2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

Data-driven Task Allocation for Multi-task Transfer Learning on the Edge

Qiong Chen[1] Zimu Zheng[2,3] Chuang Hu[2] Dan Wang[2] Fangming Liu*[1]
[1]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, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[3]Technical Innovation Department, Cloud BU, Huawei Technologies Co.Ltd

***Abstract*—Edge computing for machine learning has become a heated research topic. On edge devices, data scarcity occurs as a common problem where transfer learning serves as a widely-suggested remedy. Nevertheless, one obstacle is that transfer learning imposes heavy computation burden to the resource-constrained edge devices. Motivated by the fact that only a few tasks of Multi-task Transfer Learning (MTL) have a higher potential for overall decision performance improvement, we design a novel task allocation scheme, which assigns more important tasks to more powerful edge devices to maximize the overall decision performance. In this paper, we focus on task allocation under multi-task scenarios by introducing *task importance* and make the following contributions. First, we reveal that it is important to measure the impact of tasks on overall decision performance improvement and quantify task importance. We also observe the long-tail property of task importance, i.e., only a few tasks are important, which facilitates more efficient task allocation. Second, we show that task allocation with task importance for MTL (TATIM) is in fact a variant of the 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 innovatively propose a Data-driven Cooperative Task Allocation (DCTA) approach. Third, we evaluate the performance of our DCTA approach by applying it to a real-world industrial operation (e.g., AIOps) scenario. Experiments show that our DCTA approach can reduce 3.24 times of processing time compared with the state-of-the-art when solving TATIM. We offer our DCTA approach as an effective and practical mechanism for reducing the required resource associated with performing MTL on edge devices.**

I. 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 requirement 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 separation 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, but the computing resource of edge device is limited. Many efforts have been devoted to designing task allocation mechanisms to achieve varying objectives, e.g., optimize 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]–[13]. They are usually designed for general machine learning and typically assume that all tasks contribute

*The corresponding author is Fangming Liu (fmliu@hust.edu.cn). This work was supported in part by the NSFC under Grant 61761136014 (and 392046569 of NSFC-DFG) and 61722206 and 61520106005, in part by National Key Research & 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, in part by Hong Kong ITF UIM/363, and in part by Technical Innovation Department, Cloud BU, Huawei Technologies Co.Ltd.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00107

identically to overall performance improvement of the application. However, in MTL, tasks belonging to the same machine-learning-based application usually 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 inefficient 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 [14]. 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.

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 novel Data-driven Cooperative Task Allocation (DCTA) mechanism to maximize the application's overall decision performance among multiple tasks on the edge.

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 not only due to 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 novel 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 novel task allocation approach within a data-driven building management system. We also evaluate various distinct task allocation approaches on the real-world industrial operation (e.g., AIOps) scenario. Experiments show that DCTA saves 3.24 times of processing time when solving TATIM compared to the state-of-the-art.

II. 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.

A. 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 performance (COP) prediction of a chiller for one particular load as a task. 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

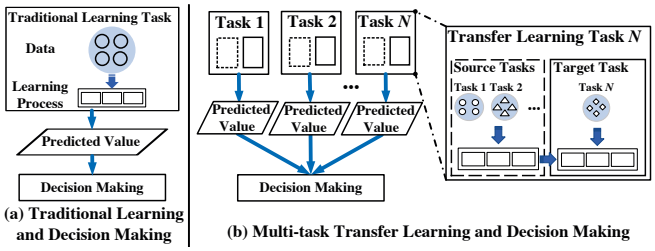

Fig. 1: Decision making with (a) traditional learning and (b) transfer learning.

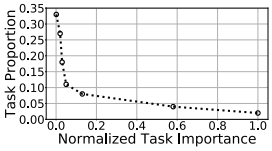

Fig. 2: The long-tail distribution of normalized task importance.

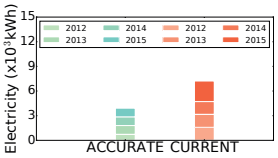

Fig. 3: The potential with ACCURATE and CURRENT schemes.

possible, MTL practitioners retrain their models repeatedly to get the final model with the best quality, including to explore feature representation [15]–[17], adjust structures of task relationship [18]–[20] and tune hyper-parameters [21]. For better understanding, a formal formulation of transfer-learning tasks on the edge is available in the following Section II-C.

