Offloading Algorithms for Maximizing Inference Accuracy on Edge Device in an Edge Intelligence System

Andrea Fresa and Jaya Prakash Champati
Edge Networks Group, IMDEA Networks Institute, Madrid, Spain
E-mail:{andrea.fresa, jaya.champati}@imdea.org

***Abstract*—With the emergence of edge computing, the problem of offloading jobs between an Edge Device (ED) and an Edge Server (ES) received significant attention in the past. Motivated by the fact that an increasing number of applications are using Machine Learning (ML) inference from the data samples collected at the EDs, we study the problem of offloading *inference jobs* by considering the following novel aspects: 1) in contrast to a typical computational job, the processing time of an inference job depends on the size of the ML model, and 2) recently proposed Deep Neural Networks (DNNs) for resource-constrained devices provide the choice of scaling down the model size by trading off the inference accuracy. Considering that multiple ML models are available at the ED, and a powerful ML model is available at the ES, we formulate an Integer Linear Programming (ILP) problem with the objective of maximizing the total inference accuracy of n data samples at the ED subject to a time constraint T on the makespan. Noting that the problem is NP-hard, we propose an approximation algorithm Accuracy Maximization using LP-Relaxation and Rounding (AMR^2) and prove that it results in a makespan at most $2T$ and achieves a total accuracy that is lower by a small constant from the optimal total accuracy implying that AMR^2 is asymptotically optimal. Further, if the data samples are identical we propose Accuracy Maximization using Dynamic Programming (AMDP), an optimal pseudo-polynomial time algorithm. Furthermore, we extend AMR^2 for the case of multiple ESs, where each ES is equipped with a powerful ML model. As proof of concept, we implemented AMR^2 on a Raspberry Pi, equipped with MobileNets, that is connected to a server equipped with ResNet, and studied the total accuracy and makespan performance of AMR^2 for image classification.**

I. Introduction

Edge computing is seen as a key component of future networks that augments the computation, memory, and battery limitations of Edge Devices (EDs) (e.g., IoT devices, mobile phones, etc.), by allowing the devices to offload computational jobs to nearby Edge Servers (ESs) [1]. Since the *offloading decision*, i.e., which jobs to offload, is the key to minimizing the execution delay of the jobs and/or the energy consumption at the ED, it received significant attention in the past [2]. Recently, an increasing number of applications are using Machine Learning (ML) inference from the data samples collected at the EDs, and there is a major thrust for deploying pre-trained Deep Neural Networks (DNNs) on the EDs as this has, among other advantages, reduced latency. Thanks to the development of DNN models with reduced computation and storage requirements, at the cost of reduced inference accuracy, and the advancements in the hardware of EDs [3],

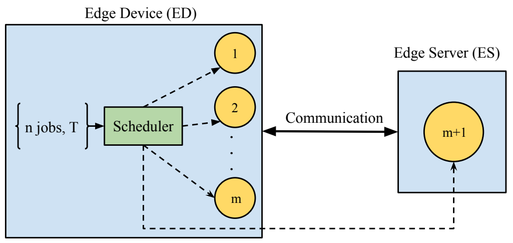


Fig. 1: Scheduling inference jobs between an ED and an ES.

ML frameworks such as Tensorflow Lite [4] and PyTorch Mobile [5] are now able to support the deployment of DNNs on EDs. In this context, we study the offloading decision between an ED and an ES for *inference jobs*, where an inference job refers to the execution of a pre-trained ML model on a data sample.

In comparison to the fixed processing time requirement of a generic computational job (typically represented by a directed acyclic task graph), the processing time requirement of an inference job depends on the ML model size: a larger model size results in a longer processing time and may provide higher inference accuracy. For example, on Pixel 3 smartphone, ResNet [6] has size 178 MB, requires 526 ms, and provides 76.8% accuracy (Top-1 accuracy) for the ImageNet dataset [7], while the smallest DNN model of MobileNet [8] has size 1.9 MB, requires 1.2 ms, but provides 41.4% accuracy [4]. Furthermore, recently developed DNNs for EDs allow for scaling the model size by simply setting a few hypeparameters (cf. [8]–[10]), enabling the EDs to choose between multiple model sizes. However, as we explain in Section II, the offloading decision for inference jobs considering the above novel aspects has received little attention in the literature.

