On-Edge Multi-Task Transfer Learning: Model and Practice With Data-Driven Task Allocation

Qiong Chen, Zimu Zheng, Chuang Hu,
Dan Wang, *Senior Member, IEEE*, and Fangming Liu, *Senior Member, IEEE*

Abstract—On edge devices, data scarcity occurs as a common problem where transfer learning serves as a widely-suggested remedy. Nevertheless, transfer learning imposes heavy computation burden to the resource-constrained edge devices. Existing task allocation works usually assume all submitted tasks are equally important, leading to inefficient resource allocation at a task level when directly applied in Multi-task Transfer Learning (MTL). To address these issues, we first reveal that it is crucial to measure the impact of tasks on overall decision performance improvement and quantify *task importance*. We then show that task allocation with task importance for MTL (TATIM) is a variant of NP-complete Knapsack problem, where the complicated computation to solve this problem needs to be conducted repeatedly under varying contexts. To solve TATIM with high computational efficiency, we propose a Data-driven Cooperative Task Allocation (DCTA) approach. Finally, we evaluate the performance of DCTA by not only a trace-driven simulation, but also a new comprehensive real-world AIOps case study which bridges model and practice via a new architecture and main components design within AIOps system. Extensive experiments show that our DCTA reduces 3.24 times of processing time, and saves 48.4 percent energy consumption compared with the state-of-the-art when solving TATIM.

Index Terms—Edge computing, transfer learning, data-driven task allocation, real-world application

1 Introduction

Nowadays, computationally intensive machine-learning applications such as image recognition are becoming popular on resource-constrained edge devices (e.g., intelligent camera). While enjoying the merits of these applications, users are also frustrated when striking the balance between execution time and resource consumption on the edge. To address this problem, many task partitioning approaches have been proposed. Generally, an edge application is partitioned into a set of tasks which can be executed on the edge devices. For example, the video analytics application usually consists of several tasks (e.g., face detection and action classification), and allocates these tasks to multiple edge nodes to execute. Application partition and task allocation reduce the burden of a single edge device and jointly improve the performance of the application.

However, in major edge computing systems, we often face challenges in learning under data scarcity, due to either prohibitive cost (e.g., privacy concern, storage limitations, and networking costs), or inherent difficulty in obtaining required proper training samples with respect to the system complexity and uncertainty on the edge. Recently, *transfer learning* shows its effectiveness to tackle the data scarcity issue [1] and serves as a widely-suggested remedy for different industrial applications with insufficient samples, e.g., image recognition [2], speech analysis [3], disease diagnosis [4], medical informatics [5] and industrial operations (e.g., AIOps) [6].

In this paper, we focus on the *Multi-task Transfer Learning (MTL)* on the edge, where a machine-learning-based application can be divided into multiple machine-learning tasks, and each task can obtain the knowledge of some other tasks to improve its performance. It is well known that the machine-learning-based application is highly computation-intensive, while the computation resource of edge device is limited. Many efforts have been devoted to designing task allocation mechanisms to achieve various objectives, e.g., optimizing the makespan [7], throughput [8] or reliability [9] of the application. However, these frameworks focus on general parallel tasks in the centralized datacenter, where the computation capacity is assumed to be infinite in terms of constantly leasing of virtual machines.

In edge computing systems, it is sometimes hard to obtain a satisfactory result within time and resource limitations if we directly utilize existing frameworks for the cloud. Admittedly, existing task allocation studies have considered that different tasks may require different resources in edge computing systems in order to jointly improve the performance of the application [10], [11], [12]. They are usually designed for general machine learning and typically assume that all tasks contribute identically to overall performance improvement of the application. However, in MTL, tasks belonging to the same machine-learning-based application usually

- *Q. Chen and F. Liu are with the National Engineering Research Center for Big Data Technology and System, Key Laboratory of Services Computing Technology and System, Ministry of Education, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, China. E-mail: {qiongchen, fmliu}@hust.edu.cn.*
- *Z. Zheng is with Edge Cloud Innovation Lab, Technical Innovation Department, Cloud BU, Huawei Technologies Co., Ltd., Shenzhen 518129, China. E-mail: zimu.zheng@huawei.com.*
- *C. Hu and D. Wang are with the Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong. E-mail: {cschu, csdwang}@comp.polyu.edu.hk.*

Manuscript received 14 July 2019; revised 18 Nov. 2019; accepted 18 Dec. 2019. Date of publication 25 Dec. 2019; date of current version 29 Jan. 2020. (Corresponding author: Fangming Liu.)
Recommended for acceptance by B. Veeravalli.
Digital Object Identifier no. 10.1109/TPDS.2019.2962435

have different potential for improving the application's overall performance. Directly applying these techniques leads to inefficient resource utilization at a task level under MTL in edge computing systems.

To solve the above inefficiency issue for multiple-task allocation in edge computing systems, the key is that more important tasks, which have the higher potential for improving the application's overall decision performance, should be allocated to more powerful edge devices for priority execution under time limits. Recently, Geng *et al*. also considered the priority of tasks by leveraging the dependency of tasks in task allocation [13]. In that study, the task dependency is predefined and remains fixed over time, e.g., installing Hadoop before Spark. However, due to the complex nature of machine-learning tasks, variables such as environmental conditions and model configurations are likely to change over time. The dependency of machine-learning tasks is dynamic and usually not available before learning. Directly applying the current allocation mechanism can easily result in significant overall application performance degradation for MTL on the edge.

Instead of assuming that all tasks contribute identically to the application's overall decision performance improvement and conducting the time-dynamic task allocation on the edge, our idea is to leverage machine learning techniques to capture the correlated and collective potential improvement of multiple tasks. Accordingly, we propose a Data-driven Cooperative Task Allocation (DCTA) mechanism to maximize the application's overall decision performance among multiple tasks on the edge. We also conduct a new comprehensive case study under the real-world industrial operation (e.g., AIOps) scenario, where MTL is necessary due to the data scarcity on edge devices.

Challenges and Solutions. In designing DCTA, we have to overcome three following major technical challenges.

First, the metric of tasks impact on overall decision performance improvement remains unknown in current studies. To tackle the challenge, we propose a metric of *task importance*, which is to measure the overall performance degradation when the measured task is not conducted in MTL. We also observe the long-tail property of task importance, i.e., only a few tasks are important, which serves as a key metric to guide task allocation and facilitate resource saving from less important tasks. We formally define the TATIM problem of task allocation with task importance for MTL on the edge.

Second, the TATIM problem is challenging due to not only its computation complexity (i.e., NP-complete) but also the varying contexts (i.e., dynamic task importance) on the edge. We first prove that TATIM is a variant of Knapsack problem and thus NP-complete. We then show that the task importance is difficult to capture, due to varying environmental conditions and configurations. Therefore, the complicated computation to solve this problem needs to be conducted repeatedly under varying contexts on the edge. To enhance the computational efficiency, we propose a data-driven task allocation mechanism based on reinforcement learning.

Third, applying the machine learning technique to solve the TATIM problem introduces a trade-off between accuracy and cost. On one hand, an accurate data-driven model requires a huge amount of expensive local data on real-world operations. On the other hand, merely using general data from simulation helps to reduce the amount of local data needed but leads to low accuracy. To tackle the challenge, we propose a cooperative learning mechanism to reduce the amount of data needed to generate a reliable data-driven model, by leveraging both general simulated data and local real-world data.

We implement DCTA as a task allocation approach within a data-driven building management system. We also evaluate various distinct task allocation approaches by not only a trace-driven simulation, but also a new comprehensive real-world AIOps case study which bridges model and practice via a new architecture and main components design within AIOps system. Extensive experiments show that our DCTA reduces 3.24 times of processing time, and saves 48.4 percent energy consumption when solving TATIM compared to the state-of-the-art.

The rest of the paper is organized as follows. In Section 2, differing from our preliminary work [14], we reorganize all the notations and observations on task importance for better understanding. In Section 3, we introduce the data-driven approach for task allocation, by leveraging both cluster reinforcement learning and support vector machine. In Section 4, we conduct trace-driven simulations to evaluate the performance of the proposed DCTA mechanism. In Section 5, we add a new comprehensive case study on AIOps for our DCTA mechanism to bridge model and practice. Specifically, we first elaborate the background of AIOps system for better understanding, and exhaustively analyze the motivation of applying DCTA mechanism to the AIOps system. We then further analyze how to apply the DCTA mechanism by proposing a new architecture and main components design within AIOps system. Extensive experiments are complemented to demonstrate the superiority of AIOps system integrating our DCTA mechanism. Section 6 discusses related work and Section 7 analyzes some future work and possible improvements. At last, we conclude this paper in Section 8.

2 Background and Problem Definition of Task Allocation With Task Importance

In this section, we first introduce the background of Multi-task Transfer Learning (MTL). We then give a formal definition of task importance. We also observe the long-tail property of task importance and the potential of leveraging task importance for task allocation in MTL. With these notations, we formally define the problem of task allocation with task importance for MTL.

2.1 Background of Multi-Task Transfer Learning (MTL)

In this paper, we study the issue of *Multi-task Transfer Learning (MTL)* on the edge, where varying tasks together can facilitate better decision performance. It basically reuses parameters or training samples of source tasks to support target tasks, e.g., which are lack of training data. The term *task* is defined as a set of data, label and its corresponding learning model for a predefined context. For example, for a self-driving car on the road, the detection of each type of object, e.g., neighboring-car, traffic-sign, or pedestrian detection, can be modeled separately as a task. Another example is to take the coefficient of

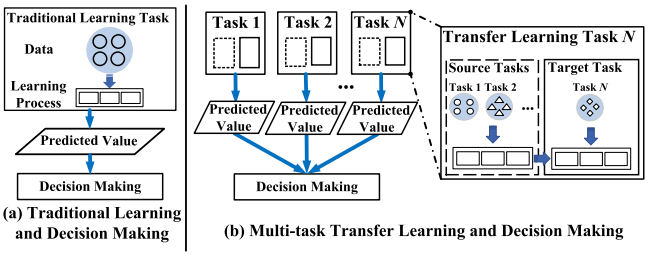


Fig. 1. Decision making with (a) traditional learning and (b) transfer learning.

performance (COP) prediction of a chiller for one particular operation as a task [15]. The process is shown in Fig. 1.

