# Toward Cost-Efficient Online Transfer Learning in Distributed Cloud-Edge Networks

Konglin Zhu, *Member, IEEE,* Fei Wang, Lei Jiao, *Member, IEEE,* Yulan Yuan, Xiaojun Lin, *Fellow, IEEE,* Lin Zhang, *Member, IEEE*

***Abstract*—Transfer learning leverages existing models to help train new models, rather than training the new models from scratch. Unfortunately, realizing transfer learning in distributed cloud-edge networks faces critical challenges such as online training, uncertain network environments, time-coupled control decisions, and the balance between resource consumption and model accuracy. In this paper, targeting classification tasks, we study the settings of both homogeneous and heterogeneous transfer learning in cloud-edge networks via orchestrating model placement, data dispatching, and inference aggregation. We formulate non-linear mixed-integer programs of long-term cost optimization over consecutive time slots, and design polynomial-time online algorithms by exploiting the real-time trade-off between preserving previous control decisions and applying new control decisions. Our approaches produce new models by combining the existing pre-trained offline models and the online models that are continuously updated based on the inference results of data samples arriving in streams. We rigorously prove that our approaches only incur the number of inference mistakes no greater than a constant times that of the single best model in hindsight, and achieve constant competitive ratios for the total cost. Evaluations have confirmed the superior performance of our approaches compared to other state-of-the-art methods upon real-world data traces, under text classification transfer learning tasks.**

***Index Terms*—Edge computing, edge AI, transfer learning, online optimization.**

## I. Introduction

MOBILE communication networks are shaping the new paradigm of how users can explore and utilize Artificial Intelligence (AI). The 5G networks often consist of centralized gigantic data centers (referred to as "clouds") in the core [2], [3], and distributed cellular base stations with co-located micro computing servers (referred to as "edges") in closer proximity to the end users. AI models can be thus trained in the cloud using abundant data and then dispatched to the edges to serve the end users' inference requests [4], [5].

One fundamental problem of the AI in this scenario is that, as time goes, AI models often have varying or even decaying accuracies. First, the underlying data distribution, i.e., the relation between data's features and labels, may drift due to the non-stationary nature of the data and the environment [6], [7]. Under such "concept drifts", AI models which capture the relation between existing data's features and labels may not work for the new data. Second, the underlying data distribution may differ across communities or areas and AI models trained by different data may reflect different relations between data's features and labels. As users move, their requests, which used to be served by the models at a previous edge, may not be properly resolved by the models at the new edge [8].

Transfer learning [9], [10] seems a promising solution to this problem. As existing AI models become less accurate, rather than dropping them and building new models from scratch, one can leverage the existing models to help build the new models, i.e., transferring "knowledge" from existing models (called *offline classifiers* in this paper) to the combination of those existing models and the new models (called *online classifiers*), especially when the new data alone may be insufficient for training new models or when the underlying data distribution may possess periodical patterns. Yet, this approach confronts multiple critical challenges in distributed cloud-edge networks.

First, it is non-trivial to design and realize transfer learning in a distributed manner. It is desired to keep data samples within the local edge networks without sending them to the remote cloud due to privacy and performance (e.g., traffic localization); yet, extracting knowledge from each of the offline classifiers that may reside across different edges to train the online classifier(s) upon data samples that dynamically arrive requires to make a comprehensive set of control decisions, such as classifier placement, data sample dispatching, inference aggregation, and model weights update. It is challenging to navigate the trade-offs among these intertwined decisions.

Second, it remains a non-trivial problem to implement distributed transfer learning cost-efficiently in an online manner in the uncertain cloud-edge environment. As the operational cost of edges, the delay between edges, and the available capacity of each edge vary unpredictably [8], [11], we need to control the system in real time to pursue the long-term optimization. This is particularly hard due to time-coupled

This work was supported in part by the National Key Research and Development Program of China (2023YFB2704500), the Fundamental Research Funds for the Central Universities (2024ZCJH07), the Beijing Natural Science Foundation (4222033), the Beijing Advanced Innovation Center for Future Blockchain and Privacy Computing, and the U.S. National Science Foundation (CNS-2047719, CNS-2225949, CNS-2113893, and CNS-2225950). A preliminary version of this work appeared in the Proceedings of the 41st IEEE International Conference on Computer Communications (INFOCOM), May 2-5, 2022 [1] (Corresponding author: Lei Jiao.)

K. Zhu, F. Wang, and L. Zhang are with the School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: {klzhu, feiwang, zhanglin}@bupt.edu.cn).

L. Jiao is with the Center for Cyber Security and Privacy, University of Oregon, Eugene, OR 97403, USA (e-mail: jiao@cs.uoregon.edu).

Y. Yuan is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology (Guangzhou), Guangzhou 511458, China (e-mail: yyuan202@connect.hkust-gz.edu.cn).

X. Lin is with the Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA (e-mail: linx@purdue.edu). Part of the work of Xiaojun Lin was completed when he moved to The Chinese University of Hong Kong.

decisions [12], [13], caused by selecting the edge to download each existing offline classifier from the cloud or host the new online classifier(s) to be created. For instance, hosting a classifier on a local edge now will save "start-up" cost of downloading the classifier and re-instantiating the edge environment if this same classifier is also needed here in the next time slot, but will waste the operational cost if this classifier turns out to be unwanted (e.g., if there exists another cheaper edge) in the next time slot.

Third, the different *homogeneous* and *heterogeneous* online transfer learning settings require different solutions over the cloud-edge networks. Transitioning from homogeneous to heterogeneous is not a simple extension, but rather a fundamental shift in design and optimization methodology. In the homogeneous setting, each classifier can access the whole feature space of the data samples. In contrast, in the heterogeneous setting, each classifier can only have access to different part of the feature space of the data samples [14], changing the transfer learning process entirely by requiring multiple online classifiers to be trained simultaneously; this also adds additional dimensions to the system's control-decision space, e.g., we now need to jointly control where the inference aggregation happens, considering additional operating cost and start-up cost. Due to all such differences, we need to treat these two online transfer learning settings separately.

Existing research falls insufficient for addressing the aforementioned challenges. Prior works [8], [14]–[18] studied transfer learning's performance, efficiency, and accuracy, but never considered resource or cost overhead, not to mention in distributed cloud-edge environments or in an online manner. Others [19]–[25] focused on resource utilization, job scheduling, and various optimizations of cloud and edge networks and applications. However, to the best of our knowledge, distributed transfer learning remains unexplored in existing studies. See Section VII for more detailed discussions.

To the best of our knowledge, this work is the first to design and optimize distributed online transfer learning in cloud-edge networks. In this paper, we make several contributions:

- We formulate cross-layer long-term joint cost optimization for homogeneous and heterogeneous online transfer learning, incorporating operational, start-up, delay, and inference error costs. The NP-hard problems make no assumptions on input dynamics or classifier types/forms.
- For homogeneous settings, we design four polynomial-time algorithms, i.e., Algorithms 1-4 that work jointly using primal-dual techniques and online adjustments, proven to bound mistakes against the best single classifier and achieve a parameterized-constant competitive ratio for the total cost against the offline optimum.
- For heterogeneous settings, we develop Algorithms 5-8, addressing additional start-up costs and split feature spaces via aggregation-first inference and paired online classifiers, with proven worst-case guarantees.
- Comprehensive evaluations on real-world datasets (e.g., traffic workload, text classification tasks) show that our approaches reduce total cost by 40%–60%, achieve lower mistake rates, and operate efficiently within seconds per time slot, outperforming existing methods.

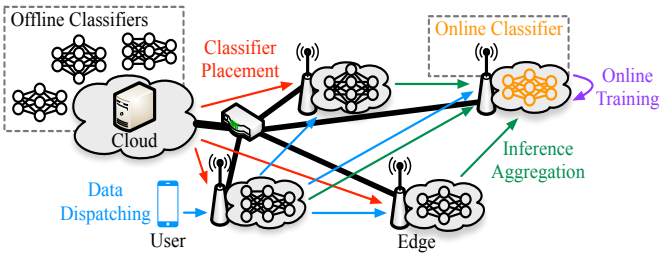



**Fig. 1:** Scenario for Homogeneous Online Transfer Learning

The rest of this paper proceeds as follows. Sections II, III, and IV elaborate our problem modeling, algorithm design, and performance analysis for the homogeneous online transfer learning in the distributed cloud-edge networks. Section V is on the heterogeneous online transfer learning, focusing on the differences from the homogeneous setting. Section VI demonstrates the trace-driven experiments and results. Section VII discusses the related work and Section VIII concludes.

## II. Models and Problem Formulation

### A. System Models

We summarize our notations in Table I.

***Cloud-Edge Networks:*** We study the system over a series of consecutive time slots $[T] = \{1, ..., T\}$, corresponding to our decision-making frequency. As shown in Fig. 1, we consider a set of geographically distributed edges $[I] = \{1, ..., I\}$, where an "edge" refers to a cellular base station or a WiFi access point equipped with a micro data center or a server cluster. The edges are connected to one another, and also to a common cloud via backhaul networks. For $i, j \in [I]$ and $t \in [T]$, we use $d_{ij}^t$ to denote the delay between the edge $i$ and the edge $j$ at the time slot $t$, and use $D_i^t$ to denote the available capacity of the edge $i$ at the time slot $t$. We also consider that this edge environment provides a virtual machine (VM) or a container to host and run each offline or online classifier, which will be further elaborated below.

***Offline and Online Classifiers:*** The cloud maintains a set $[K] = \{1, ..., K\}$ of pre-trained "offline" classifiers that are to be downloaded to the edges to serve users with ultra-low latency and used to train a single "online" classifier being updated continuously to accommodate any concept drift. For $k \in [K]$, $i \in [I]$, and $t \in [T]$, we use $a_{ki}^t$ to denote the operational cost (e.g., electricity consumption) of hosting offline classifier $k$ on edge $i$ at $t$, use $b_i^t$ to denote the operational cost of hosting the online classifier on edge $i$ at $t$, use $c_{ki}$ to denote the "start-up" cost of offline classifier $k$ on edge $i$, including the cost (e.g., traffic or bandwidth consumption) of downloading offline classifier $k$ from the cloud to edge $i$ and the cost (e.g., lead time, system oscillation) of booting and preparing the VM or container on edge $i$, and use $c_i$ to denote the "start-up" cost of the online classifier on edge $i$, which only includes booting and preparing the VM or container on edge $i$. The online classifier is directly created on edge using the training data samples per time slot, rather than being downloaded from the cloud. We also use $f^k(\cdot)$ to denote the "decision function" of the offline classifier $k$. Besides, we use $f_m^t(\cdot)$ to denote the decision function of the online classifier that is trained at the time slot $t$ for the data

**TABLE I:** Notations

| Inputs | Descriptions |
|---|---|
| $[I]$ | Set of edges |
| $[K]$ | Set of classifiers |
| $[T]$ | Set of time slots |
| $[M]^t$ | Data samples at the time slot $t$ from users |
| $D_i^t$ | Available capacity of the edge $i$ at the time slot $t$ |
| $p_m^t$ | Feature values for the data sample $m$ |
| $q_m^t$ | Ground-truth label for the data sample $m$ |
| $a_{ki}^t$ | Operational cost of hosting offline classifier $k$ on the edge $i$ at $t$ |
| $b_i^t$ | Operational cost of hosting the online classifier on the edge $i$ at $t$ |
| $c_{ki}$ | Start-up cost of downloading offline classifier $k$ from the cloud to the edge $i$ |
| $c_i$ | Start-up cost of the online classifier at edge $i$ |
| $f^k(\cdot)$ | Decision function of the offline classifier $k$ |
| $f_m^t(\cdot)$ | Decision function of the online classifier that is trained at the time slot $t$ for the data sample $m$ |
| $d_{ij}^t$ | Delay between the edge $i$ and the edge $j$ at the time slot $t$ |

| Decisions | Descriptions |
|---|---|
| $x_{ki}^t$ | Whether or not the offline classifier $k$ is downloaded and hosted on the edge $i$ at the time slot $t$ |
| $y_i^t$ | Whether or not the online classifier is trained and hosted on the edge $i$ at the time slot $t$ |
| $z_{kim}^t$ | Weight for the offline classifier $k$ on the edge $i$ for the data sample $m$ at the time slot $t$ |
| $w_{im}^t$ | Weight for the online classifier $k$ on the edge $i$ for the data sample $m$ at the time slot $t$ |
| $u_{mi}^t$ | Whether or not to transfer data sample $m$ from the edge where it arrives to the edge $i$ at $t$ |
| $v_{ij}^t$ | Whether or not to transfer the decision results of offline classifiers from the edge $i$ to the edge $j$ at $t$ |

sample $m$. Note that we write $f_m^t(\cdot)$ instead of $f^t(\cdot)$, because the single online classifier is being updated per data sample $m$ during transfer learning, further elaborated as below.

***Data Samples:*** We use $[M]^t = \{0, ..., M_t\}$ to denote the data samples that arrive at the system at the time slot $t$ from users. Each single data sample $m \in [M]^t$ is represented as $(p_m^t, q_m^t)$, where $p_m^t$ refers to its feature values and $q_m^t$ refers to its ground-truth label. Without loss of generality, we assume $q_m^t \in \{-1, 1\}$, $\forall m$, $\forall t$. We emphasize that $q_m^t$ is only observable right after we conduct the inference for $m$ using our offline and online classifiers. We also note that any data sample $m$ may arrive at one edge but be dispatched to a different edge to do the inference. We use $d_{mi}^t$ to represent the delay between the edge where the data sample $m$ arrives and the edge $i$ at $t$.

***Distributed Online Transfer Learning:*** At the time slot $t$, as the data sample $m$ arrives at the system, we design distributed transfer learning that works as follows, also in Fig. 2.

- *Step 1:* The data sample $m$ with its feature value $p_m^t$ is dispatched to every edge that has the offline classifiers or the online classifier. Note that it only needs to be dispatched to an edge once even if an edge hosts multiple classifiers. Receiving $p_m^t$, every offline classifier $k$ computes $f^k(p_m^t)$ and the online classifier computes $f_m^t(p_m^t)$.
- *Step 2:* The decisions from the offline classifiers are sent to the edge that maintains the online classifier with all the weights of all the classifiers to compute the inferred label as $F_m^t(p_m^t) = sign(\sum_k \sum_i z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))$ [32], where $sign(\cdot)$ returns 1 for a positive value, $-1$ for a negative value, and 0 for 0; $z_{kim}^t$ is the weight for the offline classifier $k$ on the edge $i$ for $p_m^t$; and $w_{im}^t$ is the weight for the online classifier on the edge $i$ for $p_m^t$.
- *Step 3:* The inferred label $F_m^t(p_m^t)$ is then sent to the edge where the data sample $m$ arrives originally, and is further sent back to the user.
- *Step 4:* The ground-truth label $q_m^t$ arrives at that same edge, and is dispatched to the edge that has the online classifier and all the weights. There, the weight for each classifier is updated and the decision function itself of the online classifier is also updated, i.e., $f_m^t(\cdot)$ is updated to $f_{m+1}^t(\cdot)$ using the received decision results of offline classifiers (see details in Section IV-B).

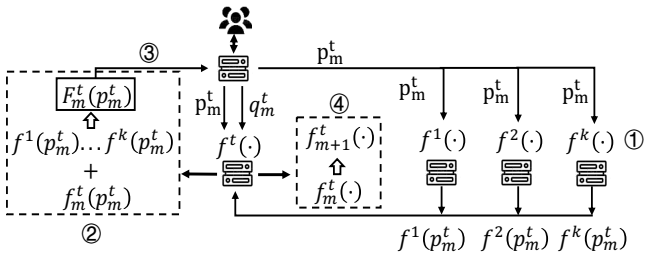



**Fig. 2:** Transfer Learning per Data Sample

Our distributed transfer learning then proceeds to the next data sample $m+1$ at the time slot $t$. In the above process, we "transfer knowledge" from the existing offline decision functions $f^k(\cdot)$, $\forall k$ to the new decision function $F_m^t(\cdot)$ which is a combination of $f^k(\cdot)$, $\forall k$ and the online decision function $f_m^t(\cdot)$ being trained upon each data sample.

***Control Decisions:*** We concentrate on making the following control decisions in this paper. We use $x_{ki}^t \in \{1, 0\}$ to denote whether or not the offline classifier $k$ is downloaded from the cloud and hosted on the edge $i$ at the time slot $t$. We use $y_i^t \in \{1, 0\}$ to denote whether or not the online classifier is trained and hosted on the edge $i$ at the time slot $t$. We $u_{mi}^t \in \{1, 0\}$ to denote whether or not to transfer data sample $m$ from the edge where it arrives to the edge $i$ at $t$, and use $v_{ij}^t \in \{1, 0\}$ to denote whether or not to transfer the decision results of offline classifiers from the edge $i$ to the edge $j$ at $t$. We also use $z_{kim}^t, w_{im}^t \in [0, 1]$ to denote the weight for the offline classifier $k$ on the edge $i$ and the weight for the online classifier on edge $i$, respectively, for the data sample $m$ at the time slot $t$, as described above.

***Cost of Transfer Learning:*** The cost of distributed homogeneous transfer learning at any individual time slot $t$ consists of multiple components: (1) the operational cost of hosting classifiers on edges: $\sum_k \sum_i a_{ki}^t x_{ki}^t + \sum_i b_i^t y_i^t$; (2) the start-up cost of downloading classifiers from the cloud to edges and preparing the VMs or containers on edges: $\sum_k \sum_i c_{ki} \left[x_{ki}^t - x_{ki}^{t-1}\right]^+ + \sum_i c_i \left[y_i^t - y_i^{t-1}\right]^+$, where $[\cdot]^+ = \max\{\cdot, 0\}$; (3) the performance overhead incurred by running distributed transfer learning across edges, including the delay of dispatching data samples $\sum_m \sum_i d_{mi}^t u_{mi}^t$, the delay of transmitting decisions of offline classifiers $\sum_i \sum_j d_{ij}^t v_{ij}^t$, and the delay of transmitting the inferred label and the ground-truth label $2 \cdot \sum_m \sum_i d_{mi}^t y_i^t$. Note that the inferred label (or the ground-truth label) is only sent from (or to) the edge $i$ that has the online classifier.