Taking into account the novel aspects for inference jobs, or simply *jobs* in the sequel, we consider the system in Figure 1, where the ED has m ML models to choose from, and the ES is equipped with a state-of-the-art ML model for a given application. Consider that n jobs (corresponding to n data samples) are available at the ED. It may offload them all to the ES to maximize the inference accuracy. However, offloading each job incurs a communication time to upload the data sample in addition to the processing time at the ES. This may result in a large *makespan*, i.e., the total time to finish all the jobs. On the other hand, executing all the jobs on the smallest ML model at the ED may result not only in a smaller makespan, but also the lowest inference accuracy. Thus, a scheduler at the ED needs to strike a trade-off between the accuracy and the makespan. Toward this end, we formulate the following problem: *given n data samples at time zero, find a schedule that offloads a partition of the jobs to the ES and assigns the remaining jobs to m models on the ED, such that the total accuracy is maximized and the makespan is within a time constraint T*. Solution to this problem will be beneficial to applications such as Google Photos where a set of photos selected by a user need to be classified into multiple categories in real time. Also, the problem has relevance to applications which do periodic scheduling, i.e., the ED periodically collects all the data samples that arrived in a time period T and aims to finish their processing within the next time period T.

Since the true accuracy (top-1 accuracy) provided by a model for a given data sample can only be inferred after the job is executed, for analytical tractability, we consider the average test accuracy of the model is its accuracy for any data sample. Given the processing and communication times of the jobs, we formulate the problem as an Integer Linear Program (ILP). We note that the ILP is agnostic to the actual ML models used on the ED and the ES. Different ML models on the ED may correspond to different instantiations of the same DNN with different hyperparameter values (cf. [8]), or they may correspond to different ML algorithms such as linear regression, SVMs, DNNs, etc., although in this work we focus on the former setting.

Solving the formulated ILP is challenging due to the following reasons. Partitioning the set of jobs between the ED and the ES is related to scheduling jobs on parallel machines [11], and assigning the jobs to the models on the ED is related to the knapsack problem [12], both are known to be NP-hard. A special case of our problem, where the ED has a single model ($m = 1$), is the Generalized Assignment Problem (GAP) with two machines [13]. GAP is known to be APX-hard[1], and the best-known approximation algorithm provides a solution that has maskespan at most $2T$ [14]. However, the algorithms for GAP and their performance guarantees are not directly applicable to our problem due to the additional aspect that, on the ED there are multiple models to choose from. We propose a novel algorithm that solves a Linear Programming relaxation (LP-relaxation) of the ILP, uses a counting argument to bound the number of fractional solutions, and used a simple rule to round the fractional solution.

Our main contributions are summarized below[2]:

- We formulate the total accuracy maximization problem subject to a constraint T on the makespan as an ILP. Noting that the ILP is NP-hard, we propose an approximation algorithm Accuracy Maximization using LP-Relaxation and Rounding (AMR^2) to solve it. The runtime of AMR^2 is $O(n^3(m+1)^3)$.
- We prove that the total accuracy achieved by AMR^2 is at most a small constant (less than 1), lower than the optimum total accuracy, and its makespan is at most $2T$. We also extend AMR^2 for the case of K ESs and prove performance bounds. A salient feature of AMR^2 is that it is asymptotically optimal.
- For the case of identical jobs, i.e., the data samples are identical, we propose an optimal algorithm Accuracy Maximization using Dynamic Programming (AMDP), which does a greedy packing for the ES and solves a Cardinality Constrained Knapsack Problem (CCKP) for job assignments on the ED. We implement AMDP and demonstrate that its runtime is much lower compared to AMR^2.
- As proof of concept, we perform experiments using a Raspberry Pi, equipped with MobileNets, and a server, equipped with ResNet50, that are connected over a Local Area Network (LAN). The MobileNets and ResNet50 are trained for classifying images from the ImageNet dataset. We estimate processing and communication times for different sizes of images and implemented AMR^2 and a greedy algorithm, Greedy-RRA, on Raspberry Pi. Our results indicate that the total test accuracy achieved by AMR^2 is close to that of the LP-relaxed solution, and its total true accuracy is, on average, 40% higher than that of the greedy algorithm. We also implemented AMR^2 and Greedy-RRA for the case of K ESs. We find that as the number of ESs increases the total accuracy increases but the total accuracy gain provided by AMR^2 over Greedy-RRA decreases.

The rest of the paper is organized as follows. In Section II, we present the related work. The system model is presented in Section III. In Sections IV and V, we present AMR^2 and its performance bounds, respectively. In section VI we present a Dynamic Programming algorithm for a particular instance of our problem: when all jobs have identical processing time when inferred on an ML model. In section VII we present the extension of AMR^2 for the case of K ESs. In Section VIII, we present the experimental results and finally conclude in Section IX.

II. Related Works

In this section, we first present the related works for computation offloading problem and then discuss closely related classical job scheduling problems.