The benefits of multiple tasks come in mainly two ways. First, similar tasks can transfer their knowledge between each other during the training process, which reduces the negative effect of data scarcity, especially on the edge. Second, in the real-world scenario, it is common to make the final decision by aggregating the output of multiple tasks. Maintaining the high performance of all these tasks contribute to the final aggregated decision performance. Again in the example of a self-driving car, the final driving operation of the car is conducted based on the result of multiple data-driven tasks, e.g., the neighboring-car, traffic-sign, and pedestrian detection.

The Computation Challenge. However, the current MTL systems are way too computationally complicated for edge devices. The reason is twofold: 1) Each task needs to be learned individually from scratch, where siloing tasks make training a new task or a comprehensive perception system a Sisyphean challenge; 2) To avoid data-driven task model being out-of-date and leverage the latest accumulated data as effectively as possible, MTL practitioners retrain their models repeatedly to get the final model with the best quality, including to explore feature representation [16], [17], [18], adjust structures of task relationship [19], [20], [21] and tune hyper-parameters [22]. For better understanding, a formal formulation of MTL tasks on the edge is available in the following Section 2.4.

2.2 Notations of Task Importance

Confronted with the computational challenge of MTL, we aim to allocate tasks for more efficient MTL on the edge. When allocating tasks, current studies usually assume that all machine-learning tasks are equally important so that resources should be allocated to ensure the accuracy of all these tasks.

However, tasks are not always related to the current context, and thus not equally important. At a specific period of time, e.g., within one hour, the number of highly important tasks are likely to be of a minor, compared with the number of all possible tasks. For example, for a self-driving car on the high way, neighboring car detection can be much more related and important compared with most tasks like pedestrian detection which are more important in a downtown area.[1]

In this part, before studying *task importance*, we first formally define it and its related notations. The key notations in this paper are also listed in Table 1 for ease of reference. A further experiment on task importance is available after the definition.

1. As a further demonstration, a real-world experiment and the corresponding observation are also available at the end of the subsection.

TABLE 1
List of Key Notations

Notation	Description
$\boldsymbol{J}$	Set of tasks where $\boldsymbol{J} = \{j\}$
$\boldsymbol{P}$	Set of edge devices where $\boldsymbol{P} = \{p\}$
$\mathcal{I}_j$	The importance of task j
$\mathcal{H}(\cdot)$	The merit function indicates the ability to provide credible decision performance
$\mathcal{D}(\cdot)$	The decision-making function indicates the best operation
D	The ideal performance
$u_{j,p}$	Whether the task j is assigned to processor p (=1) or not (=0)
t_j	The execution time of task j
v_j	The resource (e.g., battery) consumption of task j
T	The maximum time limits to conduct the decision
V_p	The maximum resource capacity of processor p
θ_j	The model parameters of task j
$L_j(\cdot)$	The learning loss of task j
$\mathbf{u}$	The task-allocation matrix where $\mathbf{u} = [u_{j,p}]$

Definition 1 (Task Importance). *Given a task set $\boldsymbol{J} = \{j\}$ which consists of a series of tasks, the importance of task j is*

$$\mathcal{I}_j = \mathcal{H}(\boldsymbol{J};\boldsymbol{\theta}) - \mathcal{H}(\boldsymbol{J}\backslash\{j\};\boldsymbol{\theta}\backslash\{\theta_j\}), \tag{1}$$

where a learning task is denoted by $j \in \mathbb{N}^+$; θ_j denotes the model parameters of task j and $\boldsymbol{\theta} = \{\theta_j\}$ denotes its vector; merit function $\mathcal{H}(\cdot)$ outputs the final potential performance improvement; $\boldsymbol{J}$ denotes the entire task set.

Thus, given model parameters $\boldsymbol{\theta}$, the task importance $\mathcal{I}_j$ can be updated using the merit function $\mathcal{H}(\cdot)$. Such a function indicates the ability to provide credible decision performance (e.g., energy saving) and outputs a value called *overall merit*, which is formally defined as below.

Definition 2 (Overall Merit). *Given the task set $\boldsymbol{J}$ and the ideal performance of final decisions D, the overall merit is defined as the similarity with the ideal performance, i.e.,*

$$OM = \mathcal{H}(\boldsymbol{J};\boldsymbol{\theta}) = 1 - \frac{|D - \mathcal{D}(\boldsymbol{J};\boldsymbol{\theta})|}{D}, \tag{2}$$

where $\mathcal{D}(\cdot)$ denotes a decision-making function given model parameters, and D denotes the ideal performance which can usually be collected after final optimization, i.e., collected manually or automatically by leveraging historical samples.

In general, the historical data records the descriptor of contexts/ scenarios, requirement, historical operations, and its results. Such information helps us to define D. For example, in the case of a self-driving car, in order to ensure the car arrive at the destination safely, it will conduct a series of decision actions where the least time-consuming situation can be regarded as the ideal performance.

Such a decision-making function $\mathcal{D}(\cdot)$ is intrinsically solving an optimization problem finding the best action according to parameters, which can be set once given the scenario. For example, in the case of a self-driving car, a possible

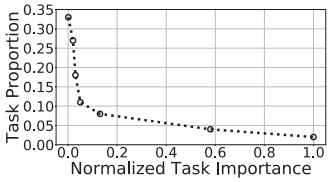


Fig. 2. The non-uniform (i.e., long-tail) distribution of normalized task importance in MTL.

decision-making function is to find an action which minimizes the probability of accident while ensures the car should be able to arrive at the destination under time limitations. For interested readers, a more concrete implementation of $\mathcal{D}(\cdot)$ is also available in Section 5.

2.3 Observations on Task Importance

We have introduced related concepts of task importance. We next justify the motivation of using task importance by further observations.

We first plot the distribution of task importance in Fig. 2, based on a real-world MTL dataset released in [15]. In there are totally 50 data-driven tasks for cooling operations running across four years in three buildings. We observe a long-tail property of task importance, i.e., merely 12.72 percent of tasks have a high contribution of over 80 percent to the overall merit. We say that a task is *unimportant* when its task importance is critically low compared with others, e.g., below 0.05 percent. We therefore have such an observation.

Observation 1. In MTL, unimportant tasks exist; The importance of tasks obeys a long-tail distribution.

This observation reveals the non-uniform distribution of task importance in the real-world environment which motivates us to break the common assumption of modern MTL. Results in a recent CVPR paper also confirm such an observation [23]. The unimportant (e.g., redundant or noisy) tasks can be the result of 1) insufficient training samples on the edge, and 2) mismatch of context and submitted tasks in practical scenarios. It also indicates the potential of speeding up MTL from those unimportant tasks.

In the machine learning community, current MTL systems usually conduct tasks in the order of time stamps, where these time-ordered tasks are of arbitrary importance. Thus, the current execution sequence can be regarded as random, e.g., normally distributed, in terms of task importance. When there are limitations on resource and execution time for MTL tasks, the current approach can suffer from lower overall merit.

We conduct experiment on the MTL dataset mentioned above [15], where the decision objective of MTL is to control

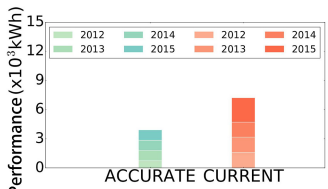


Fig. 3. The decision performance (i.e., energy consumption) with ACCURATE and CURRENT schemes.

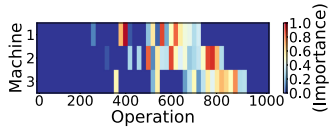


Fig. 4. Average task importance for different types of machines and operations.

the Chiller AIOps system to minimize the energy consumption for cooling, which refers to the decision performance. Fig. 3 shows the result by conducting MTL tasks in the order of task importance (called *ACCURATE* scheme), compared with the order of time with random task importance (called *CURRENT* scheme) under execution time limitations. Such an ACCURATE scheme can be obtained by computing task importance using historical data (Section 2.2), and can be regarded as ground truth. Base on the obtained accurate task importance, we can find the best task allocation strategy. For interested readers, a detailed optimization process is also available in Section 5.2.

Stacked bars on the left indicate the performance with the ACCURATE scheme, whereas the right show the CURRENT scheme using random task allocation. We see that the ACCURATE scheme considering task importance could have resulted in an average of over 45.68 percent potential improvement in terms of the overall merit. These results demonstrate that there is significant room to improve the overall merit when using a more accurate and robust scheme of task allocation. We summarize the observation as below.

Observation 2. Overall merit with MTL can be improved by task allocation according to task importance.

However, the task importance may not be always directly available for run-time usage. The above experiment is based on historical data so that we are able to compute the task importance after a task is executed. For run-time usage, we often need to know the task importance in advance, i.e., before a task is conducted. A natural question is whether the task importance is easy to predict, e.g., a fixed or stable value. Based on the above MTL dataset, we also conduct two experiments as more-detailed distribution studies showing how the importance fluctuates over operations under different industrial demands and conditions.

We first plot the average task importance as a function of different operations in terms of different types of machines in Fig. 4. We pick the first regular machine for example. It can be seen that these machines often operate at a small portion of operations, and the importance fluctuates somewhat randomly. At the same time, for the same types of machines, we plot in Fig. 5, the variation in their task importance under different operations, and note that there is a large fluctuation even for a given operation. This is because the task importance in practice is highly dependent on a variety

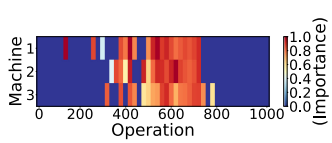


Fig. 5. Task importance variation for the same types of machines and operations.

of factors like environmental conditions and configurations. Such factors are referred as the term *context* in this paper. We therefore have such an observation.

Observation 3. Task importance fluctuates markedly over varying contexts with MTL in terms of average and variance.

This observation reveals that the time-dynamic task importance changes in varying contexts [24], [25], e.g., with different external factors (like environmental conditions and dynamic industrial demands) and internal factors (like machine configurations and response). For example, in the case of self-driving car, the *context* contains the following specific factors such as visual observations, physical information, weather, traffic conditions and etc. These factors are exceedingly difficult to capture within an analytical model. Facing such a high variance of task importance situation, natural thinking of modeling task importance using synthetic models easily suffers from low accuracy.

2.4 Problem of Task Allocation With Task Importance for MTL

Based on the above notations and observations, the intuition behind this paper is that we should allocate more important tasks to more powerful edge devices (e.g., edge server) to optimize the final decision. Here we say an edge device is more powerful which refers to it's processing speed or frequency is faster. We aim to leverage task importance to facilitate task allocation for MTL tasks on the edge, with an emphasis of time limits.