B. Introduction 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.

For further study of the importance of tasks, we plot the distribution of task importance in Fig. 2, based on a real-world transfer learning dataset released in [22]. The importance of a task is defined as the overall performance degradation of the final decision making when this task is not conducted. 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% of tasks have a high contribution of over 80% to the final operation decision performance. We therefore have such an observation. Results in a recent CVPR paper also confirm such an observation [23].

Observation 1. *In MTL, redundant or noisy tasks exist; The importance of tasks has a long-tail distribution.*

The 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. We then formally define *task importance*.

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; decision function $\mathcal{H}(\cdot)$ outputs the final decision performance; $\boldsymbol{J}$ denotes the entire task set.

Thus, given model parameters $\boldsymbol{\theta}$, the task importance $\mathcal{I}_j$ can be updated using the decision function $\mathcal{H}(\cdot)$. Given that the ideal performance of final decision can usually be collected after task evaluation, an example of implementation of $\mathcal{H}(\cdot)$ might be $\mathcal{H}(\mathbf{J}; \boldsymbol{\theta}) = 1 - \frac{|D - \mathcal{D}(\boldsymbol{\theta})|}{D}$, where D denotes the ideal performance and $\mathcal{D}(\cdot)$ is the decision-making function given model parameters. The 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 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.

More important tasks should be allocated to more powerful edge devices under MTL scenarios with execution time limits. To demonstrate the benefit, we show the performance of decision making by conducting related tasks, compared with the performance of conducting random tasks. Figure 3 shows the result based on the transfer-learning dataset mentioned above [22]. Accurate task allocation is conducted with high task importance. Stacked bars on the left indicate the final performance of decision making with accurate task allocation, i.e., energy saving for cooling. Bars on the right show the performance of the current scheme using random task allocation. We see that the accurate task allocation considering task importance could have resulted in an average of over 45.68% potential improvement in terms of the final decision making performance. These results demonstrate that there is a significant room to improve the final decision making performance when using a more accurate and robust scheme of task allocation.

Observation 2. *Final decision making with MTL can be improved by task allocation according to task importance.*

However, the task importance is not directly available. Based on the above dataset, we also conduct two experiments as a more detailed distribution study 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 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 machines,

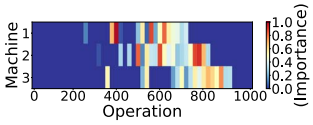

Fig. 4: Average task importance for different machines and operations.

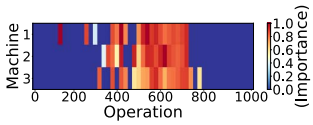

Fig. 5: Task importance variation for different machines and operations.

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.

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

The time-dynamic task importance changes in varying contexts, e.g., with different external factors (like environmental conditions and dynamic industrial demands) and internal factors (like machine configurations and response). 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.

C. Problem of Task Allocation with Task Importance for MTL

Based on the above notations and observations, we are able to leverage task importance to facilitate task allocation for MTL tasks on the edge. We start by formally define *task allocation* and *MTL tasks on the edge*.

Definition 2. *(Task Allocation) Given a processor set $\boldsymbol{P} = \{p\}$ which consists of a series of processors, the task allocation over $\boldsymbol{P}$ is a binary variable $u_{j,p}$, i.e.,*

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

where a processor is denoted by $p \in \mathbb{N}^+$.

Since each task is assigned to exactly one processor, we have the following constraint:

$$\sum_{p \in \mathbf{P}} u_{j,p} = 1, \ \forall j \in \mathbf{J}. \tag{2}$$

Additionally, the execution time and resource of all tasks assigned to the processor p should satisfy following constraints:

$$\sum_{j \in \mathbf{J}} t_j \cdot u_{j,p} \leq T, \quad \forall p \in \mathbf{P}, \tag{3}$$

$$\sum_{j \in \mathbf{J}} v_j \cdot u_{j,p} \leq V_p, \quad \forall p \in \mathbf{P}, \tag{4}$$

where t_j denotes the execution time of task j; T denotes the time limit; v_j denotes the resource required for task j; V_p denotes the resource capacity of processor 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 3. *(MTL Tasks on the Edge) Given task importance $\mathcal{I}_j$, the execution time and resource limitations of Eq. (2) - (4), an on-edge MTL tasks aims to obtain θ by*

$$\boldsymbol{\theta} = \arg\min \sum_{j \in \boldsymbol{J}} \sum_{p \in \boldsymbol{P}} \mathcal{I}_j \cdot L_j(\theta_j) \cdot u_{j,p}, \ s.t. \ Eq.(2) - (4),$$

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

$$\max_{\boldsymbol{u}} \sum_{j \in \boldsymbol{J}} \sum_{p \in \boldsymbol{P}} \mathcal{I}_j \cdot u_{j,p}, \ s.t. \ Eq. \ (2) - (4),$$

where $\boldsymbol{u} = [u_{j,p}]$ denotes the task-allocation matrix; $\mathcal{I}_j$ can be computed given $\boldsymbol{\theta}$ from Definition 3 and $\boldsymbol{J}$ using Equation 1.