***Mistakes of Transfer Learning:*** We consider the number of "mistakes" to measure the quality or accuracy of transfer

learning [14], i.e., the number of occurrences where the inferred label does not match the ground-truth label. We denote the number of mistakes for all data samples of any single time slot $t$ as $\sum_m \mathbb{I}\{sign[q_m^t \cdot (\sum_k \sum_i z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))] < 0\}$, where $\mathbb{I}\{\cdot\} = 1$ if the inequality condition contained is true and $\mathbb{I}\{\cdot\} = 0$ if not.

### B. Problem Formulation, Challenges, and Goal

***Problem Formulation:*** We minimize the sum of (i) the long-term total cost of transfer learning and (ii) the long-term total number of mistakes of transfer learning over time:

$$\begin{aligned}
\text{Min} \quad & H_1 = \textstyle\sum_t \sum_k \sum_i \left(a_{ki}^t x_{ki}^t + c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+\right) \\
& + \textstyle\sum_t \sum_i \left(b_i^t y_i^t + c_i[y_i^t - y_i^{t-1}]^+\right) \\
& + \textstyle\sum_t \sum_m \sum_i \left(d_{mi}^t \left(u_{mi}^t + 2y_i^t\right)\right) + \sum_t \sum_i \sum_j d_{ij}^t v_{ij}^t \\
& + \textstyle\sum_t \sum_m \mathbb{I}\Big\{sign\Big[q_m^t \cdot \big(\sum_k \sum_i z_{kim}^t sign(f^k(p_m^t)) \\
& + \textstyle\sum_i w_{im}^t sign(f_m^t(p_m^t))\big)\Big] < 0\Big\} \qquad (1) \\
\text{s.t.} \quad & z_{kim}^t \le y_i^t, \forall k, i, t, m, \qquad (1a) \\
& w_{im}^t \le y_i^t, \forall i, t, m, \qquad (1b) \\
& \textstyle\sum_i y_i^t = 1, \forall t, \qquad (1c) \\
& \textstyle\sum_i x_{ki}^t = 1, \forall t, k, \qquad (1d) \\
& \textstyle\sum_i (\sum_k z_{kim}^t + w_{im}^t) = 1, \forall m, t, \qquad (1e) \\
& \textstyle\sum_k x_{ki}^t + y_i^t \le D_i^t, \forall i, t, \qquad (1f) \\
& \textstyle\sum_j v_{ij}^t \ge x_{ki}^t, \forall k, i, t, \qquad (1g) \\
& v_{ij}^t \le y_j^t, \forall i, j, t, \qquad (1h) \\
& u_{mi}^t \ge x_{ki}^t, \forall k, i, m, t, \qquad (1i) \\
& u_{mi}^t \ge y_i^t, \forall i, m, t, \qquad (1j) \\
\text{var.} \quad & x_{ki}^t, y_i^t, u_{mi}^t, v_{ij}^t \in \{0, 1\}, z_{kim}^t, w_{im}^t \in [0, 1].
\end{aligned}$$

Constraints (1a) and (1b) ensure that only the edge that hosts the online classifier can maintain all the weights for all the classifiers. Constraints (1c) and (1d) ensure that the online classifier can only be hosted by a single edge, and every offline classifier can only be hosted by a single edge. Constraint (1e) states that all the weights are normalized and their sum is one. Constraint (1f) respects the capacity of each edge. Constraints (1g) and (1h) guarantee that the decision computed by every offline classifier is transmitted to the edge that hosts the online classifier. Constraints (1i) and (1j) guarantee that every data sample is dispatched to every edge that hosts the classifier(s).

We note that as different terms in the objective may have different units, each term can be associated to a weight which is used to balance and control the optimization according to the service's need and preference. This weighted-sum method is pretty common, and for brevity, we do not show such weights. We will have experimental evaluations on this aspect.

***Challenges:*** It is non-trivial to solve the above optimization problem due to three challenges. First, we want to solve the problem in an *online* manner. That is, as time goes, at any time slot, we want to make control decisions for that time slot while observing only the inputs for that single time slot and no inputs for all the future time slots. For example, for the start-up cost $c_{ki}\left[x_{ki}^t - x_{ki}^{t-1}\right]^+$, we need to make $\boldsymbol{x}^{t-1}$ at $t-1$; however, at $t-1$, we have not made the decision of $\boldsymbol{x}^t$, without which it is difficult to make a good decision of $\boldsymbol{x}^{t-1}$ in order to optimize $c_{ki}\left[x_{ki}^t - x_{ki}^{t-1}\right]^+$. It is a similar case for $\boldsymbol{y}^{t-1}$ and its start-up cost $c_i\left[y_i^t - y_i^{t-1}\right]^+$. Second, the problem contains *nonlinear* terms, i.e., the number of the mistakes of transfer learning with $sign(\cdot)$ and $\mathbb{I}\{\cdot\}$ functions, which are intertwined with online training. While $f^k(p_m^t)$ can be observed as the offline classifiers are given and the data samples arrive, we need to determine how we should train or update the online classifier $f_m^t(\cdot)$ at $t$ in our algorithms in addition to accommodating the nonlinearity. Third, the problem is *NP-hard*. The problem contains integer variables, and is actually NP-hard as it can be reduced to the existing uncapacitated facility location problem (if we only retain the variables $\boldsymbol{x}$ and $\boldsymbol{u}$ and the related terms in the formulation). The NP-hardness demands computationally efficient algorithms. It is not easy to achieve so in the offline setting, and it will be harder to do it in an online setting.

***Goal:*** Our goal is to design polynomial-time approximation algorithms which make control decisions in an online manner and ensure that such decisions lead to a provable "competitive ratio". The competitive ratio $r$ is a constant, which may contain parameters, to satisfy $H_1 \le rH_1^*$. Here, $H_1$ refers to the value of the objective function of the problem (1) evaluated with the solution produced by our online algorithms, and $H_1^*$ refers to that evaluated with the optimal solution of (1) which were to be produced in the offline manner, when all the inputs were observed all at once before the start of the entire time horizon.

***Algorithm Structure:*** First, to overcome intractability, we design a primal-dual approximation algorithm (i.e., Algorithm 1) to solve $\boldsymbol{x}^t$, $\boldsymbol{v}^t$ and $\boldsymbol{u}^t$ from the one-shot problem at any $t$, assuming $\boldsymbol{y}^t$ is given. Second, to overcome the challenge of being online, we design two algorithms (i.e., Algorithm 2, which invokes Algorithm 1, and Algorithm 3, which invokes Algorithm 2) to (re-)solve $\boldsymbol{x}^t$, $\boldsymbol{y}^t$, $\boldsymbol{v}^t$ and $\boldsymbol{u}^t$ at $t$, pursuing the dynamic trade-off between switching to a new decision and continuing to stay at the previous decision. Third, to accommodate nonlinearity and online training, we present the overall algorithm (i.e., Algorithm 4, which invokes Algorithm 3) to set the weights $\boldsymbol{z}^t$ and $\boldsymbol{w}^t$ of all classifiers given $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$ at $t$, and conduct online training by updating $f_m^t(\cdot)$ per data sample $m$. We elaborate these four algorithms and prove the performance guarantees in the next two sections.

## III. Algorithm for One-shot Problem

In this section, we formulate the innermost problem of the offline classifier placement for each individual time slot, assuming all the other control decisions are pre-specified (and these decisions will be all made in the next section). We design a primal-dual algorithm, i.e., Algorithm 1, with a provable and guaranteed approximation ratio for this one-shot problem.

### A. Innermost Problem

Consider $H_1$, i.e., the objective function of (1). If $\boldsymbol{y}^t$ is given, then at $t$, we can temporarily remove $\sum_m \mathbb{I}\{sign[q_m^t \cdot (\sum_k \sum_i z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))] < 0\}$, because as we will show, given $\boldsymbol{y}^t$ we will use Algorithm 4 in Section IV to determine $\boldsymbol{z}^t$ and $\boldsymbol{w}^t$ to satisfy Constraints

(1a)∼(1b). Also, if $\boldsymbol{y}^t$ is given, $\sum_i b_i^t y_i^t + \sum_m \sum_i 2d_{mi}^t y_i^t + \sum_i c_i \left[y_i^t - y_i^{t-1}\right]^+$ is known at $t$ accordingly. Thus, we only need to focus on the following part of the problem (1):

$$\begin{aligned}
\text{Min} \quad H_2 =& \textstyle\sum_{t,k,i} \left(a_{ki}^t x_{ki}^t + c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+\right) \\
&+ \textstyle\sum_{t,m,i} d_{mi}^t u_{mi}^t + \sum_{t,i} d_i^t v_i^t && (2)\\
\text{s.t.} \quad & \textstyle\sum_i x_{ki}^t = 1, \forall k, t, && (2a)\\
& \textstyle\sum_k x_{ki}^t \le Q_i^t, \forall i, t, && (2b)\\
& v_i^t \ge x_{ki}^t, \forall k, i, t, && (2c)\\
& u_{mi}^t \ge x_{ki}^t, \forall i, m, k, t, && (2d)\\
\text{var.} \quad & x_{ki}^t, v_i^t, u_{mi}^t \in \{0, 1\}, && (2g)
\end{aligned}$$

where $Q_i^t = D_i^t - y_i^t$. As $\boldsymbol{y}^t$ is given, we have replaced $v_{ij}^t$ by $v_i^t$, $d_{ij}^t$ by $d_i^t$, and $\sum_j v_{ij}^t \ge x_{ki}^t$ by $v_i^t \ge x_{ki}^t$ for simplification. This is because there is only one $j$ where $y_j^t = 1$, and for all the other $j$s, we have $y_j^t = 0$. So, for this specific $j$, we can set $v_{ij}^t = 1$, $\forall i$; for all the other $j$s, we naturally have $v_{ij}^t = 0$, $\forall i$, due to (1h). $v_{ij}^t$ is irrelevant to $j$ now, but corresponds to $v_i^t$ in a one-to-one manner; it is a similar case for $d_{ij}^t$ and $d_i^t$.

To tackle the time-coupled term $\sum_{t,k,i} c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+$ in $H_2$ in an online manner, we will explore the real-time trade-off between keeping the "previous" decisions and applying the "new" decisions at each $t$, which will be discussed later in details. Now, in order to obtain such "new" decisions, we temporarily remove $\sum_{k,i} c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+$ in $H_2$ to decouple the temporal dependencies introduced by the nonlinear term involving previous configurations. This removal simplifies the problem into a per-slot optimization framework, enabling real-time computation by eliminating the need to track historical configurations. By omitting the time index $t$, we further reduce the problem to a formulation that depends only on the current system state. Although this simplification temporarily ignores explicit cross-time penalties, it ensures computational tractability for latency-critical edge systems; we will later reintroduce temporal coordination through adaptive parameter tuning in Section IV to balance stability and optimality. We construct the following one-shot problem at any individual $t$ (where we have omitted the time index $t$ to simplify the presentation). This is also what we call our "innermost problem":

$$\begin{aligned}
\text{Min} \quad H_3 =& \textstyle\sum_{k,i} a_{ki} x_{ki} + \sum_{m,i} d_{mi} u_{mi} + \sum_i d_i v_i && (3)\\
\text{s.t.} \quad & \textstyle\sum_i x_{ki} = 1, \forall k, && (3a)\\
& \textstyle\sum_k x_{ki} \le Q_i, \forall i, && (3b)\\
& v_i \ge x_{ki}, \forall k, i, && (3c)\\
& u_{mi} \ge x_{ki}, \forall i, m, k && (3d)\\
\text{var.} \quad & x_{ki}, v_i, u_{mi} \in \{0, 1\}. && (3e)
\end{aligned}$$

By relaxing the binary variables $x_{ki}, v_i, u_{mi}$ into real domains and introducing dual variables $\lambda_k, \delta_i, \epsilon_{ki}, \phi_{kim}$ for (3a)∼(3d), respectively, we write the Lagrange dual problem as

$$\begin{aligned}
\text{Max} \quad H_4 =& -\textstyle\sum_i Q_i \delta_i - \sum_k \lambda_k && (4)\\
\text{s.t.} \quad & a_{ki} + \delta_i + \lambda_k + \epsilon_{ki} + \textstyle\sum_m \phi_{kim} \ge 0, \forall i, k && (4a)\\
& \textstyle\sum_k \epsilon_{ki} \le d_i, \forall i, && (4b)\\
& \textstyle\sum_k \phi_{kim} \le d_{mi}, \forall m, i, && (4c)\\
\text{var.} \quad & \delta_i, \epsilon_{ki}, \phi_{kim} \ge 0, \lambda_k \in R. && (4d)
\end{aligned}$$

**Algorithm 1:** Offline Classifier One-Shot Placement

**Input:** $a_{ki}$, $d_{mi}$, $d_i$, $y_i$, $Q_i = D_i - y_i$

1 **Initialize:** $\delta_i, \gamma_k, \epsilon_{ki}, \phi_{kim}, \Delta Q_i = 0$

2 **for** $k \in [K]$ **do**

3 $\quad i^+ = argmin_{i\in[I]}(a_{ki} + \delta_i + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$;

4 $\quad$ **while** $\Delta Q_{i^+} + 1 > Q_{i^+}$ **do**

5 $\quad\quad [I] = [I] \backslash i^+$;

6 $\quad\quad i^+ = argmin_{i\in[I]}(a_{ki} + \delta_i + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$;

7 $\quad i^* = i^+, \Delta Q_{i^*} = \Delta Q_{i^*} + 1$;

8 $\quad \delta_{i^*} = \delta_{i^*}(1 + \frac{1}{Q_i}) + \frac{a_{ki^*} + d_{i^*}/K + \sum_m d_{mi^*}/K}{Q_i \xi}$;

9 $\quad \lambda_k = -(\delta_{i^*} + a_{ki^*} + \frac{d_{i^*}}{K} + \frac{\sum_m d_{mi^*}}{K})$;

10 $\quad x_{ki^*} = 1, u_{mi^*} = 1$;

11 $\quad v_{i^*} = 1$ (i.e., $v_{i*j} = 1$ where $y_j = 1$);

**Output:** $\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{u}$

### *B. Primal-Dual Algorithm*

We design Algorithm 1 to simultaneously construct integral feasible solutions to the primal problem (3) and feasible solutions to the dual problem (4). The idea of the primal-dual algorithm is to elevate the dual variable continuously until the dual constraint becomes tight (i.e., a constraint of the form of $ax \le b$ is considered tight when $ax = b$), and then the primal variable corresponding to that tight dual constraint can be set to a non-zero value in order to still satisfy the complementary slackness of the Karush-Kuhn-Tucker (KKT) conditions. Our Algorithm 1 is for each $t$, so $t$ is omitted from the presentation.

We explain our Algorithm 1 following the above principle. By combing (4b) and (4c) with (4a), we can transform (4a) into $\sum_k(\lambda_k + \delta_i + a_{ki}) + d_i + \sum_m d_{mi} \ge 0$. Note that if this inequality is tight, then (4a) is tight. One sufficient condition to make this inequality hold is to make the following hold: $\lambda_k + \delta_i + a_{ki} + d_i/K + \sum_m d_{mi}/K \ge 0$, $\forall k$. Now, instead of increasing dual variables slowly, we can directly set the value of $\lambda_k$ to $\lambda_k = -\min_{i\in[I]}(\delta_i + a_{ki} + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$, which can make (4a) tight. Afterwards, the primal variable $x_{ki}$, which corresponds to (4a), can be set to 1; $v_i$ and $u_{mi}$ can be also set to 1 at this point. Based on what is stated above, we choose $i$ as $i = argmin_{i\in[I]}(a_{ki} + \delta_i + \frac{d_i}{K} + \frac{\sum_m d_{mi}}{K})$ for each $k$, as in Line 3 of our algorithm. Line 4 guarantees no violation of the constraint (3b), where $\Delta Q_i$ is defined as the number of classifiers that edge $i$ host. We regard the dual variable $\delta_i$ as a reflection on the potential capability of edge $i$ to host offline classifiers (i.e., the larger $\delta_i$, the less likely $i$ is to be selected). Thus, $\delta_i$ is increased for the selected edge $i^*$ due to the decrease of $\Delta Q_{i^*}$ and remains intact otherwise. The update of $\delta_i$ is carefully designed for achieving low additive loss in approximation ratio, as in Line 8, where $\xi = \max_{i\in[I]}\{Q_i\}$. Lines 9 and 10 update the dual variable $\lambda_k$ and the primal variables $x_{ki}, v_i, u_{mi}$, respectively.