We start by formally defining the conception of *task allocation* and *MTL tasks on the edge*, where the former consists of the task placement and resource allocation. Considering machine-learning tasks are usually highly computation intensive, resource-constrained edge devices can barely handle multiple tasks in parallel. Therefore, we assume that, at a certain time, a *task* occupies the whole CPU computing resource under execution.

Definition 3 (Task Allocation). *Given an edge device set $\boldsymbol{P} = \{p\}$ which consists of a series of edge devices, the task allocation over $\boldsymbol{P}$ is a binary variable $u_{j,p}$, i.e.,*

$$u_{j,p} = \begin{cases} 1, & if\ task\ j\ is\ assigned\ to\ edge\ device\ p \\ 0, & otherwise, \end{cases}$$

where an edge device is denoted by $p \in \mathbb{N}^+$.

Since each task is indivisible and must be assigned to exactly one edge device, we have the following constraint:

$$\sum_{p \in \boldsymbol{P}} u_{j,p} = 1,\ \forall j \in \boldsymbol{J}. \tag{3}$$

Considering edge devices are usually resource-constrained and discrete, we classify resources into two categories, i.e., execution-related and basic requirements. The former refers to CPU computing resources, whereas the latter refers to battery or storage resources. Therefore, the CPU execution time and basically resource requirements of all tasks assigned to edge device p should satisfy the following constraints:

$$\sum_{j \in \boldsymbol{J}} t_j \cdot u_{j,p} \le T, \quad \forall p \in \boldsymbol{P}, \tag{4}$$

$$\sum_{j \in \boldsymbol{J}} v_j \cdot u_{j,p} \le V_p, \quad \forall p \in \boldsymbol{P}, \tag{5}$$

where t_j denotes the execution time of task j; T denotes the time limitations; v_j denotes the resource required for task j; V_p denotes the resource capacity of edge device p.

The objective of traditional MTL is to minimize the collective loss of all tasks. We study the modeling and define the MTL tasks specific to the edge computing scenario for better understanding.

Definition 4 (MTL Tasks on the Edge). *Given task importance $\mathcal{I}_j$, the execution time and resource limitations of Eqs. (3) - (5), an on-edge MTL tasks aims to obtain θ by*

$$\boldsymbol{\theta} = \arg\min \sum_{j \in \boldsymbol{J}} \sum_{p \in \boldsymbol{P}} \mathcal{I}_j \cdot L_j(\theta_j) \cdot u_{j,p},\ s.t.\ Eq.(3) - (5),$$

where $L_j(\theta_j)$ denotes the learning loss of task j, e.g., prediction error and regularization terms.

Based on the above definitions, we formally define the problem of task allocation with task importance for MTL on the edge (TATIM Problem) as below.

Definition 5 (TATIM Problem). *Given the execution time and resource limitations, a TATIM problem is to obtain $\boldsymbol{u}$ by*

$$\max_{\mathbf{u}} \sum_{j \in \boldsymbol{J}} \sum_{p \in \boldsymbol{P}} \mathcal{I}_j \cdot u_{j,p},\ \text{s.t. } Eq.\ (3) - (5),$$

where $\mathbf{u} = [u_{j,p}]$ denotes the task-allocation matrix; $\mathcal{I}_j$ can be computed given $\boldsymbol{\theta}$ from Definition 4 and $\boldsymbol{J}$ using Eqs. (1)−(2).

We found that the TATIM problem under the execution time and resource limitations is in fact a 0-1 Knapsack problem, which is in general NP-complete.

Theorem 1. *Task allocation problem with task importance is a 0-1 multiply-constrained multiple Knapsack problem.*

For interested readers, the proof of Theorem 1 can be found in our conference paper [14].

3 Data-Driven Approach for Task Allocation

As shown in the previous section, when we introducing the time-varying task importance $\mathcal{I}$, task allocation becomes a TATIM problem which is challenging as an NP-complete problem twofold.

First, the complexity introduced by task importance is the reason why we adopt a reinforcement learning (RL) model. We leverage data-driven methods in order to reduce the time needed to solve the origin NP-complete Knapsack problem. Specifically, in the data-driven RL method, we integrate task importance into the environment modeling of RL.

Second, because the task importance is time-varying, an RL model cannot simply be applied. In the first part, we propose a clustered reinforcement learning (CRL) model that makes decisions based on how observations of the environment relate to those previously seen. In the second part, because the CRL model can confront with quite a few unseen environments, we further propose a Support Vector Machine (SVM) model to predict the task importance and dynamically adjust CRL model decisions based on real-time data.

In a brief summary, the reason for using data-driven technique for TATIM with task importance is because it shows its effectiveness for complicated problems in time-varying environments, including Intelligent logistics [26], Autonomous Mobility-on-Demand system [27], and Human-level game control [28]. Basically, data-driven techniques are particularly helpful for solving complicated problems repeatedly with varying parameters, because they not only help to model and reduce the environmental randomness in multi-task scenarios but also help to significantly enhance the computational efficiency due to the fast inference phase when the solution is needed.[2]

Formally, given a task set J and the corresponding historical feature space $\mathcal{X}$, we are to develop a data-driven task allocation scheme with a loss function $\mathcal{L}(\cdot)$ which maximize the overall decision performance of the task allocation, i.e.,

$$\mathbf{u} \leftarrow \mathcal{F}(J, \mathcal{X}).$$

3.1 The Clustered Reinforcement Learning (CRL) Model

Next, we consider the proper approach to solving the TATIM problem. First, in the previous section, we have proved that the TATIM problem is in fact a Knapsack problem and therefore NP-complete. RL is widely suggested to efficiently solve such problems [27], [28]. Second, decisions made by industrial systems can be highly repetitive, thus generating an abundance of training data to support complicated data-driven model. Based on the two reasons, we applied the well-known RL to solve the TATIM problem.

In general, the RL works like this: at each decision epoch, the agent will make a decision based on the current state of the environment. Once the decision is made, a reward would be provided to the agent and the state of the environment would be updated for making future decisions. The agent tries to maximize the cumulative rewards over time. With RL, our TATIM problem is optimized in a Markov Decision Process (MDP), which is a five-tuple: $< \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \lambda >$, where $\mathcal{S}$ denotes the set of states; $\mathcal{A}$ denotes the set of actions; $\mathcal{P}$ denotes the transition probability distribution; r denotes the reward function and $\lambda \in [0, 1]$ denotes the discount factor for future rewards. Note that different optimization problems have quite different objectives, constraints, and variables. To adopt our TATIM problem, the different components of RL needs to be specially designed. The detailed design of these components in RL and MDP will be discussed next.

Environment-Dynamic Task Allocation. However, RL should not be directly applied in our scenario, where the environment is diverse over time and existing RL approaches usually assume a fixed environment.

1) Novel Problem of Environment-dynamic Knapsacks. In TATIM, the task importance is critical for environment modeling and thus also important for RL. As we known, the knowledge learned by the decision of an agent is rewarded according to the environment. Once the task importance and the corresponding environment is not close to reality, the decision made by the agent will lead to poor performance.

2. Though the training phase may be long, it merely needs to be conducted once in advance.

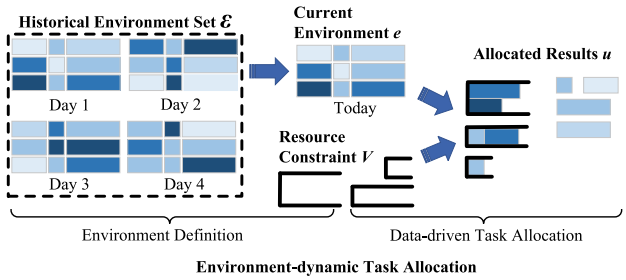


Fig. 6. The illustration of environment-dynamic task allocation.

However, due to the varying scenarios in MTL, the environment matrix of RL usually changes over time in reality. Recall the previous example where a self-driving car on the highway and pedestrians usually do not occur, the task of pedestrians detection is less important compared to other tasks. Nevertheless, when driving around the school, pedestrians are particularly frequent which makes the task of pedestrians detection more important. Therefore, we see that the environment is clearly diverse in different scenarios, especially when the task importance is encoded in the environment of RL.[3]

In this regard, directly leveraging the RL model can easily mismatch the environment and submitted less important tasks, which leads to poor decision performance [24], [29]. We also conduct an experiment to demonstrate the negative impact. It shows a 46.28 percent reduction of performance when the environment is not accurate using existing RL.

To this end, we realize that our TATIM problem can be regarded as a novel variant of the Knapsack problem. It is even more challenging than the Multiply-constrained Multiple Knapsack Problem proved in the previous section. This time, additionally, the item value (i.e., task importance) can be changed randomly over time, instead of being fixed in the traditional Knapsack problem.

2) Clustered Approach for Environment Definition. Accordingly, to solve the TATIM problem, we are to learn the current environment. Our idea is that the more similar historical days, the more similar the environment is. Such similarity can be measured by comparing the current scenarios and configuration settings, e.g., sensing data, of the predicting day and the historical days.

The overall process is illustrated in Fig. 6, which consists of two parts, i.e., environment definition and data-driven task allocation. In the figure, different days represent different environments, and the darkness of each color represents the different task importance. Through the analysis of historical data, we establish an environment data set, i.e., historical environment $\mathcal{E}$. We define the historical environment $\mathcal{E}$ as the collection of environment e, i.e.,

$$\mathcal{E} = [e_1, e_j, \ldots, e_{N'}], \quad \forall j \in [1, 2, \ldots, N'],$$

where e_j denotes the corresponding environment.

3. Even in the same scenario, the environment can change over time, due to the accumulating size of training data and the overwritten when the storage is insufficient. Experiments in Section 2.3 also indicates the fluctuation between historical and current task importance.

Through environment definition that we can find a similar environment e by clustering algorithms such as *k Nearest Neighbors* (kNN), i.e.,

$$e = kNN(\mathcal{E}, Z),$$

where Z denotes the sensing data. We then can make data-driven task allocation based on the clustered environment under the execution time and resource constraints.

Clustered Reinforcement Learning for Environment-Dynamic Task Allocation. Next, we propose key designs of our approach, i.e., the environment modeling, state space, action space, reward function, and optimization, which should be specified based on our TATIM problem.