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

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

Proof. First of all, coming back to the traditional *multiply-constrained multiple* Knapsack problem [24], there is a set of $|N|$ items, where each item $i \in \mathbf{N}$ is associated with a weight w_i, a volume o_i and a profit ρ_i. Meantime, there is also a set of $|M|$ knapsacks, where each knapsack $m \in \mathbf{M}$ has a maximum weight capacity W_j and volume capacity O_j. The objective is to infer the positioned knapsack of items $\chi_{i,m} = \arg\max \sum_{i \in \mathbf{N}} \sum_{m \in \mathbf{M}} \rho_i \cdot \chi_{i,m}$, satisfying $\sum_{m \in \mathbf{M}} \chi_{i,m} = 1, \forall i \in \mathbf{N}$.

In our TATIM problem, each data-driven task $j \in \mathbb{N}^+$ can be regarded as an item $i \in \mathbb{N}^+$ and each processor $p \in \mathbb{N}^+$ can be regarded as a knapsack $m \in \mathbb{N}^+$. Similarly, the resource and execution time requirement, i.e., $v_j, t_j \in \mathbb{R}$ for each data-driven task corresponds to the weight and volume $w_i, o_i \in \mathbb{R}$ of each item. The importance of tasks $\mathcal{I}_j \in [0, 1]$ can be regarded as the profit of items $\rho_i \in [0, 1]$. The allocation $u_{j,p} \in \{0, 1\}$ is equal to the positioning $\chi_{i,m} \in \{0, 1\}$. Therefore, our objective $\sum_{j \in \mathbf{J}} \sum_{p \in \mathbf{P}} \mathcal{I}_j \cdot u_{j,p}$ is also equal to the objective of the above Knapsack problem $\sum_{i \in \mathbf{N}} \sum_{m \in \mathbf{M}} \rho_i \cdot \chi_{i,m}$ and so as the other three constraints.

As a result, the TATIM problem is equal to the multiply-constrained multiple Knapsack problem which is NP-C. □

Therefore, our TATIM problem is in general NP-complete. Besides, the complicated computation to solve TATIM needs to be conducted repeatedly due to the time-varying parameter of task importance, making the problem solving challenging.

III. The Clustered Reinforcement Learning (CRL) Model

In this section, we first present an overview of the data-driven approach for task allocation. We then introduce the

background of reinforcement learning. At last, we give a formal definition of the environment-dynamic task allocation and propose our data-driven approach using Clustered Reinforcement Learning (CRL).

A. Data-driven Task Allocation Overview

In the previous section, we show that our TATIM problem is in general 1) NP-complete 2) with time-varying parameters, leading to complicated computations conducted repeatedly. The key to efficiently allocate tasks on the edge lies in the automatically adaptation to varying contexts.

As a remedy to such a challenge, we then propose the data-driven task allocation. Recent years have witnessed a trend of applying data-driven techniques for complicated problems in time-varying environments, including AlphaGO [25], 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.[1]

Formally, given a task set $\mathbf{J}$ 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.,

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

B. Background of Reinforcement Learning

Next, we consider the proper approach to solve 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. Reinforcement learning (RL) is widely suggested to efficiently solve such problems [25], [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.

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

C. 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.

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 decision performance.

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.[2]

In this regard, directly leveraging the RL model can easily mismatch the environment and submitted less important tasks, which leads to poor decision performance [29], [30]. We also conduct an experiment to demonstrate the negative impact. It shows a 46.28% 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.

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

[2]Even in the same scenario, the environment can change over time, due to the accumulating size of training data and the overwritten historical data when the storage is insufficient. An experiment in the previous section also indicates the fluctuation between historical and current task importance.