### *C. Performance Analysis*

First, by Lemma 1, we demonstrate that Algorithm 1 is a polynomial-time algorithm with no violation of the constraints

**TABLE II:** Abbreviative Notations

| Notation | Definition |
|---|---|
| $\mathbb{X}^t_{SC}$ | $\sum_{k,i} c_{ki}[x^t_{ki} - x^{t-1}_{ki}]^+$ |
| $\mathbb{X}^t_{-SC}$ | $\sum_{k,i} a^t_{ki} x^t_{ki} + \sum_{i,j} d^t_{ij} v^t_{ij} + \sum_{m,i} d^t_{mi} u^t_{mi}$ |
| $\Delta\mathbb{X}_{-SC}$ | $\sum_{\tau=\hat{t}}^{t-1} \mathbb{X}^\tau_{-SC}$ |
| $\mathbb{Y}^t_{SC}$ | $\sum_i c_i[y^t_i - y^{t-1}_i]^+$ |
| $\mathbb{Y}^t_{-SC}$ | $\mathbb{X}^t_{SC} + \mathbb{X}^t_{-SC} + \sum_i b^t_i y^t_i + \sum_{m,i} 2d^t_{mi} y^t_i$ |
| $\Delta\mathbb{Y}_{-SC}$ | $\sum_{\tau=\check{t}}^{t-1} \mathbb{Y}^\tau_{-SC}$ |

of the primal problem (3) and the dual problem (4). Second, by Theorem 1 on top of Lemma 1, we derive the approximation ratio $r_1$. We show $H_3 \leq r_1 H_4 \leq r_1 H_3^*$, where $H_3$ and $H_4$ refer to the objective function values of (3) and (4) evaluated with the feasible solutions returned by Algorithm 1; $H_3^*$ refers to the optimal objective function value of the primal problem. Note that $H_4 \leq H_3^*$ holds automatically due to duality.

**Lemma 1.** *Algorithm 1 returns feasible solutions to both the problem (3) and the problem (4) in the polynomial time.*

*Proof.* A solution is feasible for a problem if the solution satisfies the problem's constraints. For (3), (3a) is satisfied by Line 10. Lines 4∼5 ensure no violation of the edge capacity limit, i.e., (3b). Once the edges to host offline classifiers are determined, (3c) and (3d) are also satisfied, following Line 10. Line 10 also guarantees (3e). For (4), this inequality $\sum_k(\lambda_k + \delta_i + a_{ik}) + d_i + \sum_m d_{mi} \geq 0$ is constructed based on (4a), (4b) and (4c), guaranteeing they are satisfied according to Lines 3∼6. As for the time complexity of Algorithm 1, the *for* loop runs $K$ times, and the *while* loop in the *for* loop runs at most $I$ times according to its termination condition $\Delta Q_{i^+} + 1 > Q_{i^+}$. Thus, the total time complexity is $O(KI)$. □

**Theorem 1.** *Algorithm 1 is an $r_1$-approximation algorithm to the problem (3), i.e., $H_3 \leq r_1 H_3^*$, where $r_1 = \frac{\xi}{\xi - 1}$ and $\xi > 1$ is a tunable parameter. The parameter $\xi$ balances update responsiveness and approximation quality: smaller $\xi$ accelerates dual updates but worsens the competitive ratio, while larger $\xi$ ensures better performance with slower updates.*

*Proof.* See Appendix A. □

Note that all the appendices are placed in the supplementary material of this paper.

## IV. Algorithms for Long-Term Learning

In this section, we design Algorithms 2 and 3 that determine in real time the offline and online classifier placement with data dispatching and inference aggregation for each time slot. We also design Algorithm 4 for transfer learning upon each data sample as the classifier placement is determined dynamically. We theoretically analyze the number of mistakes of transfer learning and the competitive ratio for the total cost. We use some new notations in Table II to ease our presentation.

### A. Online Algorithms for Classifiers Placement

Our main rationale is to postpone changing the placement of the classifiers until "appropriate". That is, until the cumulative non-start-up cost (i.e., operational cost and delay of transfer learning incurred by continuing to host classifiers at previous locations) exceeds the current start-up cost (i.e., downloading cost and edge instantiation cost incurred by changing the classifier placement) times a constant which can be controlled.

**Algorithm 2:** Conditional Offline Classifier Placement

**Input:** $\boldsymbol{y}^t, \Delta\mathbb{X}_{-SC}, \hat{t}$

1 given $\boldsymbol{y}^t$, get $\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t}$ by invoking **Algorithm 1**;
2 **if** $\mathbb{X}^t_{SC}(\widetilde{\boldsymbol{x}^t}, \boldsymbol{x}^{\hat{t}}) \leq \frac{1}{\rho_2}\Delta\mathbb{X}_{-SC}$ **then**
3  $\boldsymbol{x}^t = \widetilde{\boldsymbol{x}^t}$;
4  $\Delta\mathbb{X}_{-SC} = \mathbb{X}^t_{-SC}(\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t})$ ;
5  $\hat{t} = t$;
6 **else**
7  $\boldsymbol{x}^t = \boldsymbol{x}^{\hat{t}}$;
8  set $\boldsymbol{u}^t$ and $\boldsymbol{v}^t$ according to $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$;
9  $\Delta\mathbb{X}_{-SC} = \Delta\mathbb{X}_{-SC} + \mathbb{X}^t_{-SC}(\boldsymbol{x}^{\hat{t}}, \boldsymbol{u}^t, \boldsymbol{v}^t)$;

**Output:** $\boldsymbol{x}^t$, $\boldsymbol{v}^t$, $\boldsymbol{u}^t$, $\hat{t}$

**Algorithm 3:** Offline and Online Classifier Placement

**Input:** $\Delta\mathbb{Y}_{-SC}, \hat{t}, \check{t}$

**for** $i \in [I]$ **do**
 set $\widetilde{\boldsymbol{y^t}}$ as $\widetilde{y}^t_i = 1$, and $\widetilde{y}^t_j = 0$ for $j \neq i$;
 given $\widetilde{\boldsymbol{y^t}}$, get $\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t}$ by invoking **Algorithm 2**;
 **if** $\mathbb{Y}^t_{SC}(\widetilde{\boldsymbol{y}^t}, \boldsymbol{y}^{\check{t}}) \leq \frac{1}{\rho_1}\Delta\mathbb{Y}_{-SC}$ **then**
  $\boldsymbol{y}^t = \widetilde{\boldsymbol{y}^t}$;
  $\Delta\mathbb{Y}_{-SC} = \mathbb{Y}^t_{-SC}(\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{y}^t}, \widetilde{\boldsymbol{u}^t}, \widetilde{\boldsymbol{v}^t})$;
  $\check{t} = t$;
 **else**
  $\boldsymbol{y}^t = \boldsymbol{y}^{\check{t}}$;
  given $\boldsymbol{y^t}$, get $\widetilde{\boldsymbol{x}^t}, \widetilde{\boldsymbol{v}^t}, \widetilde{\boldsymbol{u}^t}$ by invoking **Algorithm 2**;
  $\Delta\mathbb{Y}_{-SC} = \Delta\mathbb{Y}_{-SC} + \mathbb{Y}^t_{-SC}(\widetilde{\boldsymbol{x}^t}, \boldsymbol{y}^t, \widetilde{\boldsymbol{u}^t}, \widetilde{\boldsymbol{v}^t})$;
 $H^t_{-M} = \mathbb{Y}^t_{-SC} + \mathbb{Y}^t_{SC}$;
find the minimum $H^t_{-M}$ for all $i$ and its $\boldsymbol{x}^t, \boldsymbol{y}^t, \boldsymbol{u}^t, \boldsymbol{v}^t$;

**Output:** $\boldsymbol{x}^t$, $\boldsymbol{y}^t$, $\boldsymbol{v}^t$, $\boldsymbol{u}^t$, $\hat{t}$, $\check{t}$

We briefly explain our Algorithm 2.[1] In Line 2, $\rho_2$ is the controllable constant as aforementioned, and $\Delta\mathbb{X}_{-SC}$ records the cumulative non-start-up cost from $\hat{t}$ to $t-1$, where $\hat{t}$ refers to the last time slot when the offline classifier placement changes before $t$. When the condition in Line 2 is satisfied, we adopt the decision returned by Algorithm 1; otherwise, we use the most recent decision as the current decision. We design Algorithm 3 in a similar spirit as for Algorithm 2. Specifically, we traverse $I$ possible values of $\boldsymbol{y}^t$ in Lines 1∼2. Then, as in Line 4, only when $\Delta\mathbb{Y}_{-SC}$ exceeds $\mathbb{Y}^t_{SC}(\widetilde{\boldsymbol{y}^t}, \boldsymbol{y}^{\check{t}})$ times $\rho_1$ will we adopt the new decision of $\boldsymbol{y}^t$, where $\check{t}$ indicates the last time slot of the online classifier placement change before $t$. Otherwise, in Line 10, we invoke Algorithm 2 given $\boldsymbol{y}^{\check{t}}$. We record all $H^t_{-M}$ with the different $\boldsymbol{y}^t$, and find the minimum $H^t_{-M}$ over all $i$ with its corresponding decisions.

[1] In Algorithm 2, we use symbols like $\widetilde{x}$ to refer to decisions obtained from Algorithm 1. Analogously, in Algorithm 3, we use symbols like $\widetilde{x}$ to represent decisions obtained from Algorithm 2. This should be clear from the context.

### B. Online Algorithm for Transfer Learning

We propose our overall online transfer learning algorithm, i.e., Algorithm 4, to tie together every per-slot optimization of classifier placement and conduct the actual transfer learning process as data dynamically arrive. Our algorithm conducts online training in four steps: weights update, label inference, parameters update, and online classifier update. First, at each time slot, we invoke Algorithm 3 to find all the classifiers' placements in Line 4, and for the current data sample, we determine the weight for each classifier in Lines 7∼8. Then, for this data sample, we conduct the joint inference as a weighted sum of the static offline classifiers' results and the online classifier's result in Line 10. As receiving the ground-truth in Line 11, we next decrease the weights of those classifiers which misclassify instances so as to weaken their impact by updating the parameters used to determine the weights for the next data sample, as in Lines 13∼17. Finally, we update the online classifier itself based on its loss on the current data sample, as in Lines 19∼21. Here, we regard the data sample's feature $p_m^t$ as a support vector and add it into the set of the support vectors of the online classifier: $f_{m+1}^t = f_m^t + \alpha_m^t q_m^t f_m^t(p_m^t, \cdot)$, where $\alpha_m^t$ is the coefficient for the support vector; $\mathbb{K}(\cdot,\cdot)$ is the kernel function; and $C$ is a constant trade-off value used to prevent the coefficient of the vector from being too large. Note that we allow offline classifiers of arbitrary and heterogeneous types or formats, but without of loss of generality, we focus on training the online classifier as a Support Vector Machine in this paper. This is for concretizing our Algorithm 4, and does not impact our performance analysis and proofs.

**Algorithm 4:** Homogeneous Online Transfer Learning

**Input:** offline classifiers $f^{[K]} = (f^1, f^2, ..., f^K)$, trade-off $C$, and weight discount $\theta \in (0,1)$

1 **Initialize:** $t = 1$, $\hat{t} = \check{t} = 0$, $f^0 = \emptyset$, $\zeta_{k1}^0 = \psi_0^1 = \frac{1}{K+1}$
2 **for** $t = 1, 2, ..., T$ **do**
3  $f_0^t = f^{t-1}$;
4  invoke **Algorithm 3** to obtain $\boldsymbol{x}^t, \boldsymbol{y}^t$;
5  **for** $m = 0, 1, ..., M^t$ **do**
6  ▷ **Weights update**
7  $z_{kmi}^t = \begin{cases} \zeta_{km}^t / (\sum_k \zeta_{km}^t + \psi_m^t), & y_i^t = 1 \\ 0, & y_i^t = 0 \end{cases}$
8  $w_{mi}^t = \begin{cases} \psi_m^t / (\sum_k \zeta_{km}^t + \psi_m^t), & y_i^t = 1 \\ 0, & y_i^t = 0 \end{cases}$
9  ▷ **Label inference**
10  calculate inference: $\hat{q}_m^t = sign(\sum_{k,i} z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))$;
11  receive ground-truth: $q_m^t \in \{-1, +1\}$;
12  ▷ **Parameters update**
13  **for** $k = 1, 2, ..., K$ **do**
14  $\eta_{km}^t = \mathbb{I}\{sign[q_m^t \cdot sign(f^k(p_m^t))] < 0\}$;
15  $\zeta_{k,m+1}^t = \zeta_{k,m}^t \theta^{\eta_{km}^t}$;
16  $\gamma_m^t = \mathbb{I}\{sign[q_m^t \cdot sign(f_m^t(p_m^t))] < 0\}$;
17  $\psi_{m+1}^t = \psi_m^t \theta^{\gamma_m^t}$;
18  ▷ **Online classifier update**
19  calculate loss: $l^t = [1 - q_m^t f_m^t(p_m^t)]^+$;
20  **if** $l^t > 0$ **then**
21  $f_{m+1}^t = f_m^t + \alpha_m^t q_m^t \mathbb{K}(p_m^t, \cdot)$, where $\alpha_m^t = \min\{C, \frac{l^t}{\mathbb{K}(p_m^t, p_m^t)}\}$;

### C. Performance Analysis

We now introduce some new notations to simplify our descriptions. We split $H_1$, the objective function of (1), as $H_1 = \sum_t (H_M^t + H_{-M}^t)$, where $H_M^t = \sum_m \mathbb{I}\{sign[q_m^t \cdot (\sum_k \sum_i z_{kim}^t sign(f^k(p_m^t)) + \sum_i w_{im}^t sign(f_m^t(p_m^t)))] < 0\}$, and $H_{-M}^t = \sum_{k,i}(a_{ki}^t x_{ki}^t + c_{ki}[x_{ki}^t - x_{ki}^{t-1}]^+) + \sum_i (b_i^t y_i^t + c_i[y_i^t - y_i^{t-1}]^+) + \sum_m \sum_i (d_{mi}^t (u_{mi}^t + 2y_i^t)) + \sum_i \sum_j d_{ij}^t v_{ij}^t$. We also use $H_M^{t*}$ and $H_{-M}^{t*}$ to denote their optimal values. First, by Theorem 2, we exhibit that the total number of mistakes, i.e., $\sum_t H_M^t$, incurred by our transfer learning over time is no greater than a constant times the total number of mistakes incurred by the single best classifier (out of the offline classifiers and the online classifier), plus another constant. Second, by Theorem 3, we exhibit the competitive ratio of Algorithm 4. That is, we show $H_1 \le r H_1^*$ and find $r$, where $H_1$ is the objective function value of the problem (1), evaluated with the solutions produced by Algorithm 4, and $H_1^*$ is the optimal objective value. To do so, we derive $\sum_t H_M^t \le r_2 \sum_t H_M^{t*}$ and $\sum_t H_{-M}^t \le r_3 \sum_t H_{-M}^{t*}$, and then find $r$ by $H_1 \le r_2 \sum_t H_M^{t*} + r_3 \sum_t H_{-M}^{t*} \le \max\{r_2, r_3\} H_1^* = r H_1^*$.

**Theorem 2.** *Algorithm 4 incurs the total number of mistakes as* $\sum_t H_M^t \le \big(2 + 2\sqrt{2\ln(K+1)}\big)\Gamma_{min} + 2\ln(K+1)$, *where* $\Gamma_{min} = \min\{\Gamma^1, ..., \Gamma^k, ..., \Gamma^K, \Gamma^O\}$, $\Gamma^k = \sum_{t,m} \eta_{km}^t$, $\Gamma^O = \sum_{t,m} \gamma_m^t$. *Here, $\eta_{km}^t$ and $\gamma_m^t$ indicate whether the inference computed by the offline classifier $k$ and the online classifier is wrong for the data sample $m$ of the time slot $t$, respectively, as in Algorithm 4.*

*Proof.* See Appendix B. □

Theorem 2 reveals that the cumulative number of inference mistakes incurred by the proposed algorithm is tightly bounded by the performance of the best offline classifier, up to a sublinear regret term. This ensures that the online classifier learns effectively over time and minimizes errors adaptively.

**Theorem 3.** *Algorithm 4 is an $r$-competitive online algorithm to the problem (1), i.e., $H_1 \le r H_1^*$, where $r = \max\{r_2, r_3\}$, $r_2 = \frac{\ln(1/\theta) + \ln(K+1)}{1-\theta}$, and $r_3 = (1 + \frac{1}{\rho_1})(1 + \frac{1}{\rho_2} + D_{max}) r_1 \sigma$. Here, $\theta$ is a constant in Algorithm 4; $\rho_1$ and $\rho_2$ are constants in Algorithms 2 and 3; $r_1$ is the approximation ratio of Algorithm 1, as in Theorem 1; $D_{max} = \max\{\max_{i,k,t}\{\frac{b_i^t}{a_{k,i}^t K}\}, 2\}$; and $\sigma = \max_t\{\frac{\max_i\{\sum_k a_{ki} + \sum_j d_{ij} + \sum_m d_{mi}\}}{\min_i \sum_k a_{ki}}\}$.*

*Proof.* See Appendix C. □

Theorem 3 shows that our algorithm achieves an $r$-competitive ratio with respect to the offline optimum, where $r$ is tightly linked to system heterogeneity and algorithm parameters. The structure of this ratio indicates that the

performance gap is bounded and manageable under practical configurations.

## V. Heterogeneous Online Transfer Learning

While the previous sections assume that the offline classifiers and the online classifier have the same or homogeneous feature space, in this section, we consider the setting that they have different or heterogeneous feature spaces. We focus on the case where the feature space of each offline classifier is a "subset" of that of the online classifier [32]. For example, for three offline classifiers and one online classifier, if each data sample for the online classifier has the features in the form of $\{p_{m1}, p_{m2}, p_{m3}\}$, then each data sample for the three offline classifiers could have the feature in the form of $\{p_{m1}, p_{m2}\}$, $\{p_{m2}, p_{m3}\}$, and $\{p_{m3}\}$, respectively. The feature space of every offline classifier can be arbitrary but may not have the full dimensions of the feature space of the online classifier.

In this section, we model and formulate the total cost minimization problem of the distributed heterogeneous online transfer learning in the cloud-edge networks. We also design a set of new algorithms to address this case with a provable competitive ratio for the total cost.

We summarize our additional notations in Table III.