1) Environment. A key component in the RL model is the *environment*, which is everything outside the agent, and changes its state due to the action of the agent, and gives the agent corresponding rewards. For an RL predictor, the environment can be described as a matrix e which is a map of the agent, e.g., Maze problem. More specifically, one dimension represents the subject types (e.g., neighboring car detection, traffic sign detection, and pedestrian detection), and the other represents the available processors (CPU processor, GPU processor, sensors). The elements of the matrix can be viewed as a data-driven task. It is formulated as follows:

$$e = [I_j \times V_p]_{N\times M}, \quad \forall I_j, V_p \in \mathbb{R},$$

where I_j denotes the corresponding task importance and V_p denotes the corresponding processor capacity.

2) State space. We represent the state, which is the current task selection of the system. Specifically, the state is defined by a matrix $\mathcal{S}$ and the element of each position can be 0 or 1. Note that 1 represents the task is selected, otherwise, it is not selected, which is formulated as follows:

$$\mathcal{S} = [s_{ij}]_{N\times M}, \quad \forall s_{ij} \in \{0, 1\}.$$

Such a fixed state representation indicates that it can be conveniently applied as an input to a neural network.

3) Action space. At each point in time, the scheduler may want to select any subset of the $N \times M$ tasks. But this requires a large action space of size $2^{N\times M}$ leading to unbearable computation to learn on the edge. We keep the action space small using a trick: we allow the agent to execute merely one action in each time step. The action space is given by $\{1, 2, \ldots, M\}$, where $a = j$ means to conduct the jth task for the current processor in the current time step. Hence, the action space is defined as follows:

$$\mathcal{A} = \{a | a \in \{1, 2, \ldots, M\}\}.$$

In this way, we can greatly speed up our learning rate while keeping the action space linear in M.

4) Reward Function. We craft the reward signal to guide the agent towards desired solutions for our objective: maximize overall task importance. Specifically, we set the reward at each time step to $\sum_{j\in \boldsymbol{J}} I_j$ only if the agent reaches the terminal state (i.e., all tasks in the current system are assigned accordingly), where $\boldsymbol{J}$ is the set of tasks currently in the system. Otherwise, the reward was set to 0. Hence

$$r(t) = \begin{cases} \sum_{j\in \boldsymbol{J}} I_j, & \text{if the agent reaches the terminal state} \\ 0, & \text{otherwise.} \end{cases}$$

It is worth noting that the agent is set to not receive any reward for intermediate decisions during a time step, which is well-suited to apply to our real-world decision objectives.

5) Optimization. With the above key elements, we leverage Deep Q-learning $Q(s, a; \theta, \boldsymbol{J})$ [30], where θ denotes the adjustable parameter vector of neural networks. It estimates the value of executing an action a from a given state s. Formally, given the feature space $\mathcal{X}$ which consist of the environment e and the initial state s_0, we have

$$\mathbf{u} \leftarrow \mathcal{F}_1(\boldsymbol{J}, \mathcal{X}) = \mathcal{F}_1(\boldsymbol{J}, (e, s_0)) = Q(s, a; \theta, \boldsymbol{J}). \tag{6}$$

Based on the above design, we propose the Clustered Reinforcement Learning (CRL) approach, as shown in Algorithm 1.

Algorithm 1. Clustered Reinforcement Learning (CRL)

Training Phase:
1: $\mathcal{E} \leftarrow$ historical environment $s_0 \leftarrow$ initial state $Z \leftarrow$ current scenarios and configuration settings.
2: $e \leftarrow$ *EnvironmentDefinition*$(\mathcal{E}, Z)$ ▷ Find similar environment.
3: **while** not yet reach the terminal state s_N **do**
4: $\mathcal{L}(s, a|\theta) \leftarrow (r + \max_a Q(s', a|\theta) - Q(s, a|\theta))^2$ ▷ Update DNN parameters θ.
5: **end while**
6: $\boldsymbol{\theta^*} \leftarrow \arg\min \mathcal{L}(s, a|\theta)$ ▷ Obtain optimal parameter θ.
7: **return** $e,\ s_0,\ \boldsymbol{\theta^*}$
Prediction Phase:
8: $(e,\ s_0,\ \boldsymbol{\theta^*}) \leftarrow$ initialization using the return value of the training phase.
9: $\mathbf{u} \leftarrow \mathcal{F}_1((e, s_0); \boldsymbol{\theta^*})$ ▷ Make task allocation prediction.
10: **return** $\mathbf{u}$

3.2 The Cooperative Learning Model Based on CRL

However, the CRL model should not be directly applied. In our scenario, the environment is diverse over time. Although we can find similar environments in the historical environment through simple clustering methods, there is a risk that the environment is still not closed to the real environment. That is especially true for edge devices without too much data, whereas the RL model can confront with quite a few unseen environments and it requires much environment observations to cover all possible situations.

In this regard, directly leveraging the CRL model can still mismatch the environment and submitted less important tasks, which leads to poor decision performance [24]. We also conduct an experiment to demonstrate the negative impact. Based on our CRL model, when the environment is not accurate, it leads to a 28.84 percent reduction of performance.

A Cooperative Learning Approach. To tackle the challenge, our idea is to leverage runtime data to adjust the decision of the CRL model.

Accordingly, we propose a cooperative learning approach as shown in Fig. 7, which is especially well-suited to solve this problem. The proposed cooperative learning approach contains two components: 1) a CRL predictor with a huge

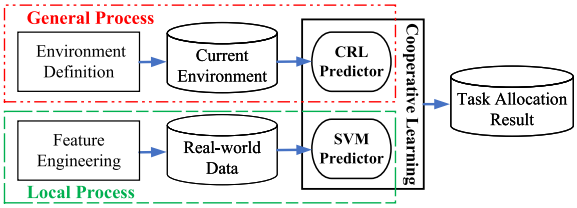


Fig. 7. Framework of cooperative learning for task allocation.

environment definition data, and 2) an SVM predictor with few real-world data. Formally, let $\mathcal{C}$ and $\mathcal{R}$ be the feature spaces of the environment definition data, i.e., $\mathcal{C} = \{(e, s_0)\}$, and real-world data, respectively. Let $\mathcal{F}(\cdot)$ denotes our cooperative learning model, which can be represented more specifically as

$$\mathcal{F}(\boldsymbol{J}, \mathcal{X}) = \mathcal{F}(\boldsymbol{J}, (\mathcal{C}, \mathcal{R})) = w_1 \mathcal{F}_1(\boldsymbol{J}, \mathcal{C}) + w_2 \mathcal{F}_2(\boldsymbol{J}, \mathcal{R}), \tag{7}$$

where $\mathcal{F}_1(\cdot)$ and $\mathcal{F}_2(\cdot)$ denote the CRL predictor and SVM predictor; w_1 and w_2 denote the weight of the corresponding model results, respectively. In addition, the task-allocation matrix $\mathbf{u}$ is outputted by our cooperative learning model $\mathcal{F}(\cdot)$, i.e., $\mathbf{u} \leftarrow \mathcal{F}(\boldsymbol{J}, (\mathcal{C}, \mathcal{R}))$.

As for the SVM predictor, we compare several state-of-the-art models of SVM, AdaBoost, and Random Forest. We select SVM because of its highest accuracy. Formally, given the target tasks feature values $\mathcal{X}$, our objective is to develop an SVM predictor $\mathcal{F}_2(\cdot)$ which infers the target tasks allocation $\mathbf{u}$. This can be formulated as follows:

$$\mathbf{u} \leftarrow \mathcal{F}_2(\boldsymbol{J}, \mathcal{X}) = SVM(\mathcal{X}; w, \boldsymbol{J}), \tag{8}$$

where w denotes its parameter vector.[4]

4 Performance Evaluation

In this section, we investigate the performance of DCTA with extensive simulations over industrial operation (e.g., AIOps) scenarios using real-world data obtained from multiple data-driven building management systems.

4.1 Experiment Setup

For generating MTL tasks, we use a real-world *building operation* dataset released in [15], which contains four-year operation data for three high-rise commercial buildings in a metropolitan, collected by a major building service provider. The total data is more than 1 TB. Supported 50 MTL tasks include independent multi-task learning, self-adapted multi-task learning and clustered multi-task learning based on SVM, AdaBoost and Random Forest.

Our simulation consists of nine Raspberry Pi (version 3) and one laptop computer as shown in Fig. 8, which are all interconnected via WiFi under a star network topology in an office building. This represents an edge computing environment where the computational capabilities of edge nodes are heterogeneous. The simulation parameters, e.g., the transmission and receiving energy consumption of the Raspberry Pi are both 1.42×10^{-7} J/bit, the processing speed and energy consumption are 4.75×10^{-7} s/bit and 3.25×10^{-7} J/bit, which are based on the settings from [31].

4. For interested readers, the design of loss function and feature engineering can be found in our conference paper [14].

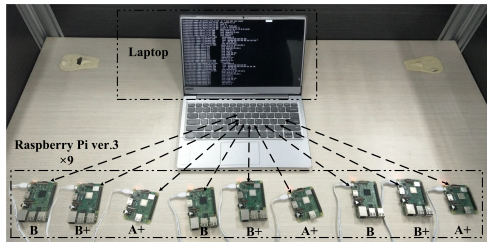


Fig. 8. The network topology and hardware choice in the experiments, where Raspberry Pi are with model types of A+, B, and B+.

4.2 Comparison Baselines and Metrics

Comparison Baselines. We employ the following state-of-the-art task allocation methods as baselines. It is worth noting that the first two are some of the non-data-driven methods (e.g., synthetic method) that have been widely suggested, and the last two are the data-driven methods we proposed.

- *Random Mapping (RM)* where each task is processed at different edge devices with equal probability [31]. In other words, tasks are randomly assigned.
- *Distributed Machine Learning (DML)* distributes tasks to multiple nodes, e.g., allocating the training iteration either to edge devices or to the cloud [32].
- *Clustered Reinforcement Learning (CRL)* conducts task allocation with our clustered reinforcement learning model.
- *Data-driven Cooperative Task Allocation (DCTA)* leverages an SVM model to adjust the decision of the CRL model.

Evaluation Metrics. From the perspective of the following metrics, we compare our proposed DCTA method with the others above state-of-the-art.

1) Overall Merit (OM). Given an allocation method, the ability to provide credible overall merit (e.g., energy saving) is crucial to all stakeholders. For interested readers, a more concrete definition of overall merit is available in previous Section 2.2.