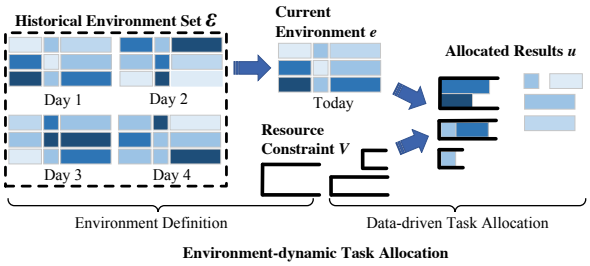

Fig. 6: The illustration of environment-dynamic task allocation.

environment $\mathcal{E}$. We define the historical environment $\mathcal{E}$ as the collection of environment e, i.e.,

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

where e_j denotes the corresponding environment. For better understanding, a formal mathematical structure of the environment is available in the following Section III-D.

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.

D. 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.

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.

State space. We represent the state, which is the current task selection of the system. More 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.

Algorithm 1 Clustered Reinforcement Learning (CRL)

Training Phase:

1: $\boldsymbol{\mathcal{E}} \leftarrow$ historical environment $s_0 \leftarrow$ initial state $Z \leftarrow$ current scenarios and configuration settings.
2: $e \leftarrow$ *EnvironmentDefinition*($\boldsymbol{\mathcal{E}}$,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 u**

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 $\{1, 2, \cdots, M\}$, where $a = j$ means to conduct the j^{th} task for the current processor in the current time step. Hence, the action space is defined as follows:

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

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

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 $\sum_{j \in \mathbf{J}} I_j$ only if the agent reaches the terminal state (i.e., all tasks in the current system are assigned accordingly), where $\mathbf{J}$ is the set of tasks currently in the system. Otherwise, the reward was set to 0. Hence,

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

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.

Optimization. With the above key elements, we leverage Deep Q-learning $Q(s, a; \theta, \mathbf{J})$ [31], 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

$$\mathbf{u} \leftarrow \mathcal{F}_1(\mathbf{J}, \mathcal{X}) = \mathcal{F}_1(\mathbf{J}, (e, s_0)) = Q(s, a; \theta, \mathbf{J}). \tag{5}$$

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

Convergence Analysis. It has been proven that the Deep Q-learning technique will gradually converge to the optimal policy under stationary Markov decision process (MDP) environment and sufficiently small learning rate [32]. Hence, the proposed CRL predictor will converge to the optimal policy when (i) the environment evolves as a stationary, memoryless Semi-Markov Decision Process and (ii) the DNN is sufficiently accurate to return the action associated with the optimal $Q(s, a)$ estimate.

IV. CRL-based Local Process with a Cooperative Learning Framework

In this section, we first show that the CRL model should not be directly applied due to the simulation limitations. We then propose a Cooperative Learning approach based on both CRL and Support Vector Machine (SVM) to fully leverage the simulated and real-world data. At last, we briefly introduce the design of the SVM model.

A. Introducing Local Process with Cooperative Learning

In the previous section, we present a detailed design of the CRL model. 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 [30]. 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% reduction of performance.

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 general process with a huge environment definition data, and 2) a local process 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:

$$\mathcal{F}(\mathbf{J}, \mathcal{X}) = \mathcal{F}(\mathbf{J}, (\mathcal{C}, \mathcal{R})) = w_1 \mathcal{F}_1(\mathbf{J}, \mathcal{C}) + w_2 \mathcal{F}_2(\mathbf{J}, \mathcal{R}), \quad (6)$$

where $\mathcal{F}_1(\cdot)$ and $\mathcal{F}_2(\cdot)$ denote the general process and local process; 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., $\mathbf{u} \leftarrow \mathcal{F}(\mathbf{J}, (\mathcal{C}, \mathcal{R}))$.

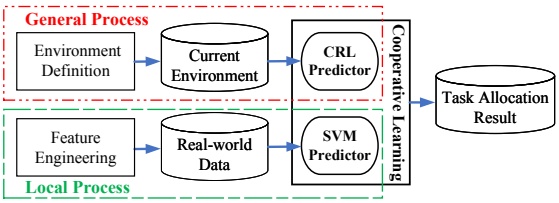

Fig. 7: Framework of Cooperative Learning for task allocation.

B. Brief Design of Local Process

For our cooperative learning, We have discussed the first component of the general process using CRL in the previous section. Now we introduce the design of the local process.

As for the second component of local process, 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:

$$\mathbf{u} \leftarrow \mathcal{F}_2(\mathbf{J}, \mathcal{X}) = SVM(\mathcal{X}; w, \mathbf{J}) \quad (7)$$

where w denotes its parameter vector.

In the following, we are to briefly introduce the design of loss function and feature engineering of the local process of SVM predictor.