**TABLE III:** Additional Notations

| Inputs | Descriptions |
|---|---|
| $A^t_{\iota ki}$ | Operational cost of hosting the offline or online (depending on $\iota$) classifier $k$ on the edge $i$ at $t$ |
| $B^t_i$ | Operational cost of inference aggregation on the edge $i$ at $t$ |
| $C_{\iota ki}$ | Start-up cost of downloading offline classifier $k$ from the cloud to edge $i$, or preparing the VMs or containers for the online classifier $k$ on the edge $i$ (depending on $\iota$) |
| $C_i$ | Start-up cost of inference aggregation on the edge $i$ |
| $f^t_{\iota km}(\cdot)$ | Decision function of the offline or online classifier $k$ at the time slot $t$ for the data sample $m$ |
| $\mathcal{F}^t_{kim}(\cdot)$ | Strong classifier which combines the decision functions $f^t_{\iota km}(\cdot)$ at the time slot $t$ for the data sample $m$ on the edge $i$ |
| **Decisions** | **Descriptions** |
| $X^t_{\iota ki}$ | Whether or not the offline or online (depending on $\iota$) classifier $k$ is downloaded from the cloud and hosted on the edge $i$ at the time slot $t$ |
| $Y^t_i$ | Whether or not the edge $i$ at the time slot $t$ is selected for inference aggregation |
| $P^t_{kim}$ | Weight for the strong classifier $\mathcal{F}^t_{kim}(\cdot)$ on the edge $i$ for the data sample $m$ at the time slot $t$ |
| $W^t_{\iota kim}$ | Weight for the offline or online (depending on $\iota$) classifier $k$ on the edge $i$ for the data sample $m$ at the time slot $t$ |
| $U^t_{mi}$ | Whether or not to transfer the ground-truth label of $m$ from the edge where it arrives to the edge $i$ that hosts the online classifier(s) at $t$ |
| $V^t_{ij}$ | Whether or not to transfer the decision results of classifiers from the edge $i$ to the edge $j$ at $t$ |

### A. System Models

***Distributed Heterogeneous Online Transfer Learning:*** For the data sample $m$ of the time slot $t$, $f^t_{0km}(\cdot)$ is the decision function of the offline classifier $k$ with the feature space $\mathbb{R}^k$, where $f^t_{0km}(\cdot) = f^{t+1}_{0k0}(\cdot)$, $\forall t, k, m$; and $f^t_{1km}(\cdot)$ and $f^t_{2km}(\cdot)$ are the decision functions of the online classifiers with the feature spaces $\mathbb{R}^k$ and $\mathbb{R}^o/\mathbb{R}^k$, respectively, where $\mathbb{R}^o$ is the feature space of the online classifier. Unlike homogeneous transfer learning, here we have two decision functions for the online classifiers corresponding to each offline classifier. That is, in this setting, the total number of the online classifiers in the system is twice as many as the offline classifiers. We thus allow that each online classifier can be on a different edge, and that the weights for all the offline and online classifiers still need to be hosted collectively on one common edge for conducting inference aggregation, no matter this edge hosts offline classifiers, online classifiers, or no classifier.

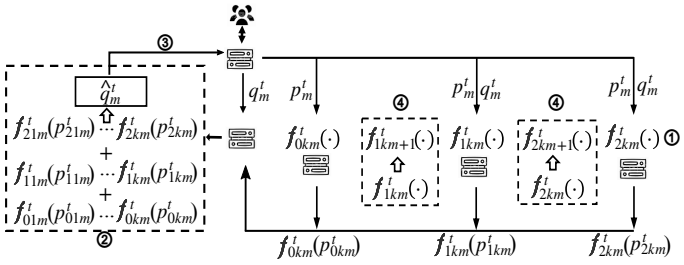



**Fig. 3:** Heterogeneous Transfer Learning per Data Sample

At the time slot $t$, as the data sample $m$ arrives at the system, we design the heterogeneous transfer learning process that works as follows, also shown in Fig. 3:

- *Step 1:* The data sample $m$ with its feature value $p^t_m$ is dispatched to every edge that has the offline classifiers or the online classifiers. Receiving $p^t_m$, every offline classifier $k$ splits the feature $p^t_m$ into $p^t_{1km} \in \mathbb{R}^k$ and $p^t_{2km} \in \mathbb{R}^o/\mathbb{R}^k$, and each offline classifier $k$ computes $f^t_{0km}(p^t_{0km})$, where we denote $p^t_{0km} = p^t_{1km}$, $\forall t, k, m$; and the online classifiers also split the feature and compute $f^t_{1km}(p^t_{1km})$ and $f^t_{2km}(p^t_{2km})$.
- *Step 2:* The decisions from the offline classifiers and online classifiers are sent to the single edge that is selected for the weights update and the inference aggregation to compute the inferred label as $\hat{q}^t_m = sign\big[\sum_k \sum_i P^t_{kim} \mathcal{F}^t_{kim}(p^t_m)\big]$, where $\mathcal{F}^t_{kim}(p^t_m) = sign\big\{\sum_\iota W^t_{\iota kim} sign[f^t_{\iota km}(p^t_{\iota km})]\big\}$, $W^t_{\iota kim}$ is the weight for the classifier $f^t_{\iota km}(\cdot)$ on the edge $i$, where $\iota = 0, 1, 2$; and $\mathcal{F}^t_{kim}(p^t_m)$ serves as a "strong" classifier which combines the decision functions $f^t_{\iota km}(\cdot)$; and $P^t_{kim}$ is the weight for this strong classifier.
- *Step 3:* The inferred label $\hat{q}^t_m$ is then sent to the edge where the data sample $m$ arrives originally, and is further sent back to the user.
- *Step 4:* The ground-truth label $q^t_m$ arrives at that same edge, and is dispatched to the selected edge as in *Step 2*, where the weight for each classifier is updated. Note that this ground-truth label $q^t_m$ is also dispatched to every edge that hosts any online classifier(s), where the decision function(s) of such online classifier(s) are updated, i.e., $f^t_{1km}(\cdot)$ and $f^t_{2km}(\cdot)$ are updated to $f^t_{1km+1}(\cdot)$ and $f^t_{2km+1}(\cdot)$, $\forall k$.

***Control Decisions:*** We focus on the following control decisions. We use $X^t_{\iota ki} \in \{1, 0\}$ to denote whether or not the offline classifier $k$ is hosted on the edge $i$ at the time slot $t$ (when $\iota = 0$), or whether or not the online classifiers that correspond to the offline classifier $k$ are hosted on the edge $i$ at the time slot $t$ (when $\iota = 1, 2$). We use $Y^t_i \in \{1, 0\}$ to

denote whether or not the edge $i$ at the time slot $t$ is chosen for maintaining the weights and conducting the inference aggregation. We use $U_{mi}^t \in \{1,0\}$ to denote whether or not to transfer the ground-truth label of $m$ from the edge where it arrives to the edge $i$ at the time slot $t$ (for updating decision functions). We use $V_{ij}^t$ to denote whether or not to transfer the decision results from the edge $i$ to the edge $j$ at the time slot $t$. We use $P_{kim}^t \in [0,1]$ to denote the weight for the strong classifier, and use $W_{\iota kim}^t \in [0,1]$ to denote the weight for the offline classifier $k$ ($\iota = 0$) and the weights for the online classifiers that correspond to the offline classifier $k$ ($\iota = 1,2$) on the edge $i$ for the data sample $m$ at the time slot $t$. Besides these decision variables, we continue to use $u_{mi}^t \in \{1,0\}$ to denote whether or not to transfer the data sample $m$ from the edge where it arrives to the edge $i$ at $t$, as in previous sections. Note that from now on $U_{mi}^t$ and $u_{mi}^t$ have different meanings.

***Cost of Heterogeneous Online Transfer Learning:*** The cost at any individual time slot $t$ consists of the following components: (1) the operational cost of hosting classifiers and conducting inference aggregation on edges: $\sum_\iota \sum_k \sum_i A_{\iota ki}^t X_{\iota ki}^t + \sum_i B_i^t Y_i^t$; (2) the start-up cost of preparing the VMs or containers on edges for hosting classifiers and for conducting inference aggregation: $\sum_\iota \sum_k \sum_i C_{\iota ki} \left[X_{\iota ki}^t - X_{\iota ki}^{t-1}\right]^+ + \sum_i C_i \left[Y_i^t - Y_i^{t-1}\right]^+$, where $[\cdot]^+ = \max\{\cdot, 0\}$; (3) the performance overhead incurred by running distributed heterogeneous transfer learning across edges, including the delay of dispatching data samples $\sum_m \sum_i d_{mi}^t u_{mi}^t$, the delay of dispatching the ground-truth label to the edges that have online classifiers for updating decision functions $\sum_m \sum_i d_{mi}^t U_{mi}^t$, the delay of transmitting decisions of offline and online classifiers $\sum_i \sum_j d_{ij}^t V_{ij}^t$, and the delay of transmitting the inferred label and the ground-truth label from (or to) the edge that is selected for conducting the inference aggregation $2 \cdot \sum_m \sum_i d_{mi}^t Y_i^t$. All the new coefficients here, such as $A_{\iota ki}^t$, $B_i^t$, $C_{\iota ki}$, and $C_i$, have similar meanings to the corresponding notations as in the homogeneous transfer learning setting.

***Mistakes of Heterogeneous Online Transfer Learning:*** We consider the number of "mistakes", i.e., the number of occurrences where the inferred label does not match the ground-truth label. We denote the number of mistakes for any time slot $t$ as $\mathbb{I}\Big\{sign\Big[q_m^t \cdot \big(\sum_k \sum_i P_{kim}^t \mathcal{F}_{kim}^t(p_m^t)\big)\Big] < 0\Big\}$.

### B. Problem Formulation

We minimize the sum of the long-term total cost and total number of mistakes of heterogeneous online transfer learning:

$$\begin{aligned}
\text{Min} \quad & H_5 = \textstyle\sum_t \sum_\iota \sum_k \sum_i \left(A_{\iota ki}^t X_{\iota ki}^t + C_{\iota ki}[X_{\iota ki}^t - X_{\iota ki}^{t-1}]^+\right) \\
& + \textstyle\sum_t \sum_i \sum_j d_{ij}^t V_{ij}^t + \sum_t \sum_i \left(B_i^t Y_i^t + C_i[Y_i^t - Y_i^{t-1}]^+\right) \\
& + \textstyle\sum_t \sum_m \sum_i d_{mi}^t U_{mi}^t + \sum_t \sum_m \sum_i \left(d_{mi}^t \left(u_{mi}^t + 2Y_i^t\right)\right) \\
& + \textstyle\sum_t \sum_m \mathbb{I}\Big\{sign\Big[q_m^t \cdot \big(\sum_k \sum_i P_{kim}^t \mathcal{F}_{kim}^t(p_m^t)\big)\Big] < 0\Big\} \qquad (5) \\
\text{s.t.} \quad & P_{kim}^t \le Y_i^t, \forall k,i,t,m, \qquad (5a) \\
& W_{\iota kim}^t \le Y_i^t, \forall \iota,k,i,t,m, \qquad (5b) \\
& \textstyle\sum_i Y_i^t = 1, \forall t, \qquad (5c) \\
& \textstyle\sum_i X_{\iota ki}^t = 1, \forall \iota,t,k, \qquad (5d) \\
& \textstyle\sum_k \sum_i P_{kim}^t = 1, \forall m,t, \qquad (5e) \\
& \textstyle\sum_\iota \sum_i W_{\iota kim}^t = 1, \forall k,m,t, \qquad (5f) \\
& \textstyle\sum_\iota \sum_k X_{\iota ki}^t + Y_i^t \le D_i^t, \forall i,t, \qquad (5g) \\
& \textstyle\sum_j V_{ij}^t \ge X_{\iota ki}^t, \forall \iota,k,i,t, \qquad (5h) \\
& V_{ij}^t \le Y_j^t, \forall i,j,t, \qquad (5i) \\
& u_{mi}^t \ge X_{\iota ki}^t, \forall \iota,k,i,m,t, \qquad (5j) \\
& U_{mi}^t \ge X_{\iota ki}^t, \forall \iota \in \{1,2\},k,i,m,t, \qquad (5k) \\
\text{var.} \quad & X_{\iota ki}^t, Y_i^t, u_{mi}^t, U_{mi}^t, V_{ij}^t \in \{0,1\}, \\
& P_{kim}^t, W_{\iota kim}^t \in [0,1].
\end{aligned}$$

Constraints (5a) and (5b) ensure that only the edge that is selected for conducting inference aggregation can maintain all the weights of all the classifiers. Constraint (5c) ensures that only one edge in the system is selected for inference aggregation. Constraint (5d) ensures that every classifier is hosted on one edge. Constraint (5e) states that the weights of the strong classifiers are normalized and their sum is one. Constraint (5f) states that the weights of the decision functions are normalized and their sum is one. Constraints (5g) $\sim$ (5j) are similar to Constraints (1f) $\sim$ (1i). Constraint (5k) guarantees that every ground-truth label is dispatched to every edge that hosts online classifiers for updating the decision functions of these online classifiers.

### C. Algorithm Design

We design Algorithms 5$\sim$8 to solve this problem, which are related to but different from Algorithms 1$\sim$4: (i) the placement algorithms change *from* firstly placing the online classifier and then placing the offline classifiers *to* firstly deciding the inference aggregation and then jointly placing both online and offline classifiers; (ii) the online transfer learning algorithm changes *from* homogeneous *to* heterogeneous, working with the split feature spaces of the data samples that arrive online.

***System Control:*** We describe Algorithms 5$\sim$7 first. Consider $H_5$, i.e., the objective function of (5). We now assume that $\boldsymbol{Y}^t$ is given, and remove the mistakes of the heterogeneous transfer learning. Unlike the homogeneous transfer learning, the model of the heterogeneous transfer learning contains an unconventional Constraint (5k). To overcome this, we introduce $\widehat{A}_{\imath\jmath i}^t$, $\widehat{X}_{\imath\jmath i}^t$, $\widehat{C}_{\imath\jmath i}$, and $\mathscr{d}_{\imath mi}^t \widehat{U}_{\imath mi}^t$ as replacements for $A_{\iota ki}^t$, $X_{\iota ki}^t$, $C_{\iota ki}$, and $d_{mi}^t U_{mi}^t$, respectively, where $\imath \in [0,1]$ and $\jmath \in [0,...,K_\imath]$. If $\imath = 0$, it represents the offline classifiers, and $K_0 = K$. If $\imath = 1$, it represents the online classifiers, and $K_1 = 2K$. With this approach, we no longer classify online classifiers into two categories, but instead focus on online or offline classifiers. Note that if $\jmath > K$, then $\widehat{A}_{1ji}^t = A_{2\jmath - Ki}^t$, $\widehat{X}_{1ji}^t = X_{2\jmath-Ki}^t$, $\widehat{C}_{1ji} = C_{2\jmath-Ki}$; if $\jmath \le K$, then $\widehat{A}_{\imath\jmath i}^t = A_{\imath\jmath i}^t$, $\widehat{X}_{\imath\jmath i}^t = X_{\imath\jmath i}^t$, $\widehat{C}_{\imath\jmath i} = C_{\imath\jmath i}$, $\forall \iota,\jmath,i$. We set $\mathscr{d}_{0mi}^t = 0$, $\mathscr{d}_{1mi}^t = d_{mi}$, $\forall m,i$ to ensure that the value of $\widehat{U}_{0mi}^t$ has no effect on our objective function. Given $\boldsymbol{Y}^t$, we obtain the following problem:

$$\text{Min} \quad H_6 = \textstyle\sum_{t,\imath,\jmath,i} \left(\widehat{A}_{\imath\jmath i}^t \widehat{X}_{\imath\jmath i}^t + \widehat{C}_{\imath\jmath i}[\widehat{X}_{\imath\jmath i}^t - \widehat{X}_{\imath\jmath i}^{t-1}]^+\right)$$

$$+\sum_{t,m,i} d_{mi}^t u_{mi}^t + \sum_{t,i} d_i^t V_i^t + \sum_{t,\imath,m,i} \ell_{\imath mi}^t \widehat{U}_{\imath mi}^t \tag{6}$$

$$\text{s.t.} \quad \sum_i \widehat{X}_{\imath\jmath i}^t = 1, \forall \imath, \jmath, t, \tag{6a}$$

$$\sum_{\imath,\jmath} \widehat{X}_{\imath\jmath i}^t \le E_i^t, \forall i, t, \tag{6b}$$

$$V_i^t \ge \widehat{X}_{\imath\jmath i}^t, \forall \imath, \jmath, i, t, \tag{6c}$$

$$u_{mi}^t \ge \widehat{X}_{\imath\jmath i}^t, \forall \imath, \jmath, m, i, t, \tag{6d}$$

$$\widehat{U}_{\imath mi}^t \ge \widehat{X}_{\imath\jmath i}^t, \forall \imath, \jmath, m, i, t, \tag{6f}$$

$$\text{var.} \quad \widehat{X}_{\imath\jmath i}^t, V_i^t, u_{mi}^t, \widehat{U}_{\imath mi}^t \in \{0,1\}, \tag{6g}$$

where $E_i^t = D_i^t - Y_i^t$. Given $\boldsymbol{Y}^t$, we can simplify the expressions in the model by replacing $V_{ij}^t$ with $V_i^t$, $d_{ij}^t$ with $d_i^t$, and $\sum_j V_{ij}^t \ge \widehat{X}_{\imath\jmath i}^t$ with $V_i^t \ge \widehat{X}_{\imath\jmath i}^t$. We can also replace Constraint (5k) with Constraint (6f) since $\ell_{0mi}^t = 0$ and the value of $\widehat{U}_{0mi}^t$ has no impact on the objective function $H_6$.