2) Processing Time (PT). Our decision should be conducted before the deadline, the processing time we measure is the time the main device needs to partition the application and receive the output of the decision results. Formally

$$PT = t_s - t_c,$$

where t_s denotes the time instant when final decision is made; t_c denotes the time when each experiment start.

3) Energy Consumption (EC). Energy consumption is significantly critical for edge devices because most edge devices are energy-constrained. Formally, the energy consumption is defined as follows:

$$EC = \sum_{p \in \boldsymbol{P}} E_p + E_t,$$

where E_p and E_t denote the processing and transmission energy consumption of processor p, respectively.

4.3 Experiment Results

Result on Processing Time. Fig. 9 shows the processing time as a function of processors. Consistent with our intuition, as

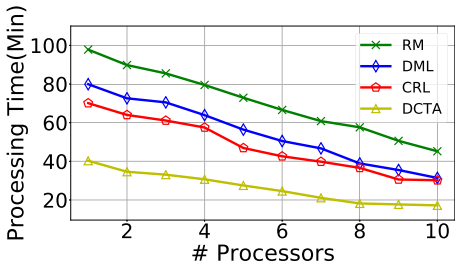


Fig. 9. The processing time of task allocation system with different number of processors.

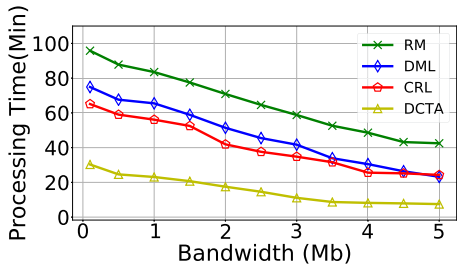


Fig. 11. The processing time of task allocation system with different bandwidth limits.

the number of processors increases, the processing time of the above methods gradually decreases. We see that DCTA can outperform RM, DML, and CRL by as much as 3.24, 2.32 and 2.01 times, respectively. On average, DCTA outperforms RM, DML, and CRL by 2.70, 2.05, and 1.80 times. That is because DCTA leverages data-driven techniques to capture the dynamic task importance and reduces the number of less important prediction tasks to perform.

Then, we compare the processing time of DCTA with that of RM, DML, and CRL for different average input data sizes. As we can see in Fig. 10, the processing time of our DCTA is always outperformed other state-of-the-art methods. For example, our DCTA has an improvement that is 2.71, 1.83, and 1.68 times to that of RM, DML, and CRL at the average input data size of 500 Mb. That is because our DCTA obtains the importance of each task which is time-dynamic changing, and then allocates to the most suitable edge devices to execute.

Finally, Fig. 11 shows the processing time as a function of network bandwidth. It is well known that network bandwidth affects the time of data transmission, and transmission time is also the main component of processing time. Thus, as the network bandwidth increases, the processing time also gradually decreases. But it is worth noting that our DCTA always outperforms RM, DML and CRL by 2.68, 1.94, and 1.71 times on average, respectively. That is mainly because our DCTA leverages data-driven techniques to capture the importance of each task and merely perform the most important tasks.

5 Case Study: Chiller AIOps on the Edge

In this section, we focus on applying our DCTA approach to the real-world edge-computing system. We first introduce the background of one core industry AIOps system, i.e, chiller AIOps system. We then present the overview of DCTA in chiller AIOps system and briefly introduce the system architecture and main components design within our chiller AIOps system. Finally, through extensive real-world experiments, we demonstrate the superiority of our chiller AIOps system integrating the DCTA mechanism.

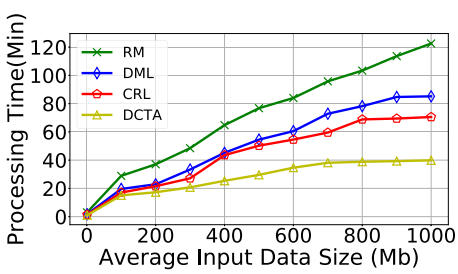


Fig. 10. The processing time of task allocation system with different data input sizes.

5.1 Background of Industry AIOps System

An important application of MTL on the edge is *AIOps*. The term *AIOps* [33] is coined as a system that utilizes big data, machine learning and other advanced analytics to enhance IT operations, such as monitoring, automation, and service desk, with proactive, customized and dynamic insight. Data-driven analytics have been widely suggested for IT Operations Management. According to Gartner Inc., by 2022, 40 percent of all large enterprises will adopt AIOps systems [34].

The industry AIOps system usually consists of two stages, i.e., Data-driven Multi-task Transfer Learning and Final Optimization, and they work as follows. First, when an industrial demand arrives, AIOps systems need to choose a series of data-driven prediction tasks to conduct, e.g., by using Data-driven Multi-task Transfer Learning. Second, it comes to Final Optimization. In this stage, the AIOps systems receive all the results of previous prediction tasks and conduct decisions until the decision performance, i.e., overall merit, is no longer improved.

Chiller AIOps System. As a case study, we focus on one of the core industry AIOps system, namely, *chiller AIOps system*, i.e., AIOps system conducting chiller sequencing, is deployed for one week on May, 2019, in a high-rise office building which serves more than three thousand people. A chiller is a machine that generates cooling power in commercial buildings and chiller sequencing is a significantly important operation, which aims to select run-time configurations of chillers at real-time so that the chiller AIOps system serves the time-varying cooling demand. For example, conducting chiller sequencing in a building with two chillers [0.5, 0.7] implies that chiller 1 and chiller 2 are operating at 50 and 70 percent of their maximum rated capacity, respectively. Thus, the chiller sequencing operation is to allocate the cooling load at any given time to the chillers in the most energy-efficient manner so that the overall cooling demand of the building is satisfied while at the same time the electricity consumed by the chillers is kept at a minimum [35]. Chiller AIOps system has been studied recently to significantly improve energy efficiency in commercial buildings and this case study is conducted based on a real-world chiller operation dataset [15].

5.2 Overview of DCTA in Chiller AIOps System

As mentioned before, the efficacy of chiller sequencing control in chiller AIOps system relies heavily on the *run-time*

performance profile of the chillers, namely the COP under different cooling load regimes. COP is a measure of the energy-efficiency of a chiller and captures the cooling power that it can output for a certain input power consumption [36]. Formally

$$COP_i = Q_i/E_i,$$

where E_i is the electrical power consumed by chiller i to deliver the required amount of cooling load Q_i.

The overall cooling load of the chiller AIOps system serves at a given time is the sum of the cooling load Q_i over all chillers i, i.e., $Q = \sum_i Q_i$, where $Q_i = c_i \times m_i \times \Delta_{Ti}$. Here, c_i is the thermal capacity of water (kJ/kg°C), m_i is the chilled water mass flow rate (kg/s) and Δ_{Ti} is the temperature difference between the returned and supplied chilled water (°C) [37]. All these quantities are logged by our chiller AIOps system.

Reliable chiller sequencing depends on the COP across all the loading conditions for chiller i. However, besides the well-known fact that COP degrades over time [38], [39], COP also fluctuates markedly over different cooling loads and environmental conditions [15], which makes it exceedingly difficult to capture within an analytical model. To this end, data-driven techniques can thus play a crucial role in accurate COP prediction for improved chiller sequencing in chiller AIOps system. Specifically, a *learning task* is defined as the coefficient of performance (COP) prediction of a chiller for one particular operation and works have been proposed for chiller AIOps [15], [40], [41], [42]. After COPs of operations is predicted, chiller sequencing conducted by selecting operation with the highest COP value to meet the cooling demand with the lowest electricity consumption.

Motivation of DCTA in Chiller AIOps. The chiller sequencing process requires performance predicted across all possible operations. There are too many controllable parameters in the industry and the number of parameter combinations is usually huge for all possible operations. However, the chiller sequencing process is typically accompanied by time limits, e.g., two hours for chiller sequencing [43]. A previous study indicates that blindly conducting all learning tasks leads to considerable time consumption which easily exceeds the time limits in chiller sequencing [15]. When merely partial operations are conducted in random order and these operations fail to meet the cooling demand, the backup chiller plant would be launched and additionally consumes a large amount of electricity [44]. Therefore, we can conduct the proposed task allocation which assigns more important tasks to more powerful edge devices for priority execution under time limits.

Based on the above real-world chiller operation dataset, while in principle all COP operations (i.e., learning tasks) may be selected to conduct the chiller sequencing, in practice only a small subset of them are frequently selected in the optimal sequencing operation. The historical best operations can be computed with the sequencing optimization based on the ground truth of COP of 1,460 days from 2012 to 2015. Then we can count the number of cases for each operation to be selected as the best operation and thus obtain the probability to become optimal. For example, if an operation is selected in 120 days as the best operation over the total 1,460 days, its probability to become optimal is computed as 120 / 1460 = 8.22%. Fig. 12 shows that the probability of becoming the best operation for different machines vary greatly among the exponential optional operation space. It can be seen that there merely a small portion of operations are frequently selected. Results also confirm our previous Observation 1 in Section 2.3.

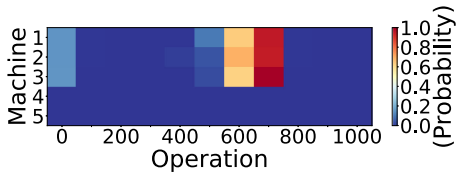


Fig. 12. Probability of becoming best operation for different machines.

Task Importance in Chiller AIOps. The key of the DCTA lies in the computation of task importance. Next, in the context of this AIOps case study, we formally present a specified formulation of the *task importance* computation.

As discussed in previous Definition 1 in Section 2.2, given model parameters $\boldsymbol{\theta}$, the task importance $\mathcal{I}_j$ can be updated using the merit function $\mathcal{H}(\cdot)$. For interested readers, a more concrete definition of $\mathcal{H}(\cdot)$ is also available in Section 2.2, where involved two following important concepts, i.e., the ideal electricity consumption D and decision-making function $\mathcal{D}(\cdot)$. Specifically, as for D, we first find the best operation of each chiller (i.e., with the highest COP value) in each day through the historical ground truth of COP data, and then compute the electricity consumption of conducting these operations as the ideal performance.