A. *Offloading and ML Inference Jobs*

Since the initial proposal of edge computing in [16], significant attention was given to the computational offloading problem, wherein the ED needs to decide which jobs to offload, and how to offload them to an ES [2]. The objectives that were considered for optimizing the offloading decision are, 1) minimize job execution times, see for example [17]–[20], 2) minimize the energy of the ED spent in computing and/or transmitting the jobs, subject to a constraint on the execution delay, see for example [21]–[23] and [24], and 3) decide scheduling strategy to guarantee statistical QoS for jobs where unreliable communication channels between EDs and ESs are considered, see for example [25]. However, the above

[1]An APx-hard problem has no polynomial-time approximation scheme unless P = NP.

[2]A part of this work was accepted to be published in ACM MSWIM, 2022 [15].

works consider generic computation jobs, and the aspect of accuracy, which is relevant for the case of inference jobs, has not been considered.

Recently, a few works considered the problem of maximizing accuracy for inference jobs on the ED [26] , [27], [28]. In [26], the authors studied the problem of maximizing the accuracy within a deadline for each frame of a video analytics application. They do not consider offloading to the edge and their solution is tailored to the DNNs that use early exits [9]. A similar problem was studied in [27], where offloading between a mobile device and a cloud is considered. The authors account for the time-varying communication times by using model selection at the cloud and by allowing the duplication of processing the job a the mobile device. A heuristic solution was proposed in [28] for offloading inference jobs for maximizing inference accuracy subject to a maximum energy constraint. In contrast to the above works, we consider multiple models on the ED and provide performance guarantees for AMR^2. The authors in [29] studied the minimization of the energy consumption on the EDs by offloading inference jobs to Cloudlets. In [30], the authors proposed a novel approach for reducing communication costs while offloading images. This work proposes the computation of a discrimination matrix P, which is computed on the Cloud, that is then used by the ESs. When an ED offloads an image X to an ES, the ES computes the product $P^T X$ that determines the features that are relevant to be offloaded to the cloud. This approach reduces the communication cost to offload the image to the cloud and also increases the accuracy of the inference result. In [31], the authors studied the properties of DNN networks in order to increase the parallelism in computation. Specifically, considering the independence of some operations in the same DNN layer, they proposed a novel technique which transforms the DNN into a Direct Acyclic Graph (DAG) DNN. With this technique, multiple operations can be executed in parallel, thus increasing the throughput of serving inference jobs on the DNN. In contrast to the above works, we study inference accuracy maximization on an ED by scheduling inference jobs between the ED and ESs with a constraint on the makespan.

B. Job Scheduling

As noted in Section I, our problem is related to the knapsack problem [12]. To see this, note that if it is not feasible to schedule on the ES and all jobs have to be assigned to the ED, then maximizing the total accuracy is equivalent to maximizing profit, and the constraint T is equivalent to the capacity of knapsack. In this case, our problem turns out to be a generalization of the CCKP [32]. Another special case of our problem, where the ED has only a single model, can be formulated as a GAP [13], [33], with two machines. In GAP, n jobs (or items) have to be assigned to r machines (or knapsacks). Each job-machine pair is characterized by two parameters: processing time and cost. The objective is to minimize the total cost subject to a time constraint T on the makespan. It is known that GAP is APX-hard [34].

In their seminal work [14], the authors proposed an algorithm for GAP that achieves minimum total cost and has makespan at most $2T$. Their method involves solving a sequence of LP feasibility problems, in order to tackle the processing times that are greater than T, and compute the minimum total cost using bisection search. Their algorithm can also be used for solving a related extension of GAP, where the cost of scheduling a job on a machine increases linearly with decrease in the processing time of the job. In comparison to this setting, the accuracies (equivalent to negative costs) are not linearly related to the processing times of the jobs and thus the proposed method in [14] is not directly applicable to the problem at hand. Our proposed algorithm AMR^2 is different from their method in that it does not require to solve LP feasibility problems and the use of bisection search. Further, we prove the performance bounds using a different analysis technique which is based on a counting argument for the LP-relaxation and solving a sub-problem of the ILP.

III. System Model

Consider an ED and an ES connected over a network and the ED enlists the help of the ES for computation offloading. At time zero, n inference jobs, each representing the processing requirement of a data sample on a pre-trained ML model, are available to a scheduler at the ED. Let j denote the job index and $J = \{1, 2, \ldots, n\}$ denote the set of job indices.

A. ML Models and Accuracy

The ED is equipped with m pre-trained ML models, or simply models. Note that these may correspond to m instantiations of the same DNN with different hyperparameter values resulting in different model sizes; see for example [8], [9], or the models may correspond to different ML algorithms such as logistic regression, support vector machines, DNN, etc. Since the ES is a computationally powerful machine, we consider that it is equipped with a state-of-the-art model. We note that our problem formulation and the solution are applicable to any family of ML models deployed on the ED and the ES. Also, later in Section VII, we study the multiple ESs scenario.