We obtain the "innermost problem" of heterogeneous online transfer learning:

$$\text{Min} \quad H_7 = \sum_{\imath,\jmath,i} \widehat{A}_{\imath\jmath i} \widehat{X}_{\imath\jmath i} + \sum_{\imath,m,i} \ell_{\imath mi} \widehat{U}_{\imath mi} + \sum_{m,i} d_{mi} u_{mi} + \sum_i d_i V_i \tag{7}$$

$$\text{s.t.} \quad \sum_i \widehat{X}_{\imath\jmath i} = 1, \forall \imath, \jmath, \tag{7a}$$

$$\sum_{\imath,\jmath} \widehat{X}_{\imath\jmath i} \le E_i, \forall i, \tag{7b}$$

$$V_i \ge \widehat{X}_{\imath\jmath i}, \forall \imath, \jmath, i, \tag{7c}$$

$$u_{mi} \ge \widehat{X}_{\imath\jmath i}, \forall \imath, \jmath, m, i, \tag{7d}$$

$$\widehat{U}_{\imath mi} \ge \widehat{X}_{\imath\jmath i}, \forall \imath, \jmath, m, i, \tag{7e}$$

$$\text{var.} \quad \widehat{X}_{\imath\jmath i}, V_i, u_{mi}, \widehat{U}_{\imath mi} \in \{0,1\}. \tag{7f}$$

By relaxing the binary variables $\widehat{X}_{\imath\jmath i}$, $V_i$, $u_{mi}$, and $\widehat{U}_{\imath mi}$ into real domains and introducing dual variables $\kappa_{\imath\jmath}$, $\varsigma_i$, $\varrho_{\imath\jmath i}$, $\upsilon_{\imath\jmath im}$, and $\Phi_{\imath\jmath im}$ for (7a)$\sim$(7e), respectively, we can express the Lagrange dual problem as

$$\text{Max} \quad H_8 = -\sum_i E_i \varsigma_i - \sum_{\imath,\jmath} \kappa_{\imath\jmath} \tag{8}$$

$$\text{s.t.} \quad \widehat{A}_{\imath\jmath i} + \varsigma_i + \kappa_{\imath\jmath} + \varrho_{\imath\jmath i} + \sum_m \upsilon_{\imath\jmath im} + \sum_m \Phi_{\imath\jmath im} \ge 0, \forall \imath, i, \jmath, \tag{8a}$$

$$\sum_{\imath,\jmath} \varrho_{\imath\jmath i} \le d_i, \forall i, \tag{8b}$$

$$\sum_{\imath,\jmath} \upsilon_{\imath\jmath im} \le d_{mi}, \forall m, i, \tag{8c}$$

$$\sum_{\jmath} \Phi_{\imath\jmath im} \le \ell_{\imath mi}, \forall \imath, m, i, \tag{8d}$$

$$\text{var.} \quad \varsigma_i, \varrho_{\imath\jmath i}, \upsilon_{\imath\jmath im}, \Phi_{\imath\jmath im} \ge 0, \kappa_{\imath\jmath} \in R. \tag{8e}$$

We design Algorithm 5 to simultaneously construct integral feasible solutions to the primal problem (7) and feasible solutions to the dual problem (8). In Algorithm 5, Lines 4 $\sim$ 11, we cautiously maintain feasible solutions to both the primal and the dual problems, where $\Delta E_i$ is defined as the number of classifiers that edge $i$ hosts, and $\varpi = \max_{i\in[I]}\{E_i\}$. Lines 12 $\sim$ 20 update the variables $\widehat{U}_{\imath mi}$, $U_{mi}$, $X_{\imath ki}$ and $V_i$.

We then design Algorithms 6 and 7 to determine in real-time the classifier placement with data dispatching and inference aggregation at each time slot. Note that although Algorithms 6 and 7 have a similar structure compared to Algorithms 2 and 3, the underlying idea has changed as described at the beginning of this section. Regarding the notations, we use $\mathcal{X}_{SC}^t$, $\mathcal{X}_{-SC}^t$, $\Delta\mathcal{X}_{-SC}$, $\mathcal{Y}_{SC}^t$, $\mathcal{Y}_{-SC}^t$ and $\Delta\mathcal{Y}_{-SC}$, to replace $\mathbb{X}_{SC}^t$, $\mathbb{X}_{-SC}^t$, $\Delta\mathbb{X}_{-SC}$, $\mathbb{Y}_{SC}^t$, $\mathbb{Y}_{-SC}^t$ and $\Delta\mathbb{Y}_{-SC}$. For the quick reference, we have summarized these new notations in Table IV.

**Algorithm 5:** Classifier One-Shot Placement

**Input:** $A_{\imath ki}$, $d_{mi}$, $d_i$, $Y_i$, $E_i = D_i - Y_i$

1 **Initialize:** $\kappa_{\imath\jmath}, \varsigma_i, \varrho_{\imath\jmath i}, \upsilon_{\imath\jmath im}, \Phi_{\imath\jmath im}, \Delta E_i = 0$

2 **for** $\imath \in \{0,1\}$ **do**

3 **for** $\jmath \in [K_\imath]$ **do**

4 $i^+ = argmin_{i\in[I]}(\widehat{A}_{\imath\jmath i} + \varsigma_i + \frac{d_i}{3K} + \frac{\sum_m d_{mi}}{3K} + \frac{\sum_{\imath,m} \ell_{\imath mi}}{3K})$;

5 **while** $\Delta E_{i^+} + 1 > E_{i^+}$ **do**

6 $[I] = [I] \backslash i^+$;

7 $i^+ = argmin_{i\in[I]}(\widehat{A}_{\imath\jmath i} + \varsigma_i + \frac{d_i}{3K} + \frac{\sum_m d_{mi}}{3K} + \frac{\sum_{\imath,m} \ell_{\imath mi}}{3K})$;

8 $i^* = i^+, \Delta E_{i^*} = \Delta E_{i^*} + 1$;

9 $\varsigma_{i^*} = \varsigma_{i^*}(1 + \frac{1}{E_i}) + \frac{\widehat{A}_{\imath\jmath i^*} + d_{i^*}/3K + \sum_m d_{mi^*}/3K + \sum_{\imath,m} \ell_{\imath mi^*}/3K}{E_i \varpi}$;

10 $\kappa_{\imath\jmath} = -(\varsigma_{i^*} + \widehat{A}_{\imath\jmath i^*} + \frac{d_{i^*}}{3K} + \frac{\sum_m d_{mi^*}}{3K} + \frac{\sum_{\imath,m} \ell_{\imath mi^*}}{3K})$;

11 $\widehat{X}_{\imath\jmath i^*} = 1, u_{mi^*} = 1$;

12 **if** $\imath == 1$ **then**

13 $\widehat{U}_{\imath mi^*} = 1, U_{mi^*} = 1$;

14 **if** $\jmath > K$ **then**

15 $X_{2\jmath - K i^*} = \widehat{X}_{\imath\jmath i^*}$;

16 **else**

17 $X_{\imath\jmath i^*} = \widehat{X}_{\imath\jmath i^*}$;

18 **else**

19 $\widehat{U}_{\imath mi^*} = 1$;

20 $V_{i^*} = 1$ (i.e., $V_{i^* j} = 1$ where $Y_j = 1$);

**Output:** $\boldsymbol{X}, \boldsymbol{U}, \boldsymbol{u}, \boldsymbol{V}$

**TABLE IV:** Abbreviative Notations for Heterogeneous Case

| Notation | Definition |
|---|---|
| $\mathcal{X}_{SC}^t$ | $\sum_{\iota,k,i} C_{\iota ki}[X_{\iota ki}^t - X_{\iota ki}^{t-1}]^+$ |
| $\mathcal{X}_{-SC}^t$ | $\sum_{\iota,k,i} A_{\iota ki}^t X_{\iota ki}^t + \sum_{i,j} d_{ij}^t V_{ij}^t + \sum_{m,i} d_{mi}^t u_{mi}^t + \sum_{m,i} d_{mi}^t U_{mi}^t$ |
| $\Delta\mathcal{X}_{-SC}$ | $\sum_{\tau=\hat{t}}^{t-1} \mathcal{X}_{-SC}^\tau$ |
| $\mathcal{Y}_{SC}^t$ | $\sum_i C_i[Y_i^t - Y_i^{t-1}]^+$ |
| $\mathcal{Y}_{-SC}^t$ | $\mathcal{X}_{SC}^t + \mathcal{X}_{-SC}^t + \sum_i B_i^t Y_i^t + \sum_{m,i} 2d_{mi}^t Y_i^t$ |
| $\Delta\mathcal{Y}_{-SC}$ | $\sum_{\tau=\hat{t}}^{t-1} \mathcal{Y}_{-SC}^\tau$ |

***Learning Control:*** We describe our heterogeneous online transfer learning algorithm, i.e., Algorithm 8. This algorithm has four steps: weights update, label inference, parameters update, and online classifiers update. First, at each time slot, we invoke Algorithm 5 to find all the classifiers' placements in Line 3, and for the current data sample, we split the corresponding data instance $p_m^t$ into two parts: $p_{1km}^t \in \mathbb{R}^k (p_{0km}^t = p_{1km}^t, \forall t, k, m)$ and $p_{2km}^t \in \mathbb{R}^o/\mathbb{R}^k$, as in Line 5. We determine the weight for the base decision function $f_{\iota km}^t(p_{\iota km}^t)$ in Line 7, and determine the weight for the strong classifier $\mathcal{F}_{kim}^t(p_m^t)$ in Line 8. Then, for this data sample, we calculate the value of the strong classifier $\mathcal{F}_{kim}^t(p_m^t)$ in Line 10, and conduct the joint inference as a weighted sum of the

**Algorithm 6:** Conditional Classifier Placement

**Input:** $\boldsymbol{Y}^t, \Delta\mathcal{X}_{-SC}, \hat{t}$

given $\boldsymbol{Y}^t$, get $\widetilde{\boldsymbol{X}}^t, \widetilde{\boldsymbol{U}}^t, \widetilde{\boldsymbol{u}}^t, \widetilde{\boldsymbol{V}}^t$ by invoking **Algorithm 5**;

**if** $\mathcal{X}^t_{SC}(\widetilde{\boldsymbol{X}}^t, \boldsymbol{X}^{\hat{t}}) \leq \frac{1}{\rho_2}\Delta\mathcal{X}_{-SC}$ **then**

  $\boldsymbol{X}^t = \widetilde{\boldsymbol{X}}^t$;

  $\Delta\mathcal{X}_{-SC} = \mathcal{X}^t_{-SC}(\widetilde{\boldsymbol{X}}^t, \widetilde{\boldsymbol{U}}^t, \widetilde{\boldsymbol{u}}^t, \widetilde{\boldsymbol{V}}^t)$ ;

  $\hat{t} = t$;

**else**

  $\boldsymbol{X}^t = \boldsymbol{X}^{\hat{t}}$;

  set $\boldsymbol{U}^t$, $\boldsymbol{u}^t$ and $\boldsymbol{V}^t$ according to $\boldsymbol{X}^t$ and $\boldsymbol{Y}^t$;

  $\Delta\mathcal{X}_{-SC} = \Delta\mathcal{X}_{-SC} + \mathcal{X}^t_{-SC}(\boldsymbol{X}^{\hat{t}}, \boldsymbol{U}^t, \boldsymbol{u}^t, \boldsymbol{V}^t)$;

**Output:** $\boldsymbol{X}^t$, $\boldsymbol{U}^t$, $\boldsymbol{u}^t$, $\boldsymbol{V}^t$, $\hat{t}$

**Algorithm 7:** Inference Aggregation Placement

**Input:** $\Delta\mathcal{Y}_{-SC}$, $\hat{t}$, $\check{t}$

**for** $i \in [I]$ **do**

  set $\widetilde{\boldsymbol{Y}^t}$ as $\widetilde{Y}^t_i = 1$, and $\widetilde{Y}^t_j = 0$ for $j \neq i$;

  given $\widetilde{\boldsymbol{Y}^t}$, get $\widetilde{\boldsymbol{X}}^t, \widetilde{\boldsymbol{U}}^t, \widetilde{\boldsymbol{u}}^t, \widetilde{\boldsymbol{V}}^t$ by invoking **Algorithm 6**;

  **if** $\mathcal{Y}^t_{SC}(\widetilde{\boldsymbol{Y}}^t, \boldsymbol{Y}^{\check{t}}) \leq \frac{1}{\rho_1}\Delta\mathcal{Y}_{-SC}$ **then**

    $\boldsymbol{Y}^t = \widetilde{\boldsymbol{Y}}^t$;

    $\Delta\mathcal{Y}_{-SC} = \mathcal{Y}^t_{-SC}(\widetilde{\boldsymbol{X}}^t, \widetilde{\boldsymbol{Y}}^t, \widetilde{\boldsymbol{U}}^t, \widetilde{\boldsymbol{u}}^t, \widetilde{\boldsymbol{V}}^t)$;

    $\check{t} = t$;

  **else**

    $\boldsymbol{Y}^t = \boldsymbol{Y}^{\check{t}}$;

    given $\boldsymbol{Y}^t$, get $\widetilde{\boldsymbol{X}}^t, \widetilde{\boldsymbol{U}}^t, \widetilde{\boldsymbol{u}}^t, \widetilde{\boldsymbol{V}}^t$ by invoking **Algorithm 6**;

    $\Delta\mathcal{Y}_{-SC} = \Delta\mathcal{Y}_{-SC} + \mathcal{Y}^t_{-SC}(\widetilde{\boldsymbol{X}}^t, \boldsymbol{Y}^t, \widetilde{\boldsymbol{U}}^t, \widetilde{\boldsymbol{u}}^t, \widetilde{\boldsymbol{V}}^t)$;

  $H^t_{-HM} = \mathcal{Y}^t_{-SC} + \mathcal{Y}^t_{SC}$;

find the minimum $H^t_{-HM}$ for all $i$ and its $\boldsymbol{X}^t, \boldsymbol{Y}^t, \boldsymbol{U}^t, \boldsymbol{u}^t, \boldsymbol{V}^t$;

**Output:** $\boldsymbol{X}^t$, $\boldsymbol{Y}^t$, $\boldsymbol{U}^t$, $\boldsymbol{u}^t$, $\boldsymbol{V}^t$, $\hat{t}$, $\check{t}$

strong classifiers' results in Line 11. As receiving the ground-truth label in Line 12, we next decrease the weights of those classifiers which misclassify the instances in Lines 15 $\sim$ 23. Finally, we update online classifiers based on their loss on the current data sample, as in Lines 25 $\sim$ 30.

### *D. Performance Analysis*

Our analysis is organized as follows. First, by Lemma 2, we show that Algorithm 5 is a polynomial-time algorithm producing the feasible solution to the problem (7). Second, by Theorem 4 on top of Lemma 2, we derive the approximation ratio $r_4$ of Algorithm 5. Then, by Theorem 5, we show that the total number of the mistakes incurred by the heterogeneous online transfer learning possesses an upper bound. Finally, by Theorem 6, we show the competitive ratio of Algorithm 8.

**Lemma 2.** *Algorithm 5 returns feasible solutions to both the problem (7) and the problem (8) in the polynomial time.*

**Algorithm 8:** Heterogeneous Online Transfer Learning

**Input:** offline classifiers $f^1_{0k0}(\cdot)$, trade-off $\mathcal{C}$, and weight discount $\beta_1, \beta_2 \in (0,1)$

**Initialize:** $t = 1$, $\hat{t} = \check{t} = 0$, $f^1_{1k0} = \emptyset, f^1_{2k0} = \emptyset$, $\psi^1_{\iota k0} = 1/3$, $\zeta^1_{k0} = 1/K$

**for** $t = 1, 2, ..., T$ **do**

  invoke **Algorithm 7** to obtain $\boldsymbol{X}^t, \boldsymbol{Y}^t$;

  **for** $m = 0, 1, ..., M^t$ **do**

    Split $p^t_m$ into two instance: $p^t_{1km}(p^t_{0km} = p^t_{1km}, \forall t, k, m), p^t_{2km}$ ;

    ▷ **Weights update**

    $W^t_{\iota kim} = \begin{cases} \psi^t_{\iota km} / \sum_\iota \psi^t_{\iota km}, & Y^t_i = 1 \\ 0, & Y^t_i = 0 \end{cases}$

    $P^t_{kim} = \begin{cases} \zeta^t_{km} / \sum_k \zeta^t_{km}, & Y^t_i = 1 \\ 0, & Y^t_i = 0 \end{cases}$

    ▷ **Label inference**

    $\mathcal{F}^t_{kim}(p^t_m) = sign\big(\sum_\iota W^t_{\iota kim} sign(f^t_{\iota km}(p^t_{\iota km}))\big)$;

    calculate inference: $\hat{q}^t_m = sign\Big[\sum_k \sum_i P^t_{kim} \mathcal{F}^t_{kim}(p^t_m)\Big]$;

    receive ground-truth: $q^t_m \in \{-1, +1\}$;

    **for** $k = 1, 2, ..., K$ **do**

      ▷ **Parameters update**

      **for** $\iota = 0, 1, 2$ **do**

        **if** $sign(q^t_m f^t_{\iota km}(p^t_{\iota km})) < 0$ **then**

          $\psi^t_{\iota km+1} = \psi^t_{\iota km} \beta_2$;

        **else**

          $\psi^t_{\iota km+1} = \psi^t_{\iota km}$;

      **if** $sign(q^t_m \sum_i \mathcal{F}^t_{kim}(p^t_m)) < 0$ **then**

        $\zeta^t_{km+1} = \zeta^t_{km} \beta_1$;

      **else**

        $\zeta^t_{km+1} = \zeta^t_{km}$;

      ▷ **Online classifiers update**

      **for** $\iota = 1, 2$ **do**

        calculate loss: $l^t_\iota = [1 - q^t_m f^t_{\iota km}(p^t_{\iota km})]^+$;

        **if** $l^t_\iota > 0$ **then**

          $f^t_{\iota km+1} = f^t_{\iota km} + \alpha^t_{\iota m} q^t_m \mathbb{K}_{k,\iota}(p^t_{\iota km}, \cdot)$, where $\alpha^t_{\iota m} = \min\{\mathcal{C}, \frac{l^t_\iota}{\mathbb{K}_{k,\iota}(p^t_{\iota km}, p^t_{\iota km})}\}$;

        **else**

          $f^t_{\iota km+1} = f^t_{\iota km}$;

*Proof.* A solution is feasible for a problem if the solution satisfies the problem's constraints. For (7), (7a) is satisfied by Line 11. Lines 5$\sim$6 ensure no violation of the edge capacity limit, i.e., (7b). Once the edges to host classifiers are determined, (7c) and (7d) are also satisfied, following Lines 11 and 20. Lines 13 and 19 guarantee (7e). For (8), this inequality $\sum_\iota \sum_j (\kappa_{\iota j} + \varsigma_i + \widehat{A}_{\iota ji}) + d_i + \sum_m d_{mi} + \sum_{\iota,m} \ell_{\iota mi} \geq 0$ is constructed based on (8a), (8b), (8c) and (8d), guaranteeing they are satisfied according to Lines 4$\sim$7. As for the time

complexity of Algorithm 5, the *for* loop runs $3K$ times, and the *while* loop in the *for* loop runs at most $I$ times according to its termination condition $\Delta E_{i^+} + 1 > E_{i^+}$. Thus, the total time complexity is $O(KI)$. □

**Theorem 4.** *Algorithm 5 is an $r_4$-approximation algorithm to the problem (7), i.e., $H_7 \leq r_4 H_7^*$, where $r_4 = \frac{\varpi}{\varpi - 1}$.*

*Proof.* See Appendix D. □

We introduce several new notations to simplify our descriptions. Now, we split $H_5$, the objective function of (5), as $H_5 = \sum_t (H_{HM}^t + H_{-HM}^t)$, where $H_{HM}^t = \sum_m \mathbb{I}\Big\{sign\Big[q_m^t \cdot (\sum_k \sum_i P_{kim}^t \mathcal{F}_{kim}^t(p_m^t))\Big] < 0\Big\}$ and $H_{-HM}^t = \sum_\iota \sum_k \sum_i (A_{\iota ki}^t X_{\iota ki}^t + C_{\iota ki}[X_{\iota ki}^t - X_{\iota ki}^{t-1}]^+) + \sum_i (B_i^t Y_i^t + C_i[Y_i^t - Y_i^{t-1}]^+) + \sum_m \sum_i (d_{mi}^t (u_{mi}^t + 2Y_i^t)) + \sum_i \sum_j d_{ij}^t V_{ij}^t + \sum_m \sum_i d_{mi}^t U_{mi}^t$. We also use $H_{HM}^{t*}$ and $H_{-HM}^{t*}$ to denote their optimal values.