Next, the decision-making function $\mathcal{D}(\cdot)$ is intrinsically solving the chiller sequencing optimization problem finding the best chiller operations combination which minimize the total electricity consumption on one day, where all time instances in one day are denoted by T and each operation is conducted at time $t \in T$. Let L_i denote the maximum cooling capacity of chiller $i < n$ and $S_{i,t}$ denote the partial load ratio of chiller i at time t. Formally

$$\mathcal{D}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \sum_{t=1}^{T} \sum_{i=1}^{n} L_i \cdot S_{i,t}/COP_{i,t}$$

$$\text{s.t.} \sum_{i=1}^{n} Q_i > Q_D \quad and \quad T_N \leq T_D,$$

where $COP_{i,t}$ denotes the data-driven prediction performance of chiller i at time t; Q_i and Q_D respectively denote the cooling load produced by chiller i and the total cooling demand; T_N and T_D denote the total processing time and the deadline, respectively. More specifically, the deadline T_D here means the total time length of one chiller sequencing operation, including the computation time and the mechanical switching time, computed considering both the periodic interval t_P and mechanical switching time t_M, e.g., $T_D = \min(t_P, t_M)$ [15].

5.3 Device Overview of Chiller AIOps System

According to above, we conduct the data-driven task allocation based on the chiller AIOps system in the Pacific Place, Hong Kong, where the network topology is shown in Fig. 14. The equipment of chillers, pumps, air-handling unit, and cooling tower differ greatly in operation, maintenance, and

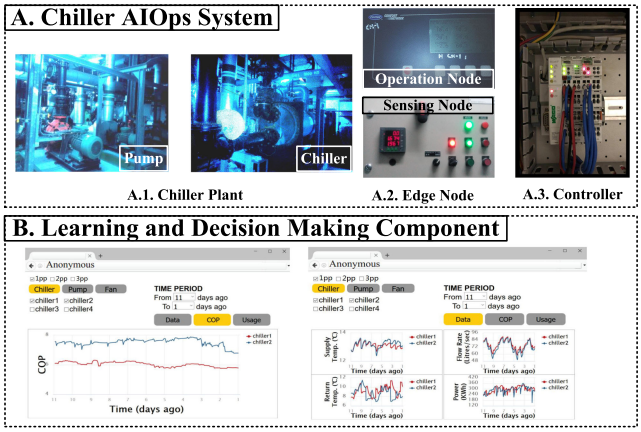


Fig. 13. Chiller AIOps system on the edge.

services. The data of each equipment in the chiller plant (Fig. 13 A.1) are captured and transmitted by 13 edge nodes, including 3 operation nodes (from the vendor of Trane, Fig. 13 A.2) conducting and recording operations, and 10 sensing nodes (from the vendor of Schneider Electric, Fig. 13 A.2) collecting sensing data. To process data from different types of equipment, we choose a centralized approach, where edge node transmits data to the controller (from the vendor of Wago, Fig. 13 A.3), and controllers are responsible for task allocation and decision making for the edge nodes. Finally, 3 operation nodes conduct data-driven COP prediction and send control sequences to devices (Fig. 13 B). Other sensing nodes without computation power are merely used to collect data.

Though hardware can be fully redeployed after introducing data-driven techniques [45], for the scalability purpose, we choose an incremental deployment for the chiller AIOps system, with minimal revision for the current HVAC system. That is to say, we leverage only the current commercial off-the-shelf components and avoid deploying any additional equipment within the HVAC system. However, we may sacrifice the probability to obtain more sensing data and have even better prediction performance, if we avoid deploying additional equipment inside the local system in each building for the scalability purpose.

5.4 Components Design Within Chiller AIOps System

To apply our DCTA approach to the chiller AIOps system, we also introduce the architecture overview of our chiller AIOps system, as shown in Fig. 15. The architecture contains four main modules: (1) *Data Collecting Module* collects the data from the surroundings for analysis. Not only the current data but also the historical data are needed to be

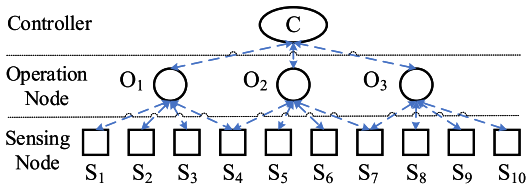


Fig. 14. The network topology of chiller AIOps system on the edge.

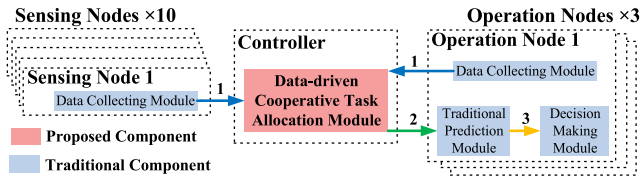


Fig. 15. Architecture overview of chiller AIOps system on the edge.

collected. (2) *Data-driven Cooperative Task Allocation (DCTA) Module* captures the time-dynamic task importance and allocates tasks with data-driven techniques, which has been introduced in detail in Section 3. (3) *Traditional Prediction Module* executes the data-driven prediction tasks at the edge nodes and outputs the prediction results. (4) *Decision Making Module* receives the prediction results from the multiple edge nodes and conducts the optimal decision which is to maximize the overall system merit.

The DCTA module for task allocation lies in the Controller and the design is elaborately introduced in previous Section 3. In the following, we are to briefly introduce the design of other components, i.e., *data collecting module*, *traditional prediction module*, and *decision making module*, within our chiller AIOps system architecture.

Data Collecting Module. The module lies in the Sensing Nodes, e.g., the temperature sensor or humidity sensor, which collects the data from the surroundings for analysis. There exist a common data storage problem due to the storage limitations on these edge nodes. To tackle this problem, we keep uploading data to a more powerful edge node, e.g., gateway or server, and overwrite historical data on these edge nodes when the storage is insufficient.

Traditional Prediction Module. The module lies in the Operation Nodes, e.g., gateway or router, which executes data-driven COP prediction tasks and outputs the prediction results. To ensure the accuracy of each data-driven task in the case of data scarcity on these edge nodes, we apply clustered multi-task learning approach [46]. It learns with training data not only from the target task, but also from other tasks, e.g., cases with similar temporal, meteorological and mechanical conditions.

Decision Making Module. The module lies in the Controller or Operation Nodes, e.g., server or gateway, which should work in an iterative optimization way. Under the circumstance, the frequency of the decision update is then critical to edge nodes network resource utilization and energy consumption. To tackle this problem, we propose an efficient algorithm to determine the frequency of decision update by analyzing the historical decision data. Specifically, we update the decision each time when an industrial demand coming. In order to reduce damage to the system, we ensure that the time interval between the two decisions can meet the needs of the system to transition from one steady state to another. As for the effects of varying this frequency, it would be an interesting future work for us to investigate the optimal frequency of decision update in industrial scenarios.

5.5 Experiment Results

Result on Overall Merit. With the chiller AIOps system, we first compare the overall merit of our DCTA with that of the

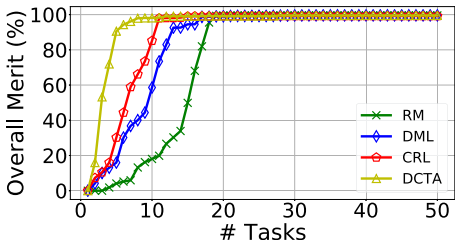


Fig. 16. The overall merit as a function of the number of performed tasks.

other state-of-the-art task allocation methods. Fig. 16 shows that, on one hand, our DCTA approach and other state-of-the-art methods can eventually achieve the same performance; on the other hand, with the same performance, our DCTA approach can greatly reduce the number of tasks performed which means significant savings in time and resources. That is because our DCTA approach is developed combined with the runtime data in the real environment and a huge amount of simulation data. In addition, it leverages the ensemble technique to avoid overfitting in non-linear modeling, which can successfully capture the system local and dynamic performance.

Result on Processing Time. To show the potential of saving time, We compare the processing time of the state-of-the-art task allocation methods. In Fig. 17, we can see that our DCTA outperforms RM, DML, and CRL by 50.2, 38.6 and 30.2 percent, respectively. That is because DCTA uses data-driven allocation to select the most important tasks for prediction, unlike other non-data-driven methods.

Result on Energy Consumption. Fig. 18 compares our DCTA approach with RM, DML and CRL method over a different number of tasks, in terms of Average Energy Consumption on edge devices. On average, our DCTA outperforms RM, DML and CRL by 48.4, 39.6 and 31.3 percent, respectively. That is because not all predictions on all operations are necessary. Our DCTA captures the top important operations and still maintains the superiority of data-driven techniques.

6 Related Work

Task Allocation has been intensively researched in cloud computing systems [7], [8], [9]. Recent years have witnessed great prospects exhibited down to the edge, e.g., from OpenCL (2008) [47] to AWS IoT Greengrass (2017) [48] and Microsoft Azure IoT Edge (2018) [49]. Under edge computing, existing works on task allocation either 1) partition the machine-learning model and its input, or 2) are conducted according to different objectives.

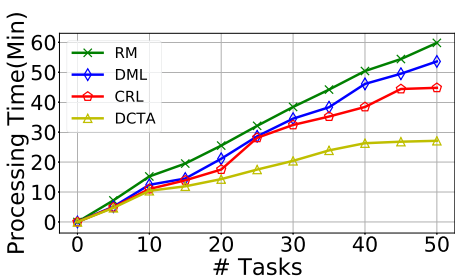


Fig. 17. The processing time of task allocation system with different number of tasks.

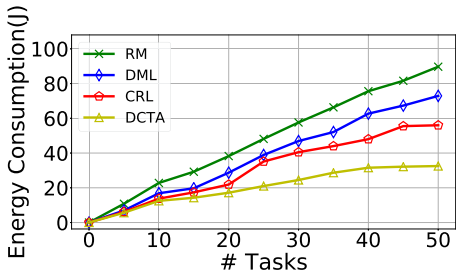


Fig. 18. The energy consumption of task allocation system with different number of tasks.

First, task allocation in many distributed machine learning systems [32], [50], [51], [52] have successfully demonstrated their effectiveness to enable big-data applications deployed on a large number of machines. For example, when allocating task for deep neural network (DNN), Neurosurgeon [53] identifies a strategy in a fine-grained layer level between edge and cloud. A similar approach presented in [54] proposes a design guideline for DNN partitioning based on the layer-wise trade-off study. These methods provide the capability to accelerate the execution of a single data-driven task on the edge.