Let $a_i \in [0, 1]$ denote the top-1 accuracy of model i. Since we do not know if a job is classified correctly by a model without first processing it on that model, for analytical tractability, we consider that the accuracy of a model i for any job is a_i. Note that assigning a job to a model with higher top-1 accuracy increases its probability of correct classification. WLOG, we assume that $a_1 \leq a_2 \leq \ldots \leq a_m$, and also assume that the model $m+1$ is a state-of-the-art model with a higher top-1 accuracy than the models on the ED, i.e., $a_m \leq a_{m+1}$. In the sequel, the term 'accuracy' refers to the top-1 accuracy, unless otherwise specified.

B. Processing and Communication Times

The processing time of job j on model $i \in M \backslash \{m+1\}$ is denoted by p_{ij}, and on model $m+1$ it is denoted by $p'_{(m+1)j}$. In several applications, the data samples may need pre-processing before they are input to the ML model. For example, in computer vision tasks, images require pre-processing and the time required for pre-processing varies with the size of the image [35]. In our experiments with the images from the ImageNet dataset, the pre-processing stage only involves reshaping the images to input to the DNN models. Let τ_{ij} denote the pre-processing time of job j on model i. We consider the pre-processing times are part of the processing times defined above.

Let c_j denote the communication time for offloading job j. It is determined by the data size of the job, i.e., the size of the data sample in bits, and the data rate of the connection between the ED and the ES. Given $p'_{(m+1)j}$ and c_j, the *total time* to process job j on the ES, denoted by $p_{(m+1)j}$, is given by $p_{(m+1)j} = c_j + p'_{(m+1)j}$. We deliberately use similar notation for the processing times p_{ij} on the ED and the total times $p_{(m+1)j}$ on the ES because it simplifies the expressions in the sequel. We consider that the communication times c_j are fixed and are known apriori. This is possible in the scenarios where the ED and the ES are connected in a LAN or in a private network with fixed bandwidth. In our experiments, the ED and the ES are connected via our institute's LAN, and the communication times have negligible variance. We also consider that the processing times of the jobs are known apriori and that they can be estimated from the historical job executions.

C. Optimization Problem

Given the set of jobs J at time zero, the *makespan* is defined as the time when the processing of the last job in J is complete. The objective of the scheduler at the ED is to assign the set of jobs J to the set of models M such that the total accuracy, denote by A, is maximized and the makespan is within the time constraint T. Note that a schedule involves the partitioning of the set J between the ED and the ES, and the constraint on the makespan implies that the completion time of all the jobs should be within T on both ED and ES. The above objective is relevant in applications where the ED periodically collects the data samples in a period T and aims to finish their processing within the next period. By choosing a small period T, a real-time application can aim for fast ML inference at reduced total accuracy.

Let x_{ij} denote a binary variable such that $x_{ij} = 1$, if the scheduler assigns job j to model i, and $x_{ij} = 0$, otherwise. Note that, if $x_{(m+1)j} = 1$, then job j is offloaded to the ES. Therefore, a schedule is determined by the matrix $\mathbf{x} = [x_{ij}]$. We impose the following constraints on $\mathbf{x}$:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} \le T \tag{1}$$

$$\sum_{j=1}^{n} p_{(m+1)j}x_{(m+1)j} \le T \tag{2}$$

$$\sum_{i=1}^{m+1} x_{ij} = 1, \forall j \in J \tag{3}$$

$$x_{i,j} \in \{0,1\}, \forall i \in M, \forall j \in J, \tag{4}$$

where constraints (1) and (2) ensure that the total processing times on the ED and the ES, respectively, are within T, and thus the makespan is within T. Constraints in (3) imply that each job is assigned to only one model and no job should be left unassigned, and (4) are integer constraints. We are interested in the following accuracy maximization problem $\mathcal{P}$:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \qquad & A = \sum_{i=1}^{m+1}\sum_{j=1}^{n} a_i x_{ij} \\ \text{subject to} \qquad & (1), (2), (3), \text{ and } (4). \end{aligned}$$

Note that $\mathcal{P}$ is an ILP. We will show later that a special case of $\mathcal{P}$ reduces to CCKP, which is NP-hard, and thus $\mathcal{P}$ is NP-hard. Let A^* denote the optimal total accuracy for $\mathcal{P}$.

IV. Accuracy Maximization using LP-Relaxation and Rounding (AMR^2)

In this section, we first present the LP-relaxation of $\mathcal{P}$ and a result that guides the design of AMR^2.