C. Loss Function of Local Process

Formally, let $\mathcal{R}$ be the feature spaces of the real-world data, sample $k \in \mathcal{R}$ usually consists of two parts. One is a vector x_k that is regarded as the input feature values; the other is a scalar y_k that is the desired output of the model. Then, we define the loss function $\mathcal{L}_k(w)$ of our SVM predictor $\mathcal{F}_2(\cdot)$ as follows:

$$\mathcal{L}_k(w) = \frac{1}{2}\|w\|^2 + \frac{1}{2}\max\{0; 1 - y_k w^T x_k\}^2, \quad (8)$$

where w denotes its parameter vector; w^T denotes the transpose of w and $\|\cdot\|^2$ denotes the $\mathcal{L}^2$ norm. Finally, our optimization process is to find the optimal parameter vector w^* that minimize the loss function $\mathcal{L}_k(w)$ on a collection of training dataset $\mathcal{R}$. Hence, $w^* = \arg\min \frac{1}{|\mathcal{R}|} \sum_{k \in \mathcal{R}} \mathcal{L}_k(w)$.

D. Feature Engineering of Local Process

In most real-world scenarios, e.g., industry domain, it is either costly or even impossible to obtain the data on environments and configurations of tasks; we usually do not have the luxury to obtain enormous data where local predictor can be trained with irrelevant features automatically eliminated. As such the challenge is to select the proper feature set for our local predictor. Thus, we propose a domain-assisted feature engineering approach which uses domain knowledge to create features relevant to the problem at hand. The feature set consists of the following two types of features, naming domain features and general features, respectively. We list our features in Table I.

General features. Overall, the general features should have some universality and can be easily applied to other scenarios.

TABLE I: The description of features in the local process of SVM predictor.

Feature Type	Feature	Description
General	Past Success	The number of cases that a task is selected in the optimal decision in the past
	Prediction Accuracy	The similarity between the predicted (e.g., COP) and the real for a task in the past
Domain	Building	The building that the operating chiller is deployed in
	Model Type	The model of the operating chiller
	Operating Power	The power measured by kilowatts for the operating chiller
	Weather Condition	The description of weather condition in a time interval
	Outdoor Temperature	The outdoor temperature measured by Celsius in a time interval
	Latest Cooling Load	The last recorded cooling load assigned on this chiller
	Water Mass Flow Rate	The mass of water flowing per second, measured by kg/s
	Water Temperature Difference	The difference between the returned and supplied chilled water temperature

More specifically, it should reflect the decision performance of a data-driven task in the statistical view over the historical data. We consider the following two factors: 1) Past Success which refers to the number of cases that a task is selected in the optimal decision in the past. In general, if the industrial demand, configuration, and environment do not change too much, a task selected in the past will likely be selected again in the current period; 2) Prediction Accuracy which is the similarity between the predicted performance and the real performance for a given task in the past. This is also an important factor since an accurate prediction indicates our final decision making can be more reliable.

Domain features. Obviously, the domain features are closely related to specific scenarios. Recall the previous example in the driver-less car, driving operation decision is critical to driver-less car safety, which consists of the following specific characteristics such as engine status, speed, radar data, GPS data and etc. While in this paper, we focus on the industrial field where chiller sequencing is a common but highly important industrial operation in energy-efficient buildings, which consists of the following specific features: 1) Meteorological information such as temperature and weather drive the cooling demand imposed on the chillers; 2) Mechanical information such as the model type, building, operating power, water temperature difference, flow rate, and the recent cooling load are used to capture the chiller specific characteristics.

V. Performance evaluation

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

A. Background of the Simulated Green-building System

Before the setting of our simulation, we begin by introducing the real-world green-building system to be simulated [22]. The green-building system conducting industrial operations assisted with machine learning. It is deployed for one week in January 2019, in a high-rise office building in Hong Kong which serves more than three thousand people. As a facility in building, chiller is a machine generate cooling power in commercial buildings and chiller operation is a significantly important operation, which aims to select run-time configurations of the chiller so that the overall system serves the cooling demand while minimizes the energy consumption.

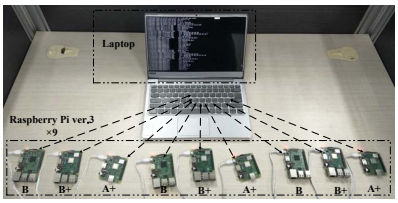

Fig. 8: The Network Topology and Hardware Choice in the Experiments, where Raspberry Pi are with model types of A+, B, and B+.