Second, existing works also consider different objectives for task allocation [55], [56]. Examples include reducing the energy consumption of edge device while predefined delay constraint is satisfied [11], finding a proper trade-off between the energy consumption and the execution delay [12], and minimizing the overall application execution cost [10]. A majority of these works are not designed for machine learning tasks. Nevertheless, though these techniques may consider a multi-task setting, they regard all submitted tasks as equally important, which leads to inefficient resource allocation at a task level when directly applied for MTL.

Different from these works, our study investigates task allocation for multiple machine-learning tasks without knowing task priority. We capture and leverage task importance to accelerate the overall learning process, which sheds some new light on task allocation for MTL on the edge.

Machine learning for Complicated Optimization Problems has been successfully employed especially with time-varying parameters and complicated solutions which are repeatedly conducted [57], [58]. Examples include intelligent logistics [26], code optimization [59], [60], [61], task scheduling [62], [63], [64]. Our cooperative approach is closely related to ensemble learning where multiple models are used to solve an optimization problem. Ensemble learning is shown to be useful when scheduling parallel tasks [65] and optimizing application memory usage [66]. This work is the first attempt in applying ensemble techniques to optimize task allocation of MTL with task importance on the edge.

Industry AIOps. Recent advances in machine learning have been adopted in various business applications for both individuals and enterprises, whereas the industry sector receives relatively less attention mainly due to the common issue of data scarcity, especially in the past. However, nowadays in the industry sector, the lowered cost of sensing, computing, and communications has made the impractical data-driven techniques in the late 1980s eminently practical, e.g., industrial robots, driver-less cars, and recently, energy-efficient buildings [67]. It is time to deliver a punch and reduce the cost using

data-driven techniques on each of the industry sector. E.g., in building management systems, since the release of BLUED [68] on 2012, a dataset of electricity consumption of buildings from the data analytics community of SIGKDD, various works demonstrated the need for using data analytics in building management systems. Then, in SIGKDD 2016, a data-driven study on energy breakdown in buildings reveals the huge electricity demand [69]. Nevertheless, how machine learning can be deployed is still vague in each of the industry sectors to guide mechanical operations, especially on the edge.

7 Discussion

Naturally, there is room for further work and possible improvements. We discuss a few points here.

Data Scarcity on the Edge. For industrial edge-computing applications, data scarcity often exists even though cloud storage can still cooperate for big data. The data scarcity is the result from 1) prohibitive cost or inherent difficulty in obtaining required proper training samples, 2) with respect to the application complexity and uncertainty. First, when considering the privacy concern, storage limitations, budget, and real-time requirements, partial or even the whole data set is not possible to be stored, transmitted and processed for the edge-computing applications, compared with that of cloud-computing applications. Meantime, due to the instability of the sensing devices, data loss also occurs frequently in some environments. Worse still, an industrial application can be complex or highly uncertain which requires a larger amount of data. For example, many robots for text production, such as search engines or translation programs, have difficulties in finding sufficient samples for each context. The reason lies in the context of words which can result in ambiguities and there exists a huge amount of possible contexts. Thus, we believe moves should be conducted for the data scarcity issue on the edge and we provide an edge-based MTL.

Real-Time Sensing Data. Real-time sensing data facilitate the learning process by incorporating the run-time observations on environmental dynamics. In order to capture the run-time effect from real-time sensing data, we discuss two learning modes, i.e., the offline and online modes. First, the offline mode divides historical samples into multiple clusters in advance, e.g., using K-means. When the real-sensing data is coming, the system selects the most similar clustered samples to train and predict. Its drawback lies in the possibly low prediction accuracy due to the offline clustering. Second, the online mode prepares the training samples in a run-time manner by finding those which are the most similar with the real-time data, e.g., using KNN. This mode guarantees a high prediction accuracy but could lead to extra time to choose the proper training data. In this paper, we adopt the online mode to guarantee that our final decision making can be more reliable. The additional time overhead can be significantly reduced through our proposed data-driven task allocation mechanism.

Multi-Task Assumption. In this study, our approach is designed to tackle time-varying environments. We assume that 1) there are multiple related and indivisible machine-learning tasks, and 2) there is no strong pre- and post-dependency, which is also a prerequisite for performing multi-task transfer learning, and 3) there is not all tasks need to be learned individually from scratch to make the final decision. Thus, those cases 1) under single-task settings, or 2) under multi-task settings but with the sequential dependency between tasks, or 3) under multi-task settings but all tasks must be finished to produce the final result, are beyond the scope of this paper. It would be an interesting future work to extend our approach to those scenarios.

8 Conclusion

In this paper, we study task allocation for MTL scenarios on the edge, by introducing task importance and making the following contributions. First, we reveal that it is important to measure the impact of tasks on decision performance improvement and quantify task importance. We also observe the long-tail property of task importance, which serves as a key metric to guide task allocation, and facilitates resource saving from less important tasks. Second, we show that task allocation with task importance for MTL (TATIM) is a variant of NP-complete Knapsack problem, where the complicated computation to solve this problem needs to be conducted repeatedly under varying contexts. To solve TATIM with high computational efficiency, we propose a Data-driven Cooperative Task Allocation (DCTA) approach. Third, we conduct trace-driven simulations to evaluate the performance of the proposed DCTA approach. Extensive simulations show that our DCTA approach saves 3.24 times of processing time compared to the state-of-the-art. Finally, we add a new comprehensive real-world case study on AIOps for our DCTA approach to bridge model and practice, by proposing a new architecture and main components design within AIOps system. Extensive experiments are complemented to demonstrate the superiority, i.e., 48.4 percent energy saving, of AIOps system integrating our DCTA approach. We believe that our DCTA approach offers an effective and practical mechanism for reducing the required resource associated with performing MTL on the edge.

Acknowledgments

The authors would like to thank Zihan Lin for his valuable discussion and feedback. This work was supported in part by the NSFC under Grant 61722206 and 61761136014 (and 392046569 of NSFC-DFG) and 61520106005, in part by National Key Research and Development (R&D) Plan under Grant 2017YFB1001703, in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFKJXX009 and 3004210116, in part by the National Program for Support of Top-notch Young Professionals in National Program for Special Support of Eminent Professionals. Dan Wang's work was supported in part by RGC GRF PolyU 15210119, CRF C5026-18G, ITF UIM/363, ITF ITS/070/19FP, PolyU 1-ZVPZ, and a Huawei Collaborative Grant. All opinions, findings, conclusions and recommendations in this article are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] M. L. Hutchinson, E. Antono, B. M. Gibbons, S. Paradiso, J. Ling, and B. Meredig, "Overcoming data scarcity with transfer learning," *CoRR*, vol. abs/1711.05099, 2017. [Online]. Available: http://arxiv.org/abs/1711.05099

[2] C. Yuan, W. Hu, G. Tian, S. Yang, and H. Wang, "Multi-task sparse learning with beta process prior for action recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 423–429.

[3] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 4460–4464.

[4] S. Emrani *et al.*, "Prognosis and diagnosis of Parkinson's disease using multi-task learning," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2017, pp. 1457–1466.

[5] J. Zhou, L. Yuan, J. Liu, and J. Ye, "A multi-task learning formulation for predicting disease progression," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2011, pp. 814–822.

[6] T. Idé, D. T. Phan, and J. Kalagnanam, "Multi-task multi-modal models for collective anomaly detection," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 177–186.

[7] T. Biswas, P. Kuila, and A. K. Ray, "Multi-level queue for task scheduling in heterogeneous distributed computing system," in *Proc. IEEE 4th Int. Conf. Adv. Comput. Commun. Syst.*, 2017, pp. 1–6.

[8] B. Hong and V. Prasanna, "Adaptive allocation of independent tasks to maximize throughput," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 10, pp. 1420–1435, Oct. 2007.

[9] Y. Jiang, Y. Zhou, and Y. Li, "Reliable task allocation with load balancing in multiplex networks," *ACM Trans. Auton. Adaptive Syst.*, vol. 10, no. 1, 2015, Art. no. 3.

[10] S. Sundar and B. Liang, "Offloading dependent tasks with communication delay and deadline constraint," in *Proc. IEEE INFOCOM*, 2018, pp. 37–45.

[11] S. Cao, X. Tao, Y. Hou, and Q. Cui, "An energy-optimal offloading algorithm of mobile computing based on HetNets," in *Proc. IEEE Int. Conf. Connected Vehicles Expo*, 2015, pp. 254–258.

[12] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf.*, 2016, pp. 1–6.

[13] Y. Geng, Y. Yang, and G. Cao, "Energy-efficient computation offloading for multicore-based mobile devices," in *Proc. IEEE INFOCOM*, 2018, pp. 46–54.

[14] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst.*, 2019, pp. 1040–1050.

[15] Z. Zheng *et al.*, "Data driven chiller sequencing for reducing HVAC electricity consumption in commercial buildings," in *Proc. 9th Int. Conf. Future Energy Syst.*, 2018, pp. 236–248.

[16] P. Yang and J. He, "Heterogeneous representation learning with structured sparsity regularization," in *Proc. IEEE 16th Int. Conf. Data Mining*, 2016, pp. 539–548.

[17] K. Lin, J. Xu, I. M. Baytas, S. Ji, and J. Zhou, "Multi-task feature interaction learning," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1735–1744.

[18] P. Gong, J. Ye, and C. Zhang, "Robust multi-task feature learning," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 895–903.

[19] Y. Zhang and Q. Yang, "Learning sparse task relations in multi-task learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 2914–2920.

[20] K. Lin and J. Zhou, "Interactive multi-task relationship learning," in *Proc. IEEE Int. Conf. Data Mining*, 2016, pp. 241–250.

[21] D. Oyen and T. Lane, "Leveraging domain knowledge in multitask Bayesian network structure learning," in *Proc. AAAI Conf. Artif. Intell.*, 2012, pp. 1091–1097.

[22] D. Isele, M. Rostami, and E. Eaton, "Using task features for zero-shot knowledge transfer in lifelong learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1620–1626.

[23] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3712–3722.

[24] H. Hu, L. Chen, B. Gong, and F. Sha, "Synthesize policies for transfer and adaptation across tasks and environments," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 1176–1185.

[25] A. Sax, B. Emi, A. R. Zamir, L. J. Guibas, S. Savarese, and J. Malik, "Mid-level visual representations improve generalization and sample efficiency for learning active tasks," *CoRR*, vol. abs/1812.11971, 2018. [Online]. Available: http://arxiv.org/abs/1812.11971

[26] X. Li, "Development of intelligent logistics in china," in *Contemporary Logistics in China*. Berlin, Germany: Springer, 2018, pp. 181–204.

