of a cumulative regret and first time provided a theoretical upper bound for the time sensitive regret in the GP-EI framework. We evaluated our algorithm on two real-world datasets, which significantly outperforms the standard GP-EI serving users randomly or in a round robin fashion. Both our theoretical results and experiments show that our algorithm can provide near linear speedups when multiple devices are available.

Acknowledgements

Chen Yu and Ji Liu are in part supported by NSF CCF1718513, IBM faculty award, and NEC fellowship. Ce Zhang and the DS3Lab gratefully acknowledge the support from Mercedes-Benz Research & Development North America, MeteoSwiss, Oracle Labs, Swiss Data Science Center, Swisscom, Zurich Insurance, Chinese Scholarship Council, and the Department of Computer Science at ETH Zurich.

References

R. Arora, O. Dekel, and A. Tewari. Online bandit learning against an adaptive adversary: from regret to policy regret. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1747–1754, 2012.

J.-Y. Audibert and S. Bubeck. Minimax policies for adversarial and stochastic bandits. In *Conference on Learning Theory*, pages 217–226, 2009.

P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

S. Bubeck, N. Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

A. D. Bull. Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, 12(Oct):2879–2904, 2011.

E. A. Daxberger and B. K. H. Low. Distributed batch Gaussian process optimization. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 951–960, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

N. de Freitas, A. Smola, and M. Zoghi. Regret bounds for deterministic gaussian process bandits. *arXiv preprint arXiv:1203.2177*, 2012.

A. A. Deshmukh, U. Dogan, and C. Scott. Multi-task learning for contextual bandits. In *Advances in Neural Information Processing Systems*, pages 4851–4859, 2017.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter. Efficient and Robust Automated Machine Learning. In *NIPS*, pages 2962–2970.

V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220, 2012.

D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. E. Karro, and D. Sculley. Google Vizier: A Service for Black-Box Optimization. In *KDD*.

D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017.

GPyOpt. {GPyOpt}: A Bayesian Optimization framework in python. \url{http://github.com/SheffieldML/GPyOpt}, 2016.

E. Hazan et al. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2 (3-4):157–325, 2016.

R. Huang, M. M. Ajallooeian, C. Szepesvári, and M. Müller. Structured best arm identification with fixed confidence. In *International Conference on Algorithmic Learning Theory*, pages 593–616, 2017.

V. T. J. Mockus and A. Zilinskas. *Toward Global Optimization*, volume 2, chapter The application of Bayesian methods for seeking the extremum, pages 117–128. Elsevier, 1978.

D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. Pac subset selection in stochastic multi-armed bandits. In *International Conference on Machine Learning*, volume 12, pages 655–662, 2012.

K. Kandasamy, G. Dasarathy, J. B. Oliva, J. Schneider, and B. Poczos. Gaussian process bandit optimisation with multi-fidelity evaluations. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 992–1000. Curran Associates, Inc., 2016.

K. Kandasamy, G. Dasarathy, J. Schneider, and B. Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1799–1808, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Póczos. Parallelised bayesian optimisation via thompson sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 133–142, 2018.

T. Kathuria, A. Deshpande, and P. Kohli. Batched gaussian process bandit optimization via determinantal point processes. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4206–4214. Curran Associates, Inc., 2016.

E. Kaufmann and W. M. Koolen. Monte-carlo tree search by best arm identification. In *Advances in Neural Information Processing Systems*, pages 4904–4913, 2017.

L. Kocsis, C. Szepesvári, and J. Willemson. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1, 2006.

L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown. Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA. *Journal of Machine Learning Research*, (25):1–5.

T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

J. Langford and T. Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824, 2008.

T. Li, J. Zhong, J. Liu, W. Wu, and C. Zhang. Ease.ml: Towards multi-tenant resource sharing for machine learning workloads. *The Proceedings of the Very Large Database Endowment*, 2018.

G. Malkomes, C. Schaff, and R. Garnett. Bayesian optimization for automated model selection. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2900–2908. Curran Associates, Inc., 2016.

S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh. High dimensional bayesian optimization with elastic gaussian process. In *International Conference on Machine Learning*, pages 2883–2891, 2017.

I. O. Ryzhov. On the convergence rates of expected improvement methods. *Operations Research*, 64(6): 1515–1528, 2016.

J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

E. R. Sparks, A. Talwalkar, D. Haas, M. J. Franklin, M. I. Jordan, and T. Kraska. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing - SoCC '15*, pages 368–380, New York, New York, USA. ACM Press. ISBN 9781450336512. doi: 10.1145/2806777.2806945.

N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, 2012.

C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Auto-WEKA. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 847, New York, New York, USA. ACM Press. ISBN 9781450321747. doi: 10.1145/2487575.2487629.

J. Zhong, Y. Huang, and J. Liu. Asynchronous parallel empirical variance guided algorithms for the thresholding bandit problem. *arXiv preprint arXiv:1704.04567*, 2017.