A. LP-Relaxation

Given $\mathcal{P}$, we proceed with solving the LP-relaxation of $\mathcal{P}$, where the integer constraints in (4) are replaced using the following non-negative constraints:

$$x_{ij} \ge 0, \forall i \in M \text{ and } \forall j \in J. \tag{5}$$

Note that, the constraints $x_{ij} \le 1$ are not required as this is ensured by the constraints in (3). Let the matrix $\bar{\mathbf{x}} = [\bar{x}_{ij}]$ and A^*_{LP} denote the schedule and the total accuracy, respectively, output by the LP-relaxation. The LP-relaxed solution provides an upper bound on the total accuracy achieved by an optimal schedule, and thus we have $A^*_{\text{LP}} \ge A^*$.

Note that the solution to the LP-relaxation may contain x_{ij} values that are fractional, and the rounding procedure is critical to proving the performance bounds. To design a rounding procedure, we first refer to a key result in [36], where the author studied the problem of assigning N jobs to K parallel machines with the objective of minimizing the makespan. For the LP-relaxation of this problem, the author presented the following counting argument: there exists an optimal basic solution in which there can be at most $K-1$ *fractional jobs*, i.e., the jobs that are divided between machines, and all the other jobs are fully assigned. Further, the simplex algorithm outputs such a basic optimal solution. In our problem, there are two parallel machines, the ED and the ES, but in contrast to [36], the ED has multiple models and the jobs assigned to the ED are processed in sequence. Taking this new aspect into account, we extend the counting argument for the problem at hand and show that solving the LP-relaxation of $\mathcal{P}$ results in at most two fractional jobs. This structural result is stated in the following lemma.

Lemma 1. *For the LP-relaxation of $\mathcal{P}$, there exists an optimal basic solution with at most two fractional jobs.*

Proof. Since LP-relaxation of $\mathcal{P}$ has $n+2$ constraints, apart from the non-negative constraints in (5), one can show using LP theory that there exists an optimal basic solution with $n+2$ basic variables that may take positive values and all the non-basic variables take value zero. Under such an optimal basic solution, for the n constraints in (3) to be satisfied, at least one positive basic variable should belong to each of those n

constraints. The remaining 2 basic variables may belong to at most two equations. This implies that at least $n-2$ equations should have exactly one positive basic variable whose value should be 1 in order to satisfy the constraint. Therefore, there can be at most two equations with multiple basic variables whose values are in $(0,1)$, and the two jobs that correspond to these equations are the fractional jobs. □

Given the basic optimal solution to the LP-relaxation, the result in Lemma 1 reduces the rounding procedure to assigning at most two fractional jobs. WLOG, we re-index the jobs and refer to the fractional jobs by job 1 and job 2. We define the set $I = J\backslash\{1,2\}$ and refer to the assignment of I under $\bar{\mathbf{x}}$ as the *integer solution of the LP-relaxation*. We define:

$$P_1 = \sum_{i=1}^{m}\sum_{j\in I} p_{i,j}\bar{x}_{i,j}, \tag{6}$$

$$P_2 = \sum_{j\in I} p_{m+1,j}\bar{x}_{m+1,j}. \tag{7}$$

With a slight abuse in notation, we use i_j and k_j to denote the indices of the machines on which the fractional job $j \in \{1,2\}$ is scheduled. We have

$$\bar{x}_{i_1} + \bar{x}_{k_1} = 1, \tag{8}$$

$$\bar{x}_{i_2} + \bar{x}_{k_2} = 1. \tag{9}$$

Algorithm 1: AMR^2

```
1: Input: $p_{ij}$, for all $i \in M$ and $j \in J$.
2: Solve the LP-relaxation of $\mathcal{P}$.
3: if One fractional job then
4:    if $P_2 + p_{m+1,1} \leq 2T$ then
5:       Assign job 1 to model $m+1$
6:    else
7:       Assign job 1 to model
         $\arg\max_{i\in M\backslash\{m+1\}}\{a_i : p_{i1} + P_1 \leq 2T\}$.
8:    end if
9: end if
10: if Two fractional jobs then
11:    for all $j \in \{1,2\}$ do
12:       if $\bar{x}_{i_j} > \bar{x}_{k_j}$ then
13:          $x^\dagger_{i_j} = 1$
14:       else
15:          $x^\dagger_{k_j} = 1$
16:       end if
17:    end for
18: end if
19: Output: Assignment matrix $\mathbf{x}^\dagger$ and total accuracy $A^\dagger$
```

B. AMR^2 Description

The main steps of AMR^2 are summarized in Algorithm 1. In the first step, AMR^2 solves the LP-relaxation. In the second step, if there is one fractional job, it is assigned to model with largest accuracy such that the makespan does not exceed $2T$. If there are two fractional jobs, we use the simple rounding rule and assign the job to the model on which it has a higher fraction. Though the algorithm is not sophisticated, we will later see that proving the performance bounds is involved. We use $\mathbf{x}^\dagger$ and $A^\dagger$ to denote the schedule and the total accuracy, respectively, output by AMR^2.