In the green-building system, the equipment of chillers, pumps, air-handling unit, and cooling tower differ greatly in operation, maintenance, and services. The data of each equipment in the chiller plant are captured and transmitted by 10 edge nodes, including one operation node conducting and recording operations, and nine sensing nodes collecting run-time data. To process data from different types of equipment, a centralized approach is leveraged, where edge node transmits data to the controller, and controllers are responsible for task allocation and decision making for the edge nodes. Finally, the operation node conducts data-driven COP prediction and send control sequences to devices. Other sensing nodes without computation power are merely used to collect data. The simulation will be conducted according to the above setting to better simulate the real-world green-building system.

B. Experiment Setup

For generating transfer learning tasks, we use a real-world *building operation* dataset released in [22], 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 transfer-learning 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

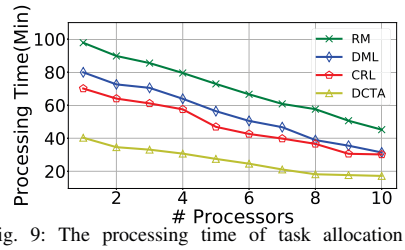

Fig. 9: The processing time of task allocation system with different number of processors.

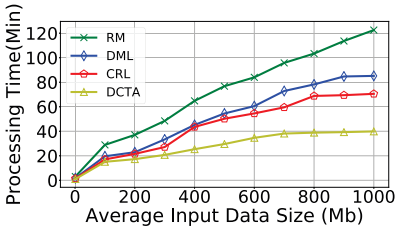

Fig. 10: The processing time of task allocation system with different data input sizes.

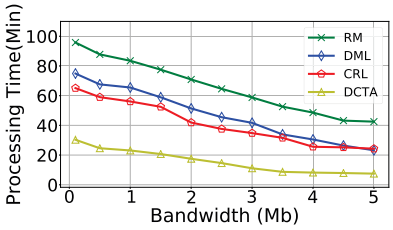

Fig. 11: The processing time of task allocation system with different bandwidth limits.

computation time of the Raspberry Pi A+ is 4.75×10^{-7} s/bit, which is based on the settings from [33].

C. 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 [33]. In other words, tasks are randomly assigned.
- **Distributed Machine Learning (DML)** distributes tasks to multiple computing nodes, e.g., allocating the training iteration either to edge devices or to the cloud [34].
- **Clustered Reinforcement Learning (CRL)** conducts 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. For a task allocation method, the ability to provide credible decision performance under execution time limits is crucial to all stakeholders. Time is always the first concern, and we measure the Processing Time (PT), which is the time the main device needs to partition the application and receive the output of the decision making. Formally,

$$PT = t_s - t_c,$$

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

D. Experiment Results

Result on Processing Time. Figure 9 shows the processing time as a function of processors. Consistent with our intuition, as the number of processors increases, the processing time of the above allocation 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, Figure 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.

VI. Related Work

Task Allocation has been intensively researched in cloud and mobile cloud computing systems [35]–[39]. Recent years have witnessed great prospects exhibited down to the edge, e.g., from OpenCL (2008) [40] to AWS IoT Greengrass (2017) [41] and Microsoft Azure IoT Edge (2018) [42]. 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.

First, task allocation in many distributed machine learning systems [34], [43]–[45] 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 [46] identifies a strategy in a fine-grained layer level between edge and cloud. A similar approach presented in [47] 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 [48], [49]. Examples also include reducing the energy consumption of edge device while predefined delay constraint is satisfied [11] [12], finding a proper trade-off between the energy consumption and the execution delay [13], 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 [50], [51]. Examples include intelligent logistics [26], code optimization [52], [53], task scheduling [54], [55]. 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 [56] and optimizing application memory usage [57]. This work is the first attempt in applying ensemble techniques to optimize task allocation of MTL with task importance on the edge.

VII. Discussion

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

Leveraging Existing Edge Nodes. Admittedly, it is possible to fully redeploy hardware after introducing data-driven techniques and provide strong computational power within a low purchasing budget [58]. However, for the scalability purpose, only an incremental software installment for the real-world system is conducted for edge computing, with a minimal revision of hardware for the current system. That is to say, only the current commercial off-the-shelf components in the system are leveraged for computation. The proposed system avoids deploying any additional equipment within the real-world system, e.g., without adding high-performance servers which may be in low purchasing price. Such a design ensures privacy and enables low-intrusive or even non-intrusive installment, which are both critical for scalability of the proposed approach. We understand that 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 for the scalability purpose. As for the case where powerful edge nodes are available, the proposed approach can be easily extended to support the case by changing the budget constraints in the problem formulation.