**Theorem 5.** *Algorithm 8 incurs the total number of mistakes as $\sum_t H_{HM}^t \leq \Upsilon_{min}(4(\frac{1+\beta_2}{2\beta_2} + \frac{\ln 3}{1-\beta_2})(\frac{1+\beta_1}{2\beta_1} + \frac{\ln K}{1-\beta_1}))$, where $\Upsilon_{\min} = \min\{\Upsilon^k\}$, $\Upsilon^k = \min\{\Lambda_0^k, \Lambda_1^k, \Lambda_2^k\}$. Here, $\Lambda_\iota^k = \sum_{t=1}^T \sum_{m=0}^{M^t} \mathbb{I}\{sign[q_m^t f_{\iota km}^t(p_{\iota km}^t)] < 0\}, \forall \iota \in \{0,1,2\}$, $\mathbb{I}\{sign[q_m^t f_{\iota km}^t(p_{\iota km}^t)] < 0\}$ indicates whether the inference computed by the offline or online classifier is wrong for the data sample $m$ of the time slot $t$, respectively, as in Algorithm 8.*

*Proof.* See Appendix E. □

**Theorem 6.** *Algorithm 8 is an $\mathcal{R}$-competitive online algorithm to the problem (5), i.e., $H_5 \leq \mathcal{R} H_5^*$, where $\mathcal{R} = \max\{r_5, r_6\}$, $r_5 = 4(\frac{1+\beta_2}{2\beta_2} + \frac{\ln 3}{1-\beta_2})(\frac{1+\beta_1}{2\beta_1} + \frac{\ln K}{1-\beta_1})$, and $r_6 = (1 + \frac{1}{\rho_1})(1 + \frac{1}{\rho_2} + \mathcal{D}_{max}) r_4 \sigma_1$. Here, $\beta_1$ and $\beta_2$ are constants in Algorithm 8; $\rho_1$ and $\rho_2$ are constants in Algorithms 6 and 7, as in Algorithms 2 and 3; $r_4$ is the approximation ratio of Algorithm 5, as in Theorem 4; $\mathcal{D}_{max} = \max\{\max_{t,\iota,k,i}\{\frac{B_i^t}{3A_{\iota ki}^t K}\}, 2\}$; and $\sigma_1 = \max_t\{\frac{\max_i\{\sum_{\iota,k} A_{\iota ki} + \sum_j d_{ij} + 2\sum_m d_{mi}\}}{\min_i \sum_{\iota,k} A_{\iota ki}}\}$.*

*Proof.* See Appendix F. □

## VI. Experimental Study

### A. Experimental Settings

***Transfer Learning Datasets:*** We use the text classification dataset `20Newsgroups` [29], which contains nearly 20,000 newsgroup documents with 61,188 unique words (i.e., features), associated to multiple topics. Each topic has several sub-topics, and each document has been labelled with one and only one sub-topic. We consider the 8843 documents associated to all the sub-topics of *comp* and *sci*, as in Table V. We treat all the sub-topics of *comp* as the label of $+1$, and all the sub-topics of *sci* as the label of $-1$. By matching one sub-topic from $+1$ with another sub-topic from $-1$, we have 20 pairs of sub-topics in total and for each of such pairs, we can train a Support Vector Machine (SVM). All our evaluations use this dataset by default, unless explicitly specified otherwise.

We also use the dataset `Wine Reviews` [30], which contains 130,000 reviews with the information on variety, location, winery, price, points, and description. We use `description` as the feature and treat `points` no less than 90 as the label $+1$, and `points` less than 90 as the label $-1$. We use `country` to identify the top twelve countries with the highest data volume. For each of these countries, we can train an SVM.

**TABLE V:** Sub-topics of Documents

| Label of $+1$ | Label of $-1$ |
|---|---|
| comp.graphics | sci.crypt |
| comp.os.ms-windows.misc | sci.electronics |
| comp.sys.ibm.pc.hardware | sci.med |
| comp.sys.mac.hardware | sci.space |
| comp.windows.x | |

***Transfer Learning Classifiers:*** For the first dataset as aforementioned, for the homogeneous transfer learning setting, we select the SVM with *comp.windows.x* and *sci.space* as our online classifier (with a linear kernel function) to be trained, and the rest 19 SVMs as our existing offline classifiers. For the heterogeneous transfer learning setting, we select the rest 19 SVMs as our offline classifiers. The feature space is divided into 19 portions on average, corresponding to the feature space of each offline classifier. We select the 38 SVMs as our online classifier (with a linear kernel function) to be trained.

For the second dataset as aforementioned, for the homogeneous transfer learning setting, we select the SVM with `US` as our online classifier (with a linear kernel function) to be trained, and the remaining 11 SVMs as our existing offline classifiers. For the heterogeneous transfer learning setting, we select the remaining 11 SVMs as our offline classifiers. The feature space is divided into 11 portions on average, corresponding to the feature space of each offline classifier. We select the 22 SVMs as our online classifiers (with a linear kernel function) to be trained.

We note that in the heterogeneous case, two online classifiers correspond to one offline classifier, where one online classifier is trained with the same features as the offline classifier and the other is trained with the remaining features.

***Edge Networks and Data Samples:*** We adopt the data of the 268 underground stations in London with dynamic passenger counts [26]. We choose the first 25 stations based on the total passenger count at each station, and envisage that each of such stations has an edge. We study the system for $T = 24$ hours and set the length of a single time slot as 15 minutes. We consider one document as one data sample. Based on the ratio of the passenger count at each edge in each time slot over the total passenger count across all edges and time slots, we spread the 8843 documents proportionally. We use the geographical distance [27] to estimate the network delay between the two edges. We set the dynamic operational cost as within $[2, 8]$ cents/kWh, following the wholesale electricity prices [28]. We vary the unit start-up cost as multiplied by a weight in order to demonstrate a spectrum of different results. Note that the operational and start-up costs for both the homogeneous and heterogeneous settings are the same. We assume that each edge can host $2 \sim 8$ VMs or classifiers at most.

***Algorithms and Implementation:*** We implement the following algorithms for comparison: (i) `Proposed` refers to our proposed homogeneous or heterogeneous online transfer learning algorithms; (ii) `Delay_only` chooses edges for classifiers only based on optimizing delay regardless of other costs,

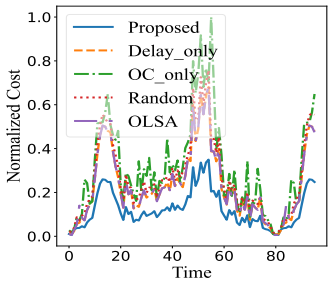



**Fig. 4:** Total Cost per Time Slot

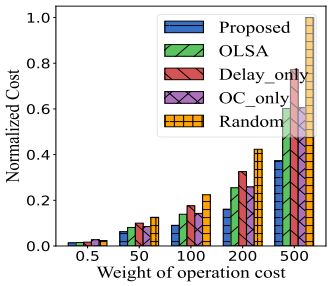



**Fig. 5:** Impact of Operational Cost

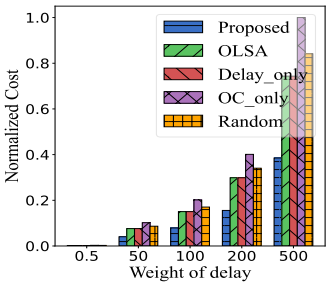



**Fig. 6:** Impact of Delay

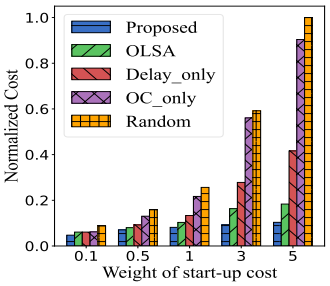



**Fig. 7:** Impact of Start-up Cost

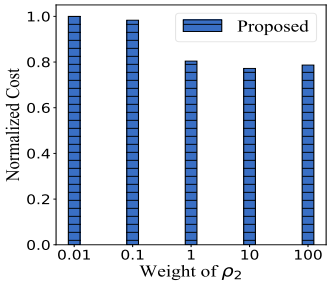



**Fig. 8:** Impact of $\rho_2$

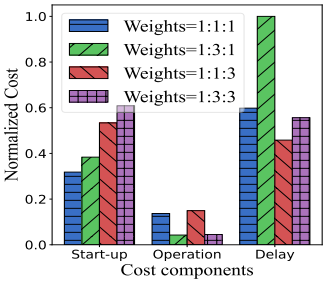



**Fig. 9:** Cost Dissection

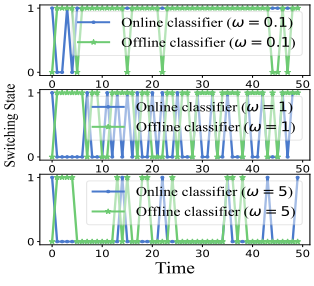



**Fig. 10:** Switching Operations

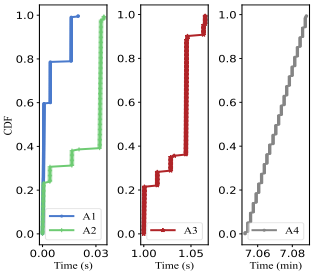



**Fig. 11:** Execution Time

and directly downloads classifiers and/or preparing VMs as the one-shot optimum indicates (i.e., without postponing start-up); (iii) `OC_only` chooses edges only based on optimizing operational cost regardless of others and without postponing state switching; (iv) `Random` selects edges randomly without considering any cost optimization; (v) `OLSA` is a state-of-the-art method that selects edges based on delay, operational cost, and start-up cost via a so-called lazy switching strategy [31].

We also adopt and implement state-of-the-art homogeneous and heterogeneous online transfer learning algorithms for comparison. For homogeneous settings, we implement (i) `HomOTL_1` [32], (ii) `HomOTL_2` [32], and (iii) `OMTL_MC` [33], which all dynamically update the combination weights for the classifiers according to their performance. `HomOTL_1` uses a fixed combination of online and offline classifiers, while `HomOTL_2` adjusts the weights based on prediction mistakes. `OMTL_MC` employs a two-stage ensemble strategy by first combining each offline classifier with the online classifier, and then aggregating the resulting models. For heterogeneous settings, we implement (iv) `HetOTL_3` [32], which partitions the feature space and uses separate classifiers for each part, updating their weights based on prediction mistakes, and (v) `OHKT`, which is an approach that predicts the combination weights via SVM [34].

Our implementation includes around 9,000 lines of Python codes, and we conduct all evaluations on a commodity laptop with an Intel Core i5 2.9-GHz CPU and 8-GB RAM.

### *B. Results for Homogeneous Setting*

***Total Cost:*** Fig. 4 visualizes the total cost of the different algorithms per time slot as time goes. `Proposed` always has the lowest total cost, achieving 47% less total cost than `Delay_only`, 61% less total cost than `OC_only`, 54% less total cost than `Random`, and 46% less total cost than `OLSA`. This improvement comes from our method's ability to jointly optimize delay, operational cost, and start-up cost. In contrast, `Delay_only` and `OC_only` overlook important cost components, resulting in either high latency or frequent switching. `Random` performs worst due to its lack of optimization, while `OLSA` considers multiple costs but lacks joint optimization of system capacity constraints in the online setting.

***Impact of Operational Cost and Delay:*** Fig. 5 and Fig. 6 compare the total cost incurred by different algorithms as the weight of the operational cost and the weight of the delay varies, respectively. `Proposed` beats others no matter how these weights change. In Fig. 5, `OC_only` which only optimizes operational cost embodies more advantages compared to `Delay_only` and `Random`, and is getting closer to `OLSA` because the weight of the operational cost increases; yet, as the operational cost becomes more weighted, `Proposed` still outperforms all others by balancing among the different cost components. The maximum cost reduction of `Proposed` is 51.3% compared to `Delay_only`, 37.3% compared to `OC_only`, 63.1% compared to `Random`, and 37.0% compared to `OLSA`. In Fig. 6, `Proposed` yields the maximum reduction of 48.3%, 61.5%, 55.6%, and 48.2% compared to `Delay_only`, `OC_only`, `Random`, and `OLSA`.

***Impact of Start-up Cost:*** Fig. 7 compares the normalized total costs of different algorithms as the weight on the start-up cost varies. `Proposed` beats all others. `Proposed` beats `Delay_only` and `OC_only` because the latter always pursue one-shot optimum in each time slot and essentially neglect the start-up cost. Thus, as the weight grows (i.e., the start-up becomes more dominating), `Proposed` becomes better. The maximum cost reduction of `Proposed` is 300% compared to `Delay_only`, 450% compared to `OC_only`, 500% compared to `Random`, and 75% compared to `OLSA`.

***Impact of $\rho_2$:*** Fig. 8 investigates how $\rho_2$ affects the total cost. Algorithm 2 is controlled by its parameter $\rho_2$, which is used to compare the cumulative non-start-up cost against the start-up cost and then determine whether to change the offline classifier placement. The total cost decreases as $\rho_2$ goes up

when below 10, and as $\rho_2$ goes down when exceeding 10, respectively. A smaller $\rho_2$ incurs frequent switches and a larger $\rho_2$ means a stricter criterion for the switch, both of which could lead to suboptimal costs. Therefore, $\rho_2$ needs to be carefully configured to prevent the suboptimums.

***Cost Dissection:*** Fig. 9 illustrates each cost component as the weights associated to the start-up cost, the operating cost, and the delay vary. We see that the larger the weight is, the smaller the corresponding cost becomes. That is, our approach can indeed work with different weight configurations to optimize different cost components to different extents. The weight configurations can be controlled by the service provider following its own needs and preferences.

***Switching Operations:*** Fig. 10 shows how the variation of the weight of the start-up cost, denoted as $\omega$, influences the total number of the occurrences of state switching for the online classifier (i.e., preparing VMs) and the offline classifiers (i.e., downloading classifiers plus preparing VMs). In this figure, 1 means new and different decisions are applied to the current time slot; 0 means decisions of the previous time slot are applied to the current time slot. Our approach incurs more frequent state switching (i.e., there are more 1s than 0s) when $\omega$ is small because the state switching criterion can be satisfied easily, and leads to less frequent state switching as $\omega$ becomes larger, due to a stricter criterion.

***Execution Time:*** Fig. 11 depicts the cumulative distribution of the execution time of each of our proposed algorithms. Algorithms 1$\sim$3 can finish within several seconds per 15-minute-long single time slot. For the time horizon of 24 hours, it takes no more than 7.1 minutes in total to finish across all time slots, which includes the homogeneous transfer learning process. Hence, our proposed algorithms are practically efficient.

***Mistakes of Homogeneous Transfer Learning:*** Fig. 12 presents the rate of the mistakes (i.e., the ratio of the number of incorrect inferences compared to the ground-truth over the total number of inferences). With both datasets, `Proposed` is always effective in transferring knowledge from existing classifiers, with an acceptable rate of mistakes lower than those of `HomOTL_1`, `HomOTL_2`, and `OMTL_MC`. This benefits from our adaptive weight update strategy, which exponentially downweights misclassified results, allowing our algorithm to focus more on reliable classifiers and adapt quickly to changing data.