Computational complexity: The computational complexity of solving an LP with l variables is $O(l^3)$ (cf. [37]). In the LP-relaxation, the number of variables are $n(m+1)$ and thus its runtime is $O(n^3(m+1)^3)$. The rounding technique has negligible complexity when compared to the complexity of solving the LP-relaxation. In conclusion, the runtime is $O(n^3(m+1)^3)$.

V. Analysis of AMR^2

In this section, we analyse AMR^2 and present a $2T$ bound for its makespan and show that its total accuracy is at most $a_{m+1} - a_1$ lower than the optimal accuracy.

Theorem 1. *If $\mathcal{P}$ is feasible, then the makespan of the system under AMR^2 is at most 2T.*

Proof. For the case of one fractional job, the result follows by the construct of AMR^2. For the case of two fractional jobs, based on the fractional job assignment output by the LP-relaxation solution we consider three cases and for each case, we consider sub-cases based on the schedule of AMR2. For the proof, we assume (10) and (11) are true. The proof steps are similar for other cases.

$$\bar{x}_{k_1} < \bar{x}_{i_1}, \tag{10}$$

$$\bar{x}_{i_2} < \bar{x}_{k_2}. \tag{11}$$

Case 1: Both jobs are assigned as fractional on the ED, i.e., i_1 and k_2 are in $\{1,2,\ldots,m\}$. Clearly, in this case the makespan on the ES is at most T, same as that given in the LP-relaxed solution. Suppose that after rounding, the completion time on the ED under AMR^2 violates $2T$, i.e.,

$$P_1 + p_{i_1} + p_{k_2} > 2T. \tag{12}$$

From the LP-relaxed solution, we obtain

$$T - P_1 = p_{i_1}\bar{x}_{i_1} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2} + p_{k_2}\bar{x}_{k_2}. \tag{13}$$

Substituting (13) in (12), we obtain

$$p_{i_1} + p_{k_2} > T + p_{i_1}\bar{x}_{i_1} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2} + p_{k_2}\bar{x}_{k_2}. \tag{14}$$

Using (8) and (9) in (14), we obtain

$$p_{i_1}\bar{x}_{k_1} + p_{k_2}\bar{x}_{i_2} - p_{k_1}\bar{x}_{k_1} - p_{i_2}\bar{x}_{i_2} > T. \tag{15}$$

The inequality in (15) implies that

$$p_{i_1}\bar{x}_{k_1} + p_{k_2}\bar{x}_{i_2} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2} > T. \tag{16}$$

Given (10) and (11), (16) should hold if we substitute $\bar{x}_{i_1}$ in place of $\bar{x}_{k_1}$ and $\bar{x}_{k_2}$ in place of $\bar{x}_{i_2}$, i.e.,

$$p_{i_1}\bar{x}_{i_1} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2} + p_{k_2}\bar{x}_{k_2} > T. \tag{17}$$

However, the left hand side (LHS) of (17) is equal to $T - P_1$ (cf. (13)), which is smaller than T. Therefore, (15) is false and by contradiction $P_1 + p_{i_1} + p_{k_2} \leq 2T$ is true.

Case 2: One job is assigned as fractional between the ED and the ES and the other job is assigned as fractional between two models on the ED. WLOG, we consider job 1 is assigned to models on the ED and job 2 is assigned between the ED and the ES. We consider the following sub-cases.

Case 2a: Job 2 is scheduled on the ES and from (11) we must have $k_2 = m+1$. From the LP-relaxed solution, we have

$$P_1 + p_{i_1}\bar{x}_{i_1} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2} \leq T, \tag{18}$$

$$P_2 + p_{k_2}\bar{x}_{k_2} \leq T. \tag{19}$$

For the ES, consider that

$$P_2 + p_{k_2} > 2T. \tag{20}$$

Because of (11), $p_{k_2}\bar{x}_{k_2} > T$ is true. If $p_{k_2}\bar{x}_{k_2} > T$ then also $P_2+p_{k_2}\bar{x}_{k_2} > T$, which contradicts (19), and thus $P_2+p_{k_2} \leq 2T$.

For the ED consider that

$$P_1 + p_{i_1} > 2T. \tag{21}$$

Substituting (18) in (21), we obtain:

$$p_{i_1} > T + p_{i_1}\bar{x}_{i_1} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2}.$$

Using (9) we arrive to

$$p_{i_1}\bar{x}_{k_1} - p_{k_1}\bar{x}_{k_1} - p_{i_2}\bar{x}_{i_2} > T. \tag{22}$$

We consider (18) and (22), we have

$$P_1 + p_{i_1}\bar{x}_{i_1} + p_{k_1}\bar{x}_{k_1} + p_{i_2}\bar{x}_{i_2} < p_{i_1}\bar{x}_{k_1} - p_{k_1}\bar{x}_{k_1} - p_{i_2}\bar{x}_{i_2}$$
$$\implies P_1 + p_{i_1}(\bar{x}_{i_1} - \bar{x}_{k_1}) + 2p_{k_1}\bar{x}_{k_1} + 2p_{i_2}\bar{x}_{i_2} < 0,$$

which is false as all quantities on LHS are positive. This means that (22) is false, thus (21) too.