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 transfer learning.

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 machine-learning tasks, and 2) there is no strong pre- and post-dependency, which is also a prerequisite for performing multi-task transfer learning. Thus, those cases 1) under single-task settings, or 2) under multi-task settings but with the sequential dependency between tasks, are beyond the scope of this paper. It would be an interesting future work to extend our approach to those scenarios.

VIII. Conclusion

Edge computing for machine learning has become a heated research topic. In this paper, we focus on task allocation for MTL scenarios by introducing task importance and make the following contributions. First, we reveal that it is important to measure the impact of tasks on overall 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 and accelerates the overall learning process. We formally define the TATIM problem of task allocation with task importance for MTL. Second, we show TATIM is in fact a variant of Knapsack problem, which is NP-complete. Accordingly, we propose a Data-driven Cooperative Task Allocation (DCTA) approach to accelerate the computational efficiency without performance degradation. Third, we evaluate the performance of our DCTA approach by applying it to a real-world industrial operation (e.g., AIOps) scenario. Experiments show that our DCTA approach can save 3.24 times of processing time compared to the state-of-the-art. We believe that our DCTA approach offers an effective and practical mechanism for reducing the required resource associated with performing MTL on edge devices.

REFERENCES

[1] M. L. Hutchinson, E. Antono, B. M. Gibbons, S. Paradiso, J. Ling, and B. Meredig, "Overcoming data scarcity with transfer learning," *arXiv preprint arXiv:1711.05099*, 2017.

[2] C. Yuan, W. Hu, G. Tian, S. Yang *et al.*, "Multi-task sparse learning with beta process prior for action recognition," in *IEEE CVPR*, 2013.

[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 *IEEE ICASSP*, 2015, pp. 4460–4464.

[4] S. Emrani, A. McGuirk, and W. Xiao, "Prognosis and diagnosis of parkinson's disease using multi-task learning," in *ACM SIGKDD*, 2017.

[5] J. Zhou, L. Yuan, J. Liu, and J. Ye, "A multi-task learning formulation for predicting disease progression," in *ACM SIGKDD*, 2011.

[6] T. Idé, D. T. Phan, and J. Kalagnanam, "Multi-task multi-modal models for collective anomaly detection," in *IEEE ICDM*, 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 *Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on*. IEEE, 2017, pp. 1–6.

[8] B. Hong and V. Prasanna, "Adaptive allocation of independent tasks to maximize throughput," *IEEE Transactions on Parallel & Distributed Systems*, no. 10, pp. 1420–1435, 2007.

[9] Y. Jiang, Y. Zhou, and Y. Li, "Reliable task allocation with load balancing in multiplex networks," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 10, no. 1, p. 3, 2015.

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

[11] S. Cao, X. Tao, Y. Hou, and Q. Cui, "An energy-optimal offloading algorithm of mobile computing based on hetnets," in *IEEE ICCVE*, 2015.

[12] Y. Wang, M. Sheng, X. Wang *et al.*, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[13] Y. Mao, J. Zhang, S. Song *et al.*, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *IEEE GLOBECOM*, 2016.

[14] Y. Geng, Y. Yang, and G. Cao, "Energy-efficient computation offloading for multicore-based mobile devices," in *IEEE INFOCOM*, 2018.

[15] P. Yang and J. He, "Heterogeneous representation learning with structured sparsity regularization," in *IEEE ICDM*, 2016.

[16] K. Lin *et al.*, "Multi-task feature interaction learning," in *SIGKDD*, 2016.

[17] P. Gong, J. Ye, and C. Zhang, "Robust multi-task feature learning," in *ACM SIGKDD*, 2012, pp. 895–903.

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

[19] K. Lin and J. Zhou, "Interactive multi-task relationship learning," in *IEEE ICDM*, 2016, pp. 241–250.

[20] D. Oyen, T. Lane *et al.*, "Leveraging domain knowledge in multitask bayesian network structure learning." in *AAAI*, 2012.

[21] D. Isele, M. Rostami, and E. Eaton, "Using task features for zero-shot knowledge transfer in lifelong learning." in *IJCAI*, 2016.

[22] Z. Zheng, Q. Chen, C. Fan, N. Guan, A. Vishwanath, D. Wang, and F. Liu, "Data driven chiller sequencing for reducing hvac electricity consumption in commercial buildings," in *ACM e-Energy*, 2018.