[27] R. Iglesias, F. Rossi, K. Wang, D. Hallac, J. Leskovec, and M. Pavone, "Data-driven model predictive control of autonomous mobility-on-demand systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–7.

[28] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015, Art. no. 529.

[29] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *Proc. 12th USENIX Symp. Netw. Syst. Des. Implementation*, 2015, pp. 455–468.

[30] N. Liu *et al.*, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 372–382.

[31] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Proc. IEEE Int. Conf. Commun.*, 2016, pp. 1–6.

[32] S. Teerapittayanon, B. McDanel, and H. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 328–339.

[33] A. Lerner, AIOps Platforms. 2018. [Online]. Available: https://blogs.gartner.com/andrew-lerner/2017/08/09/aiops-platforms/

[34] W. Cappelli and others. Market Guide for AIOps Platforms. 2018. [Online]. Available: https://goo.gl/CHkqyN

[35] Z. Liu, H. Tan, D. Luo, G. Yu, J. Li, and Z. Li, "Optimal chiller sequencing control in an office building considering the variation of chiller maximum cooling capacity," *Energy Buildings*, vol. 140, pp. 430–442, 2017.

[36] Wikipedia. Coefficient of performance. 2019. [Online]. Available: https://en.wikipedia.org/wiki/Coefficient_of_performance

[37] Y. Liao, Y. Sun, and G. Huang, "Robustness analysis of chiller sequencing control," *Energy Convers. Manage.*, vol. 103, pp. 180–190, 2015.

[38] F. Yu and K. Chan, "Optimization of water-cooled chiller system with load-based speed control," *Appl. Energy*, vol. 85, no. 10, pp. 931–950, 2008.

[39] N. Firdaus *et al.*, "Chiller: Performance deterioration and maintenance," *Energy Eng.*, vol. 113, no. 4, pp. 55–80, 2016.

[40] A. Michopoulos *et al.*, "Three-years operation experience of a ground source heat pump system in northern greece," *Energy Buildings*, vol. 39, no. 3, pp. 328–334, 2007.

[41] K. M. Powell *et al.*, "Optimal chiller loading in a district cooling system with thermal energy storage," *Energy*, vol. 50, pp. 445–453, 2013.

[42] T. Hartman, "All-variable speed centrifugal chiller plants," *ASHRAE J.*, vol. 56, no. 6, pp. 68–79, 2014.

[43] Y. Sun, S. Wang, and F. Xiao, "In situ performance comparison and evaluation of three chiller sequencing control strategies in a super high-rise building," *Energy Buildings*, vol. 61, pp. 333–343, 2013.

[44] F. Yu and K. Chan, "Economic benefits of optimal control for water-cooled chiller systems serving hotels in a subtropical climate," *Energy Buildings*, vol. 42, no. 2, pp. 203–209, 2010.

[45] P. K. Agyapong, M. Iwamura, D. Staehle, W. Kiess, and A. Benjebbour, "Design considerations for a 5G network architecture," *IEEE Commun. Mag.*, vol. 52, no. 11, pp. 65–75, Nov. 2014.

[46] L. Jacob, J.-P. Vert, and F. R. Bach, "Clustered multi-task learning: A convex formulation," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2009, pp. 745–752.

[47] The Khronos OpenCL Working Group, "OpenCL-The open standard for parallel programming of heterogeneous systems," 2019. [Online]. Available: https://www.khronos.org/opencl/

[48] AWS, "IoT Greengrass," 2019. [Online]. Available: https://aws.amazon.com/cn/greengrass/

[49] Microsoft, "Azure IoT Edge," 2019. [Online]. Available: https://azure.microsoft.com/zh-cn/services/iot-edge/

[50] K. Hsieh *et al.*, "Gaia: Geo-distributed machine learning approaching LAN speeds," in *Proc. 14th USENIX Conf. Netw. Syst. Des. Implementation*, 2017, pp. 629–647.

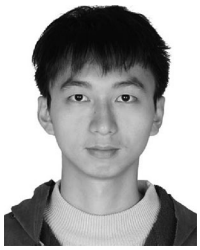
[51] E. P. Xing *et al.*, "Petuum: A new platform for distributed machine learning on big data," in *Proc. 21st ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1335–1344.

[52] M. Li *et al.*, "Communication efficient distributed machine learning with the parameter server," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 19–27.

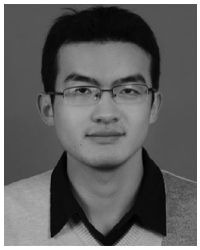
[53] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, 2017.

[54] A. Sax, B. Emi, A. R. Zamir, L. J. Guibas, S. Savarese, and J. Malik, "Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies," *Comput. Vis. Pattern Recognit.*, 2018, *arXiv:1812.11971*.
[55] B. Gao, Z. Zhou, F. Liu, and F. Xu, "Winning at the starting line: Joint network selection and service placement for mobile edge computing," in *Proc. IEEE INFOCOM*, 2019, pp. 1459–1467.
[56] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *Proc. IEEE INFOCOM*, 2019, pp. 2566–2574.
[57] F. Samreen, Y. Elkhatib, M. Rowe, and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," *IEEE/IFIP Netw. Operations Manage. Symp.*, pp. 557–563, 2016.
[58] Z. Wang and M. O'Boyle, "Machine learning in compiler optimization," *Proc. IEEE*, vol. 106, no. 11, pp. 1879–1901, Nov. 2018.
[59] C. Cummins, P. Petoumenos, Z. Wang, and H. Leather, "End-to-end deep learning of optimization heuristics," in *Proc. 26th Int. Conf. Parallel Archit. Compilation Techn.*, 2017, pp. 219–232.
[60] W. F. Ogilvie, P. Petoumenos, Z. Wang, and H. Leather, "Minimizing the cost of iterative compilation with active learning," in *Proc. Int. Symp. Code Gener. Optim.*, 2017, pp. 245–256.
[61] B. Taylor, V. S. Marco, and Z. Wang, "Adaptive optimization for OpenCL programs on embedded heterogeneous systems," *ACM SIGPLAN Notices*, vol. 52, no. 5, pp. 11–20, 2017.
[62] Y. Wen, Z. Wang, and M. F. O'boyle, "Smart multi-task scheduling for OpenCL programs on CPU/GPU heterogeneous platforms," in *Proc. 21st Int. Conf. High Perform. Comput.*, 2014, pp. 1–10.
[63] J. Ren, L. Gao, H. Wang, and Z. Wang, "Optimise web browsing on heterogeneous mobile platforms: A machine learning based approach," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.
[64] S. Chen, J. Fang, D. Chen, C. Xu, and Z. Wang, "Optimizing sparse matrix-vector multiplication on emerging many-core architectures," *CoRR*, vol. abs/1805.11938, 2018. [Online]. Available: http://arxiv.org/abs/1805.11938
[65] M. K. Emani and M. O'Boyle, "Celebrating diversity: A mixture of experts approach for runtime mapping in dynamic environments," *ACM SIGPLAN Notices*, vol. 50, no. 6, pp. 499–508, 2015.
[66] V. S. Marco, B. Taylor, B. Porter, and Z. Wang, "Improving spark application throughput via memory aware task co-location: A mixture of experts approach," in *Proc. 18th ACM/IFIP/USENIX Middleware Conf.*, 2017, pp. 95–108.
[67] Z. Zheng *et al.*, "An edge based data-driven chiller sequencing framework for HVAC electricity consumption reduction in commercial buildings," *IEEE Trans. Sustain. Comput.*, to be published, doi: 10.1109/TSUSC.2019.2932045.
[68] A. Filip *et al.*,, "BLUED: A fully labeled public dataset for event-based nonintrusive load monitoring research," in *Proc. 2nd ACM SIGKDD Workshop Data Mining Appl. Sustainability*, 2012.
[69] N. Batra, A. Singh, and K. Whitehouse, "Gemello: Creating a detailed energy breakdown from just the monthly electricity bill," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 431–440.

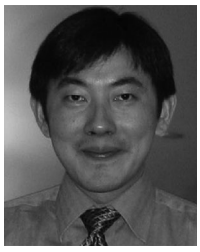
Qiong Chen received the BEng degree from the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China where he is currently working toward the MEng degree. His research interests include applied machine learning and edge computing. He received the Best Paper Award of ACM International Conference on Future Energy Systems (ACM e-Energy), in 2018.



Zimu Zheng received the BEng degree in software engineering from the South China University of Technology, Guangzhou, China, in 2014, and the PhD degree in computer science from the Hong Kong Polytechnic University, Hong Kong, in 2019. He is currently a research staff with Cloud BU of Huawei. He has received several awards for outstanding technical contributions in Huawei. He also received the Best Paper Award of ACM e-Energy and the Best Paper Award of ACM BuildSys in 2018. His research interest includes applied machine learning with IoT Data.



Chuang Hu received the BSc and MSc degrees from the Department of Computing Science, Wuhan University, Wuhan, China, in 2013 and 2016, respectively. He is currently working toward the PhD degree in the Department of Computing, Hong Kong Polytechnic University. His research interests include networking, wireless technologies, IoT, cloud computing, and big data.



Dan Wang (S'05–M'07–SM'13) received the BSc degree from Peking University, Beijing, China, in 2000, MSc degree from Case Western Reserve University, Cleveland, Ohio, in 2004, and PhD degree from Simon Fraser University, Burnaby, British Columbia, Canada, in 2007, all in computer science. He is currently an associate professor with the Department of Computing, Hong Kong Polytechnic University. His research interests include network architecture and QoS, smart buildings, and Industry 4.0. He is a senior member of the IEEE.

Fangming Liu (S'08–M'11–SM'16) received the BEng degree from the Tsinghua University, Beijing, China, and the PhD degree from the Hong Kong University of Science and Technology, Hong Kong. He is currently a full professor with the Huazhong University of Science and Technology, Wuhan, China. His research interests include cloud computing and edge computing, datacenter and green computing, SDN/NFV/5G and applied ML/AI. He received the National Natural Science Fund (NSFC) for Excellent Young Scholars, and the National Program Special Support for Top-Notch Young Professionals. He is a recipient of the Best Paper Award of IEEE/ACM IWQoS 2019, ACM e-Energy 2018 and IEEE GLOBECOM 2011, as well as the First Class Prize of Natural Science of Ministry of Education in China. He is a senior member of the IEEE.