Case 2b: Job 2 is scheduled on the ED. When using the basic solution of the LP problem, the completion time of the ED is

$$P_1 + p_{i_1}\bar{x}_{i_1} + p_{k_2}\bar{x}_{k_2} + p_{k_1}\bar{x}_{k_1} \leq T. \tag{23}$$

We claim that $P_1 + p_{i_1} + p_{k_2} \leq 2T$.

Consider

$$P_1 + p_{i_1} + p_{k_2} > 2T. \tag{24}$$

Substituting (23) in (24), and using (8) and (9), we obtain

$$p_{i_1}\bar{x}_{k_1} + p_{k_2}\bar{x}_{i_2} - p_{k_1}\bar{x}_{k_1} > T. \tag{25}$$

Substituting (23) in (25), we obtain

$$-P_1 - p_{i_1}(\bar{x}_{k_1} - \bar{x}_{i_1}) + p_{k_2}(\bar{x}_{i_2} - \bar{x}_{k_2}) \geq 0.$$

The LHS is the sum of negative terms because of (10) and (11), thus the inequality is false.

Case 3: Both jobs are assigned as fractional between ED and ES by the LP-relaxation. We have three different sub cases based on the fractional assignation of the LP-relaxation problem.

Case 3a: Both jobs are scheduled on the ES. The completion time of the ES, when considering the basic solution $\bar{x}$ is:

$$P_2 + p_{i_1}\bar{x}_{i_1} + p_{k_2}\bar{x}_{k_2} \leq T \tag{26}$$

We claim that $P_2 + p_{i_1} + p_{k_2} \leq 2T$. Suppose that

$$P_2 + p_{i_1} + p_{k_2} > 2T. \tag{27}$$

We substitute (26) in (27) and obtain

$$p_{i_1}\bar{x}_{k_1} + p_{k_2}\bar{x}_{i_2} > T. \tag{28}$$

Substituting (26) in (28) we have

$$-P_2 - p_{i_1}(\bar{x}_{i_1} - \bar{x}_{k_1}) - p_{k_2}(\bar{x}_{k_2} - \bar{x}_{i_2}) \geq 0$$

which is false because of hypotheses (10) and (11).

Case 3b: One job is scheduled on the ED and the other on the ES. The completion time of the ED under the LP-relaxed solution satisfies

$$P_1 + p_{i_1}\bar{x}_{i_1} + p_{i_2}\bar{x}_{i_2} \leq T, \tag{29}$$

while the completion on the ES satisfies:

$$P_2 + p_{k_2}\bar{x}_{k_2} + p_{k_1}\bar{x}_{k_1} \leq T. \tag{30}$$

We claim $P_1+p_{i_1} \leq 2T$, and $P_2+p_{k_2} \leq 2T$. Suppose that

$$P_1 + p_{i_1} > 2T. \tag{31}$$

We substitute (29) in (31) and obtain

$$p_{i_1}\bar{x}_{k_1} - p_{i_2}\bar{x}_{i_2} > T. \tag{32}$$

Using (32) and (29) we have:

$$P_1 + p_{i_1}\bar{x}_{i_1} + p_{i_2}\bar{x}_{i_2} < p_{i_1}\bar{x}_{k_1} - p_{i_2}\bar{x}_{i_2}$$
$$\implies P_1 + p_{i_1}(\bar{x}_{i_1} - \bar{x}_{k_1}) + 2p_{i_2}\bar{x}_{i_2} < 0,$$

which is false, thus (31) is not true.

On the ES, suppose that

$$P_2 + p_{k_2} > 2T. \tag{33}$$

Substituting (30) in (33) we obtain

$$p_{k_2}\bar{x}_{i_2} - p_{k_1}\bar{x}_{k_1} > T. \tag{34}$$

Using (34) and (30) we have

$$P_2 + p_{k_2}\bar{x}_{k_2} + p_{k_1}\bar{x}_{k_1} < p_{k_2}\bar{x}_{i_2} - p_{k_1}\bar{x}_{k_1}$$
$$\implies P_2 + p_{k_2}(\bar{x}_{k_2} - \bar{x}_{i_2}) + 2p_{k_1}\bar{x}_{k_1} < 0$$

which is false, thus (33) is not true.