[23] A. R. Zamir, A. Sax, W. B. Shen, L. J. Guibas, J. Malik, and S. Savarese, "Taskonomy: Disentangling task transfer learning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

[24] H. Shachnai *et al.*, "On two class-constrained versions of the multiple knapsack problem," *Algorithmica*, vol. 29, no. 3, pp. 442–467, 2001.

[25] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[26] X. Li, "Development of intelligent logistics in china," in *Contemporary Logistics in China*. 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 *IEEE ICRA, 2018*, pp. 1–7.

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

[29] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, 2015, pp. 455–468.

[30] H. Hu, L. Chen, B. Gong, and F. Sha, "Synthesize policies for transfer and adaptation across tasks and environments," in *Advances in Neural Information Processing Systems*, 2018, pp. 1176–1185.

[31] N. Liu, Z. Li, J. Xu, Z. Xu, S. Lin, Q. Qiu, J. Tang, and Y. Wang, "A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning," in *IEEE ICDCS*, 2017.

[32] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[33] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading decision and resource allocation for multi-user multi-task mobile cloud," in *Communications (ICC), 2016 IEEE International Conference on*, 2016, pp. 1–6.

[34] S. Teerapittayanon, B. McDanel *et al.*, "Distributed deep neural networks over the cloud, the edge and end devices," in *IEEE ICDCS*, 2017.

[35] F. Liu, P. Shu, and J. C. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE transactions on computers*, vol. 64, no. 11, pp. 3051–3063, 2015.

[36] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[37] T. Zhang, X. Zhang, F. Liu, H. Leng, Q. Yu, and G. Liang, "etrain: Making wasted energy useful by utilizing heartbeats for mobile data transmissions," in *2015 IEEE 35th International Conference on Distributed Computing Systems*. IEEE, 2015, pp. 113–122.

[38] J. Park, H. Byun, and J.-R. Lee, "Bio-inspired load-balancing framework for loosely coupled heterogeneous server systems," *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3280–3292, 2016.

[39] Y. Jin, F. Liu, X. Yi, and M. Chen, "Reducing cellular signaling traffic for heartbeat messages via energy-efficient d2d forwarding," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1301–1311.

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

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

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

[43] K. Hsieh, A. Harlap, N. Vijaykumar *et al.*, "Gaia: Geo-distributed machine learning approaching lan speeds." in *NSDI*, 2017.

[44] E. P. Xing, Q. Ho, W. Dai, J.-K. Kim *et al.*, "Petuum: A new platform for distributed machine learning on big data," in *ACM SIGKDD*, 2015.

[45] M. Li, D. G. Andersen *et al.*, "Communication efficient distributed machine learning with the parameter server," in *NIPS*, 2014.

[46] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 615–629, 2017.

[47] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," *arXiv preprint arXiv:1802.03835*, 2018.

[48] 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 *IEEE INFOCOM*, 2019.

[49] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *IEEE INFOCOM*, 2019.

[50] F. Samreen, Y. Elkhatib, M. Rowe, and G. S. Blair, "Daleel: Simplifying cloud instance selection using machine learning," *arXiv preprint arXiv:1602.02159*, 2016.

[51] Z. Wang and M. O'Boyle, "Machine learning in compiler optimization," *Proceedings of the IEEE*, 2018.

[52] S. Chen, J. Fang, D. Chen *et al.*, "Adaptive optimization of sparse matrix-vector multiplication on emerging many-core architectures," 2018.

[53] C. Cummins, P. Petoumenos, Z. Wang, and H. Leather, "End-to-end deep learning of optimization heuristics," in *IEEE PACT*, 2017.

[54] J. Ren, L. Gao, H. Wang *et al.*, "Optimise web browsing on heterogeneous mobile platforms: a machine learning based approach," 2017.

[55] S. Chen, J. Fang, D. Chen, C. Xu, and Z. Wang, "Optimizing sparse matrix-vector multiplication on emerging many-core architectures," *arXiv preprint arXiv:1805.11938*, 2018.

[56] M. K. Emani and M. O'Boyle, "Celebrating diversity: a mixture of experts approach for runtime mapping in dynamic environments," in *ACM SIGPLAN Notices*, vol. 50, no. 6. ACM, 2015, pp. 499–508.

[57] 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 *ACM Middleware*, 2017, pp. 95–108.

[58] P. K. Agyapong *et al.*, "Design considerations for a 5g network architecture," *IEEE Communications Magazine*, vol. 52, no. 11, 2014.