### C. Results for Heterogeneous Setting

***Cost of Heterogeneous Transfer Learning:*** We omit the visualization of such results here because we observe similar trends in the heterogeneous transfer learning setting compared to Fig. 5, Fig. 6, Fig. 8, Fig. 7 and Fig. 9 in the homogeneous transfer learning setting. It takes no more than 8 minutes in total to finish running Algorithms 5, 6, 7, and 8.

***Mistakes of Heterogeneous Transfer Learning:*** Fig. 13 presents the rate of the mistakes of different algorithms for different datasets. `Proposed` has lower mistake rates than those other heterogeneous transfer learning algorithms. This is attributed to the hierarchical integration of offline and online classifiers and the adaptive weight updates, which enhance reliable predictions and suppress erroneous ones for stable performance across dynamic environments.

***Impact of Feature Spaces of Offline Classifiers:*** Fig. 14 reflects the impact of the feature spaces of the offline classifiers on the mistake rate. The mistake rate drops as the feature spaces of the offline classifiers become larger. This implies that the more information about the features we have, the better inference accuracy we can achieve in transfer learning.

***Edge Occupation:*** In Fig. 15, we compare the number of edges occupied by offline and online classifiers in each time slot in both homogeneous and heterogeneous settings. In the homogeneous setting, the number of edges occupied by offline classifiers is 15 or less, while the number of edges occupied by online classifiers is only 1. In the heterogeneous setting, online classifiers occupy more edges than offline classifiers due to the greater number of online classifiers compared to offline classifiers. Across the two settings, the offline classifiers occupy a similarly varying number of the edges.

## VII. Related Work

We discuss existing research in two groups, and for each group, we highlight its insufficiency compared to our work.

***Transfer Learning Methods and Systems:*** Daga et al. [8] designed a distributed transfer learning system for adapting to varying workload and data shift. Wu et al. [14] proposed online transfer learning for both homogeneous and heterogeneous environments. Yang et al. [15] focused on evaluating which source domain could be more suitable for transfer learning and the amount of knowledge transferred. Ding et al. [16] studied the minimization of the divergence among different sources by realizing cross-domain and cross-source knowledge transfer. Yang et al. [17] minimized domain discrepancy by promoting positive knowledge and decreasing the effect of unrelated instances. Yan et al. [18] introduced neural data servers to select relevant data in the transfer learning process.

These works focus on transfer learning, and almost all of them neglect resource usage and cost minimization from the systems perspective. The last work mentioned above is not typically for the cloud-edge and 5G environments.

***Cloud-Edge System and Network Optimization:*** Castellano et al. [19] explored optimal partitioning of shared resources in heterogeneous edge networks. Wang et al. [20] studied online resource allocation for edge computing in response to high dynamism of user mobility. Tu et al. [21] developed distributed learning optimization of the costs associated to device processing, offloading, and data disgarding. Meng et al. [22] optimized bandwidth and computing resource for deadline-restricted tasks. You et al. [23] explored dynamic resource provisioning in edge networks. Zhou et al. [24] proposed an online framework for cost-efficiency of cross-edge service functions. Han et al. [25] minimized the response time for latency-sensitive jobs in edge-cloud computing.

These works study cloud-edge systems and networks, but are unfortunately not about (distributed) transfer learning which has unique computing and and communication pattern. Thus, such existing research generally do not apply.

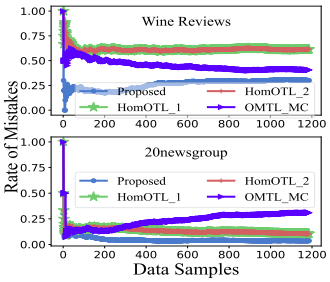



**Fig. 12:** Mistakes of Homogeneous Setting

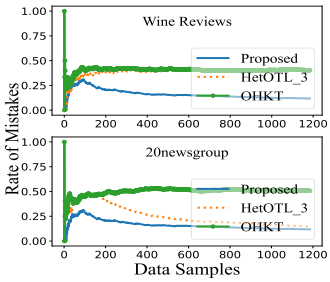



**Fig. 13:** Mistakes of Heterogeneous Setting

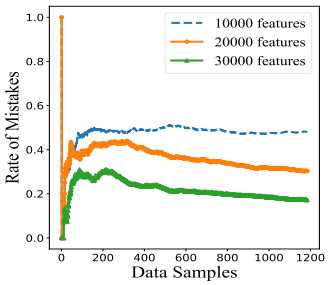



**Fig. 14:** Mistakes Impacted by Feature Space

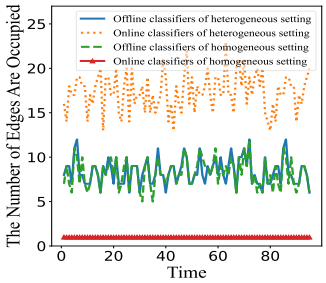



**Fig. 15:** Placement of Classifiers

## VIII. Conclusion

Transfer learning is a useful and important technique, yet gets largely overlooked in the context of mobile communication networks. This paper aims to bridge this gap. We consider both homogeneous and heterogeneous online transfer learning settings and formulate non-linear mixed-integer programs by considering operational cost of edges, delay of networks, start-up cost of downloading classifiers and preparing local edge environments, and the performance of transfer learning in terms of the mistakes of the combined classifiers. We design online optimization algorithms and prove their theoretical guarantees. Using real-world data, we conduct extensive expriments and validate the practical efficacy and efficiency of our algorithms. For future work, we intend to further explore transfer learning that involves pre-trained foundation models such as the Large Language Models (LLMs) in the cloud-edge environments.

## References


[1] Y. Yuan, L. Jiao, K. Zhu, X. Lin, and L. Zhang, "Ai in 5g: The case of online distributed transfer learning over edge networks," in *IEEE INFOCOM*, 2022.

[2] A. Kiani, N. Ansari, and A. Khreishah, "Hierarchical capacity provisioning for fog computing," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 962–971, 2019.

[3] A. Alnoman and A. Anpalagan, "Computing-aware base station sleeping mechanism in h-cran-cloud-edge networks," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 958–967, 2021.

[4] A. Anand, G. De Veciana, and S. Shakkottai, "Joint scheduling of urllc and embb traffic in 5g wireless networks," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 477–490, 2020.

[5] J. Tang, B. Shim, and T. Q. Quek, "Service multiplexing and revenue maximization in sliced c-ran incorporated with urllc and multicast embb," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 881–895, 2019.

[6] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832, 2018.

[7] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014.

[8] H. Daga, P. K. Nicholson, A. Gavrilovska, and D. Lugones, "Cartel: A system for collaborative transfer learning at the edge," in *ACM SoCC*, 2019.

[9] T. Tommasi, F. Orabona, and B. Caputo, "Learning categories from few examples with multi model knowledge transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 928–941, 2013.

[10] N. Segev, M. Harel, S. Mannor, K. Crammer, and R. El-Yaniv, "Learn on source, refine on target: A model transfer learning framework with random forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1811–1824, 2016.

[11] H. McKay, N. Griffiths, P. Taylor, T. Damoulas, and Z. Xu, "Online transfer learning for concept drifting data streams." in *BigMine @ ACM KDD*, 2019.

[12] 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.

[13] L. Zhang, C. Wu, Z. Li, C. Guo, M. Chen, and F. C. Lau, "Moving big data to the cloud: An online cost-minimizing approach," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 12, pp. 2710–2721, 2013.

[14] Q. Wu, H. Wu, X. Zhou, M. Tan, Y. Xu, Y. Yan, and T. Hao, "Online transfer learning with multiple homogeneous or heterogeneous sources," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 7, pp. 1494–1507, 2017.

[15] L. Yang, L. Jing, J. Yu, and M. K. Ng, "Learning transferred weights from co-occurrence data for heterogeneous transfer learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 11, pp. 2187–2200, 2016.

[16] Z. Ding, M. Shao, and Y. Fu, "Incomplete multisource transfer learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 2, pp. 310–323, 2018.

[17] C. Yang, Y.-M. Cheung, J. Ding, and K. C. Tan, "Concept drift-tolerant transfer learning in dynamic environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3857–3871, 2022.

[18] X. Yan, D. Acuna, and S. Fidler, "Neural data server: A large-scale search engine for transfer learning data," in *IEEE/CVF CVPR*, 2020.

[19] G. Castellano, F. Esposito, and F. Risso, "A distributed orchestration algorithm for edge computing resources with guarantees," in *IEEE INFOCOM*, 2019.

[20] L. Wang, L. Jiao, J. Li, J. Gedeon, and M. Mühlhäuser, "Moera: Mobility-agnostic online resource allocation for edge computing," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1843–1856, 2019.

[21] Y. Tu, Y. Ruan, S. Wagle, C. G. Brinton, and C. Joe-Wong, "Network-aware optimization of distributed learning for fog computing," in *IEEE INFOCOM*, 2020.

[22] J. Meng, H. Tan, C. Xu, W. Cao, L. Liu, and B. Li, "Dedas: Online task dispatching and scheduling with bandwidth constraint in edge computing," in *IEEE INFOCOM*, 2019.

[23] W. You, L. Jiao, S. Bhattacharya, and Y. Zhang, "Dynamic distributed edge resource provisioning via online learning across timescales," in *IEEE SECON*, 2020.

[24] Z. Zhou, Q. Wu, and X. Chen, "Online orchestration of cross-edge service function chaining for cost-efficient edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 8, pp. 1866–1880, 2019.

[25] Z. Han, H. Tan, X.-Y. Li, S. H.-C. Jiang, Y. Li, and F. C. M. Lau, "Ondisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds," *IEEE/ACM Transactions on Networking*, vol. 27, no. 6, pp. 2472–2485, 2019.

[26] "London underground passenger counts data," https://tfl.gov.uk/info-for/open-data-users/.

[27] "List of london underground stations," https://wiki.openstreetmap.org/wiki/List_of_London_Underground_stations.

[28] "Hourly pricing," https://hourlypricing.comed.com/live-prices/.

[29] "20newsgroups," http://qwone.com/~jason/20Newsgroups/.

[30] "Wine review," https://www.kaggle.com/datasets/zynicide/wine-reviews.

[31] B. Gao, Z. Zhou, F. Liu, F. Xu, and B. Li, "An online framework for joint network selection and service placement in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 11, pp. 3836–3851, 2021.

[32] P. Zhao, S. Hoi, J. Wang, and B. Li, "Online transfer learning," *Artificial Intelligence*, vol. 216, pp. 76–102, 2014.

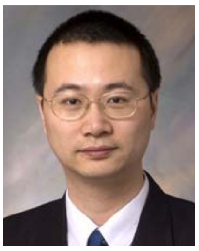
[33] Z. Kang, B. Yang, S. Yang, X. Fang, and C. Zhao, "Online transfer learning with multiple source domains for multi-class classification," *Knowledge-Based Systems*, vol. 190, p. 105149, 2020.
[34] H. Wu, Y. Yan, Y. Ye, H. Min, M. K. Ng, and Q. Wu, "Online heterogeneous transfer learning by knowledge transition," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 3, pp. 1–19, 2019.

**Konglin Zhu** received the master's degree in computer Science from the University of California, Los Angeles, CA, USA, and the Ph.D. degree from the University of Göttingen, Germany, in 2009 and 2014, respectively. He is now a Professor with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include Internet of Vehicles, Edge Computing and Distributed Learning.

**Fei Wang** received the master's degree in Information and Communication Engineering from Harbin Engineering University, China, in 2021. He is currently working towards the Ph.D. degree in School of Artificial Intelligence in Beijing University of Posts and Telecommunications. His research interests are in the areas of online learning and federated learning.

**Lei Jiao** received the Ph.D. degree in computer science from the University of Göttingen, Germany. He is currently with the University of Oregon, USA, and was previously a member of technical staff at Nokia Bell Labs, Ireland. He researches AI infrastructures, cloud/edge networks, energy systems, cybersecurity, and multimedia. He has published 80+ papers mainly in leading journals such as IEEE Journal on Selected Areas in Communications, IEEE/ACM Transactions on Networking, IEEE Transactions on Mobile Computing, and IEEE Transactions on Parallel and Distributed Systems and conferences such as INFOCOM, MOBIHOC, ICDCS, SECON, and ICNP. He is a U.S. National Science Foundation CAREER awardee, and is also a recipient of the Ripple Faculty Fellowship, the Alcatel-Lucent Bell Labs UK and Ireland Recognition Award, and the Best Paper Awards of IEEE CNS 2019 and IEEE LANMAN 2013. He has been on the program committees as a track chair for ICDCS and as a member for many conferences such as INFOCOM, MOBIHOC, ICDCS, WWW, and IWQoS, and has also served as the program chair of multiple workshops with INFOCOM and ICDCS.

**Yulan Yuan** received the master degree in Information and Communication Engineering, and the B.Eng degree in Telecommunications Engineering with Management from Beijing University of Posts and Telecommunications, China, in 2019 and 2022, respectively. She is currently pursuing the Ph.D. degree at Hong Kong University of Science and Technology (HKUST). Her research interests are in the areas of stochastic optimization and federated learning.



**Xiaojun Lin** received his B.S. from Zhongshan University, Guangzhou, China, in 1994, and his M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 2000 and 2005, respectively. He joined the School of Electrical and Computer Engineering at Purdue University in 2005, and became a Professor of ECE in 2017. Since June 2023, he joined the Department of Information Engineering, The Chinese University of Hong Kong, as a Professor and Global STEM Scholar. Dr. Lin's research interests are in the analysis, control and optimization of large and complex networked systems, including both communication networks and power grid. He received the NSF CAREER award in 2007. He received 2005 best paper of the year award from Journal of Communications and Networks, IEEE INFOCOM 2008 best paper award, ACM MobiHoc 2021 best paper award, and ACM e-Energy 2022 best paper award. He was the Workshop co-chair for IEEE GLOBECOM 2007, the Panel co-chair for WICON 2008, the TPC co-chair for ACM MobiHoc 2009, the Mini-Conference co-chair for IEEE INFOCOM 2012, and the General cochair for ACM e-Energy 2019. He has served as an Area Editor for (Elsevier) Computer Networks Journal, an Associate Editor for IEEE/ACM Transactions on Networking, and a Guest Editor for (Elsevier) Ad Hoc Networks journal.

**Lin Zhang** received the B.S. and the Ph.D. degrees in 1996 and 2001, both from the Beijing University of Posts and Telecommunications, Beijing, China. From 2000 to 2004, he was a Postdoctoral Researcher with Information and Communications University, Daejeon, South Korea, and Nanyang Technological University, Singapore, respectively. He joined Beijing University of Posts and Telecommunications in 2004, where he has been a Professor since 2011. He is also the director of Beijing Big Data Center. His current research interests include mobile cloud computing and Internet of Things.

## APPENDIX

### A. Proof of Theorem 1

*Proof.* Let $\Delta P$ and $\Delta D$ denote the increment of the objective function in the problem (3) and (4), respectively, $\Delta D = -(\lambda_{k^*} + Q_{i^*}\Delta\delta_{i^*})$, where $\Delta\delta_{i^*} = \frac{\delta_{i^*}}{Q_i} + \frac{a_{i^*k^*}+d_{i^*}/K+\sum_m d_{mi^*}/K}{Q_i\xi}$ stands for the increment in $\delta_{i^*}$. Thus, we have $\Delta D = -(\lambda_{k^*} + Q_{i^*}\Delta\delta_{i^*}) = -\lambda_{k^*} - Q_i(\frac{\delta_{i^*}}{Q_i} + \frac{a_{i^*k^*}+d_{i^*}/K+\sum_m d_{mi^*}/K}{Q_i\xi}) = (1-\frac{1}{\xi})(a_{i^*k^*} + \frac{d_i^*}{K} + \frac{\sum d_{mi^*}}{K}) = \frac{\xi-1}{\xi}\Delta P$. Let $P^k$ and $D^k$ denote the value of the objective function in the problem (3) and (4) after handling the offline classifer $k$. Due to $H_3 = P^K = \sum_k(P^k - P^{k-1}) = \frac{\xi}{\xi-1}\sum_k(D^k - D^{k-1}) = \frac{\xi}{\xi-1}(D^K - D^0) = \frac{\xi}{\xi-1}D^K = \frac{\xi}{\xi-1}H_4 \le \frac{\xi}{\xi-1}H_3^{t*}$, followed by $P^0$ and $D^0$ are initialized with 0 and duality, we obtain $r_1 = \frac{\xi}{\xi-1}$. □

### B. Proof of Theorem 2

*Proof.* In order to simplify our proof, we introduce some new symbols. We use $p_n$ to identify $n$th data sample, use $\omega_{k,n}$ to replace $\zeta_{k,m}^t$ and $\psi_m^t$, where $k \in \{1, ..., K, K+1\}$ (including the offline and online classifiers), use $P_{k,n} = \frac{\omega_{k,n}}{\sum_k \omega_{k,n}}$ to denote the normalized weight and $m_{k,n}$ to denote the mistakes of the classifier $k$, which are all updated as our Algorithm 4 shows.

Firstly, we prove that $\mathbb{I}\{q_n \cdot \hat{q}_n < 0\} = \mathbb{I}\{\sum_k P_{k,n}m_{k,n} > 0.5\}$. By assuming that there are only $K_1$ classifiers predict correctly (i.e., $sign(f^k(p_n)) = q_n$), we have $\hat{q}_n = sign(q_n(\sum_{k=1}^{K_1} P_{k,n} - \sum_{k=K_1+1}^{K+1} P_{k,n}))$. Then, based on $\sum_k P_{k,n} = 1$, we obtain $q_n \cdot \hat{q}_n < 0 \iff \sum_{k=1}^{K1} P_{k,n} - \sum_{k=K_1+1}^{K+1} P_{k,n} < 0 \iff \sum_{k=K_1+1}^{K+1} P_{k,n} > 0.5 \iff \sum_k P_{k,n}m_{k,n} > 0.5$.

Next, we have $\ln(\frac{\sum_k \omega_{k,n+1}}{\sum_k \omega_{k,n}}) = \ln(\sum_k P_{k,n}\theta^{m_{k,n}}) \le -(1-\theta)\sum_k P_{k,n}m_{k,n}$, thus $\ln(\frac{\sum_k \omega_{k,N}}{\sum_k \omega_{k,1}}) \le -(1-\theta)\sum_{k,n} P_{k,n}m_{k,n}$, and have $\ln(\frac{\sum_k \omega_{k,N}}{\sum_k \omega_{k,1}})$ lower bounded as $\ln(\frac{\sum_k \omega_{k,N}}{\sum_k \omega_{k,1}}) \ge \ln(\omega_{k,1}\theta^{\sum_n m_{n,k}}) = \ln(\frac{1}{K+1}) + \sum_n m_{k,n}\ln(\theta)$. Based on the above, we have $\sum_{k,n} P_{k,n}m_{k,n} \le \frac{\ln(1/\theta)\sum_n m_{k,n} + \ln(K+1)}{1-\theta} \le \frac{\ln(1/\theta)\Gamma_{min} + \ln(K+1)}{1-\theta}$. Finally, we upper bound the mistakes as $\sum_n \mathbb{I}\{q_n \cdot \hat{q}_n < 0\} \le 2\sum_{k,n} P_{k,n}m_{k,n} \le \frac{2\ln(1/\theta)\Gamma_{min} + 2\ln(K+1)}{1-\theta}$. When we set $\theta = \sqrt{\Gamma_{min}}/(\sqrt{\Gamma_{min}} + \sqrt{\ln(K+1)})$, we further obtain $\big(2 + 2\sqrt{2\ln(K+1)}\big)\Gamma_{min} + 2\ln(K+1)$. □

### C. Proof of Theorem 3

*Proof.* Firstly, the start-up cost $\mathbb{X}_{SC}^t$ is no more than $\frac{1}{\rho_2}$ times $\Delta\mathbb{X}_{-SC}$ within $[\hat{t}, t-1]$. Hence we have $\sum_t \mathbb{X}_{SC}^t \le \frac{1}{\rho_2}\sum_t \mathbb{X}_{-SC}^t$ as the worst case, i.e., the change of offline classifier placement always happens at each $t$. Similarly, we can obtain $\sum_t \mathbb{Y}_{SC}^t \le \frac{1}{\rho_1}\sum_t \mathbb{Y}_{-SC}^t$. Then, we have $\sum_t H_{-M}^t \le \sum_t \mathbb{Y}_{SC}^t + \sum_t \mathbb{Y}_{-SC}^t \le (1+\frac{1}{\rho_1})\sum_t \mathbb{Y}_{-SC}^t$. Followed by the constraints of $\sum_i y_i^t = 1$ and $y_i^t \in \{0,1\}, \forall i$, we have $\sum_{t,i}(b_i^t + \sum_m 2d_{mi}^t)y_i^t \le \max\{\max_{i,k,t}\{\frac{b_i^t}{a_{k,i}^t K}\}, 2\}\sum_t \mathbb{X}_{-SC}^t$, and $\sum_t \mathbb{Y}_{-SC}^t = \sum_t(\mathbb{X}_{SC}^t + \mathbb{X}_{-SC}^t) + \sum_{t,i}(b_i^t + \sum_m 2d_{mi}^t)y_i^t \le (1 + \frac{1}{\rho_2} + D_{max})\sum_t \mathbb{X}_{-SC}^t$.

Next, we focus on $\frac{\max_{\boldsymbol{y}^t} \mathbb{X}_{-SC}(Alg_2(\boldsymbol{y}^t))}{\min_{\boldsymbol{y}^t} \mathbb{X}_{-SC}(Alg_2(\boldsymbol{y}^t))}$, where $Alg_2(\cdot)$ refers to Algorithm 2. We construct a new problem $P_0$ with $C_0 = \sum_{i,k} a_{ki}x_{ki} + \sum_{i,m} d_{mi}u_{mi} + \sum_{i,j} d_{ij}v_{ij}$ and the constraints of (1c)∼(1d), (1f)∼(1j), we can obtain $\frac{\max_{\boldsymbol{y}^t} \mathbb{X}_{-SC}(Alg_2(\boldsymbol{y}^t))}{\min_{\boldsymbol{y}^t} \mathbb{X}_{-SC}(Alg_2(\boldsymbol{y}^t))} = \frac{\mathrm{Max}P_0}{\mathrm{Min}P_0}$. Based on duality, we have $\frac{\mathrm{Max}P_0}{\mathrm{Min}P_0} \le \frac{P_1}{P_2} \le \frac{D_1}{D_2}$, where $P_1$ is the problems which maximizes $C_0$ with (1d) (1g) (1i), and $\sum_k x_{ki} \le Q_i, \forall i$, $P_2$ is minimization problem with the same constraints as $P_1$, and $D_1$ and $D_2$ are their dual problems. We introduce the dual variables $\bar{\lambda}_k, \bar{\delta}_i, \bar{\epsilon}_{kij}, \bar{\phi}_{kim}$ and $\widetilde{\lambda}_k, \widetilde{\delta}_i, \widetilde{\epsilon}_{kij}, \widetilde{\phi}_{kim}$ for $P_1$ and $P_2$, respectively. By choosing $\bar{\lambda}_k = a_{ki} + \frac{\sum_j d_{ij}}{K} + \frac{\sum_m d_{mi}}{K}, \bar{\delta}_i = 0, \bar{\epsilon}_{kij} = \frac{-d_{ij}}{K}, \bar{\phi}_{kim} = \frac{-d_{mi}}{K}$ and $\widetilde{\lambda}_k = -a_{ik}, \widetilde{\delta}_i = \widetilde{\epsilon}_{kij} = \widetilde{\phi}_{kim} = 0$, we obtain that $\frac{D_1}{D_2} \le \frac{\max_i\{\sum_k a_{ki} + \sum_j d_{ij} + \sum_m d_{mi}\}}{\min_i \sum_k a_{ki}}$, and define the ratio $\sigma$ as $\max_t \frac{\max_i\{\sum_k a_{ki} + \sum_j d_{ij} + \sum_m d_{mi}\}}{\min_i \sum_k a_{ki}}$.

Based on the above, $\sum_t H_{-M}$ can be bounded as follows, $\sum_t H_{-M} = \sum_t \mathbb{Y}_{SC} + \sum_t \mathbb{Y}_{-SC} \le (1+\frac{1}{\rho_1})\sum_t \mathbb{Y}_{-SC} \le (1+\frac{1}{\rho_1})(1+\frac{1}{\rho_2}+D_{max})\sum_t \mathbb{X}_{-SC} \le (1+\frac{1}{\rho_1})(1+\frac{1}{\rho_2}+D_{max})\sigma\sum_t \min_{\boldsymbol{y}^t} \mathbb{X}_{-SC}(Alg_2(\boldsymbol{y}^t)) \le r_3 \cdot \sum_t H_{-M}^*$, where $r_3 = (1+\frac{1}{\rho_1})(1+\frac{1}{\rho_2}+D_{max})\sigma r_1$. According to Theorem 2, we can obtain $\sum_t H_M \le \Gamma_{min}(\frac{2\ln(1/\theta)+2\ln(K+1)}{1-\theta}) = r_2 \cdot \Gamma_{min}$. Finally, we exhibit the competitive ratio $r$ as follows, $H_1 = \sum_t(H_M + H_{-M}) \le r_2\sum_t H_M^{t*} + r_3\sum_t H_{-M}^{t*} \le \max\{r_2, r_3\} H_1^*$. □

### D. Proof of Theorem 4

*Proof.* Let $\Delta P$ and $\Delta D$ denote the increment of the objective function in the problem (7) and (8), respectively, $\Delta D = -(\kappa_{\imath^*\jmath^*} + E_{i^*}\Delta\varsigma_{i^*})$, where $\Delta\varsigma_{i^*} = \frac{\varsigma_{i^*}}{E_i} + \frac{\widehat{A}_{\imath^*\jmath^*i^*} + d_{i^*}/3K + \sum_m d_{mi^*}/3K + \sum_{\imath m}\mathscr{d}_{\imath mi^*}/3K}{E_i\varpi}$ stands for the increment in $\varsigma_{i^*}$. Thus, we have $\Delta D = -(\kappa_{\imath^*\jmath^*} + E_{i^*}\Delta\varsigma_{i^*}) = -\kappa_{\imath^*\jmath^*} - E_i(\frac{\varsigma_{i^*}}{E_i} + \frac{\widehat{A}_{\imath^*\jmath^*i^*} + d_{i^*}/3K + \sum_m d_{mi^*}/3K + \sum_{\imath m}\mathscr{d}_{\imath mi^*}/3K}{E_i\varpi}) = (1-\frac{1}{\varpi})(\widehat{A}_{\imath^*\jmath^*i^*} + \frac{d_i^*}{3K} + \frac{\sum_m d_{mi^*}}{3K} + \frac{\sum_{\imath,m}\mathscr{d}_{\imath mi^*}}{3K}) = \frac{\varpi-1}{\varpi}\Delta P$. Let $P^k$ and $D^k$ denote the value of the objective function in the problem (7) and (8) after handling the offline or online classifier. We have $H_7 = P^{3K} = \sum_k(P^k - P^{k-1}) = \frac{\varpi}{\varpi-1}\sum_k(D^k - D^{k-1}) = \frac{\varpi}{\varpi-1}(D^{3K} - D^0) = \frac{\varpi}{\varpi-1}D^{3K} = H_8 \le \frac{\varpi}{\varpi-1}H_7^*$, followed by $P^0$ and $D^0$ initialized to 0; then, due to duality, we obtain $r_4 = \frac{\varpi}{\varpi-1}$. □

### E. Proof of Theorem 5

*Proof.* To simplify our proof, we introduce new symbols. We use $\mathscr{F}_{kn}$ to replace the strong classifier $\mathscr{F}_{kim}^t$, use $\zeta_{k,n}$ to replace $\zeta_{km}^t$, use $P_{k,n} = \frac{\zeta k,n}{\sum_k \zeta_{k,n}}$ to denote the normalized weight of the strong classifier, use $m_{k,n}$ to denote the mistakes of the strong classifier $k$, which are all updated as our Algorithm 8 shows, and use $q_n$ and $\hat{q}_n$ to denote the ground-truth label and the inferred label for the data sample $n$.

Firstly, we prove that $\mathbb{I}\{q_n \cdot \hat{q}_n < 0\} = \mathbb{I}\{\sum_k P_{k,n}m_{k,n} > 0.5\}$. By assuming that there are only $K_1$ classifiers predicting correctly, we have $\hat{q}_n = sign(q_n(\sum_{k=1}^{K_1} P_{k,n} - \sum_{k=K_1+1}^{K+1} P_{k,n}))$. Then, based on $\sum_k P_{k,n} = 1$, we obtain $q_n \cdot \hat{q}_n < 0 \iff \sum_{k=1}^{K_1} P_{k,n} - \sum_{k=K_1+1}^{K+1} P_{k,n} < 0 \iff \sum_{k=K_1+1}^{K+1} P_{k,n} > 0.5 \iff \sum_k P_{k,n}m_{k,n} > 0.5$.

By convexity, it can be shown $\alpha^r \le 1-(1-\alpha)r$ for $\alpha \ge 0$ and $r \in [0,1]$. We can get $\sum_{k=1}^K \zeta_{k,n+1} = \sum_{k=1}^K \zeta_{k,n}\beta_1^{m_{k,n}} \le \sum_{k=1}^K \zeta_{k,n}(1-(1-\beta_1)m_{k,n}) = (\sum_{k=1}^K \zeta_{k,n})(1-(1-\beta_1)\sum_k P_{k,n}m_{k,n})$. Applying repeatedly for $n = 0, ..., N$ data samples, we have $\sum_{k=1}^K \zeta_{k,N+1} \le \prod_{n=0}^N(1-(1-\beta_1)\sum_k P_{k,n}m_{k,n}) \le \exp(-(1-\beta_1)\sum_{k,n} P_{k,n}m_{k,n})$. Thus, we further get $\sum_{k,n} P_{k,n}m_{k,n} \le \frac{-\ln(\sum_k \zeta_{k,N+1})}{1-\beta_1}$. Note $\zeta_{k,N+1} = \zeta_{k,0}\prod_{n=1}^N \beta_1^{m_{k,n}} = \zeta_{k,0}\beta_1^{L_k}$, $\sum_{k=1}^K \zeta_{k,N+1} \ge \zeta_{k,N+1} = \zeta_{k,0}\beta_1^{L_k}$, where $L_k = \sum_n \mathbb{I}(sign(q_n^t\mathscr{F}_{kn}^t(p_n)) < 0)$. Then, we have

$\sum_{k,n} P_{k,n} m_{k,n} \leq \frac{-\ln \zeta_{k,0} - L_k \ln \beta_1}{1-\beta_1}, \forall k, \quad \sum_{k,n} P_{k,n} m_{k,n} \leq \frac{-\ln \zeta_{k,0} - L_{\min} \ln \beta_1}{1-\beta_1}$ where $L_{\min} = \min\{L_k\}$. Afterwards, we get $\sum_n \mathbb{I}\{q_n \cdot \hat{q}_n < 0\} \leq 2 \sum_{k,n} P_{k,n} m_{k,n} \leq 2 \frac{-\ln \zeta_{k,0} - L_{\min} \ln \beta_1}{1-\beta_1}$. It can be shown that $\ln(1/\beta) \leq (1-\beta^2)/2\beta$ for $\beta \in (0,1]$. We can obtain $\sum_{k,n} P_{k,n} m_{k,n} \leq \frac{-\ln \zeta_{k,0} - L_{\min} \ln \beta_1}{1-\beta_1} \leq \frac{L_{\min}(1+\beta_1)}{2\beta_1} - \frac{\ln \zeta_{k,0}}{1-\beta_1}$ We set $\beta_1 = \sqrt{L_{\min}}/(\sqrt{L_{\min}} + \sqrt{\ln K})$ and $\zeta_{k,0} = 1/K$. We can get $\frac{L_{\min}(1+\beta_1)}{2\beta_1} - \frac{\ln \zeta_{k,0}}{1-\beta_1} \leq L_{\min} + \frac{3}{2}\sqrt{\ln K L_{\min}} + \ln K$. We obtain that $\sum_n \mathbb{I}\{q_n \cdot \hat{q}_n < 0\} \leq 2 \sum_{k,n} P_{k,n} m_{k,n} \leq 2L_{\min} + 3\sqrt{\ln K L_{\min}} + 2 \ln K$.

We can also obtain $L_{\min} \leq 2(\frac{\Upsilon_{min}(1+\beta_2)}{2\beta_2} + \frac{\ln 3}{1-\beta_2})$, where $\Upsilon_{\min} = \min\{\Upsilon^k\}, \Upsilon^k = \min\{\Lambda_0^k, \Lambda_1^k, \Lambda_2^k\}, \Lambda_\iota^k = \sum_{t=1}^{T} \sum_{m=0}^{M^t} \mathbb{I}\{sign[q_m^t f_{\iota km}^t(p_{\iota km}^t)] < 0\}$ . The proof is similar to the preceding paragraph. We set $\beta_2 = \sqrt{\Upsilon_{\min}}/(\sqrt{\Upsilon_{\min}} + \sqrt{\ln 3})$ and $\psi_{\iota k0}^1 = 1/3, \iota \in \{0,1,2\}$. We can get $L_{\min} \leq 2\Upsilon_{\min} + 3\sqrt{\ln 3 \Upsilon_{\min}} + 2 \ln 3$ .

Finally, we can get $\sum_t H_{HM} \leq \frac{L_{\min}(1+\beta_1)}{2\beta_1} - \frac{\ln \zeta_{k,0}}{1-\beta_1} \leq L_{\min}\big(\frac{(1+\beta_1)}{2\beta_1} - \frac{\ln \zeta_{k,0}}{1-\beta_1}\big) \leq \Upsilon_{min}(4(\frac{1+\beta_2}{2\beta_2} + \frac{\ln 3}{1-\beta_2})(\frac{1+\beta_1}{2\beta_1} + \frac{\ln K}{1-\beta_1}))$. □

### *F. Proof of Theorem 6*

*Proof.* Based on Theorem 3, we can get $r_6 = (1 + \frac{1}{\rho_1})(1 + \frac{1}{\rho_2} + \mathcal{D}_{max}) r_4 \sigma_1$, where $\rho_1$ and $\rho_2$ are constants in Algorithms 6 and 7; $r_4$ is the approximation ratio of Algorithm 5, as in Theorem 4; $\mathcal{D}_{max} = \max\{\max_{\iota,i,k,t}\{\frac{B_i^t}{3A_{\iota ki}^t K}\}, 2\}$; and $\sigma_1 = \max_t\{\frac{\max_i\{\sum_{\iota,k} A_{\iota ki} + \sum_j d_{ij} + 2\sum_m d_{mi}\}}{\min_i \sum_{\iota,k} A_{\iota ki}}\}$. According to Theorem 5, we can obtain $\sum_t H_{HM} \leq \Upsilon_{min}(4(\frac{1+\beta_2}{2\beta_2} + \frac{\ln 3}{1-\beta_2})(\frac{1+\beta_1}{2\beta_1} + \frac{\ln K}{1-\beta_1})) \leq r_5 \cdot \Upsilon_{min}$. Finally, we exhibit the competitive ratio $\mathcal{R}$ as in $H_5 = \sum_t (H_{HM} + H_{-HM}) \leq r_5 \sum_t H_{HM}^{t*} + r_6 \sum_t H_{-HM}^{t*} \leq \max\{r_5, r_6\} H_5^*$. □