Case 3c: Both jobs are scheduled on the ED. We claim

$$P_1 + p_{i_1} + p_{k_2} \leq 2T. \tag{35}$$

The completion time equation on the ED using the basic solution $\bar{x}$ is

$$P_1 + p_{i_1}\bar{x}_{i_1} + p_{k_2}\bar{x}_{k_2} \leq T. \tag{36}$$

We negate (35):

$$P_1 + p_{i_1} + p_{i_2} > 2T. \tag{37}$$

We substitute (36) in (37) and obtain

$$p_{i_1}\bar{x}_{k_1} + p_{k_2}\bar{x}_{i_2} > T. \tag{38}$$

Substituting (36) in (38), we obtain

$$-P_1 - p_{i_1}(\bar{x}_{i_1} - \bar{x}_{k_1}) - p_{k_2}(\bar{x}_{k_2} - \bar{x}_{i_2}) \geq 0,$$

which is false because of hypotheses (10) and (11). □

Theorem 2. *The difference between the optimal total accuracy A^* and $A^\dagger$, the total accuracy achieved by AMR^2, is upper bounded by $a_{m+1} - a_1$.*

Proof. Since $A^* \le A^*_{LP}$, we prove the result with respect to A^*_{LP}. WLOG, we consider that i_1 and i_2 as the indices of the models with lower accuracy, respectively, for jobs 1 and 2, and k_1 and k_2 are the index of the models which provide higher accuracy. To prove the performance bound we distinguish the following three cases.

Case 1: $\bar{x}_{k_1} \ge \frac{1}{2}, \bar{x}_{k_2} \ge \frac{1}{2}$. In this case, AMR^2 will schedule job 1 on model k_1 and job 2 on model k_2. The contribution of the following jobs to the $A^\dagger$ is $a_{k_1} + a_{k_2}$. The contribution of the same jobs to the optimal solution A^*_{LP} is: $a_{i_1}\bar{x}_{i_1} + a_{k_1}\bar{x}_{k_1} + a_{i_2}\bar{x}_{i_2} + a_{k_2}\bar{x}_{k_2}$. However, $a_{k_1} > a_{i_1}$ and $a_{k_2} > a_{i_2}$: thus it is trivial that $A^\dagger > A^*$.

Case 2: $\bar{x}_{k_1} \ge \frac{1}{2}$ and $\bar{x}_{k_2} < \frac{1}{2}$. Here, AMR^2 schedules job 1 on k_1 and job 2 on i_2.

$$\begin{aligned} A^*_{LP} - A^\dagger &= a_{i_1}\bar{x}_{i_1} + a_{k_1}\bar{x}_{k_1} + a_{i_2}\bar{x}_{i_2} + a_{k_2}\bar{x}_{k_2} - a_{i_2} - a_{k_1} \\ &= a_{i_1}\bar{x}_{i_1} + a_{i_2}(\bar{x}_{i_2} - 1) + a_{k_1}(\bar{x}_{k_1} - 1) + a_{k_2}\bar{x}_{k_2} \\ &= a_{i_1}\bar{x}_{i_1} - a_{i_2}\bar{x}_{k_2} - a_{k_1}\bar{x}_{i_1} + a_{k_2}\bar{x}_{k_2} \\ &= \bar{x}_{i_1}(a_{i_1} - a_{k_1}) + \bar{x}_{k_2}(a_{k_2} - a_{i_2}). \end{aligned}$$

Substituting $\bar{x}_{k_1} \ge \frac{1}{2}$ and $\bar{x}_{k_2} < \frac{1}{2}$ in the above equation we have $a_{i_1} - a_{k_1} < 0$. In conclusion

$$A^*_{LP} - A^\dagger \le \frac{1}{2}(a_{k_2} - a_{i_2}). \tag{39}$$

Proof for the case $\bar{x}_{k_1} < \frac{1}{2}$, $\bar{x}_{k_2} \ge \frac{1}{2}$ is similar to **Case 2**.

Case 3: $\bar{x}_{k_1} < \frac{1}{2}$ and $\bar{x}_{k_2} < \frac{1}{2}$. AMR^2 schedules job 1 on i_1, and job 2 on i_2.

$$\begin{aligned} A^*_{LP} - A^\dagger &= a_{i_1}\bar{x}_{i_1} + a_{k_1}\bar{x}_{k_1} + a_{i_2}\bar{x}_{i_2} + a_{k_2}\bar{x}_{k_2} - a_{i_1} - a_{i_2} \\ &= \bar{x}_{k_1}(a_{k_1} - a_{i_1}) + \bar{x}_{k_2}(a_{k_2} - a_{i_2}) \le a_{m+1} - a_1. \end{aligned}$$

In the last equation above, we used $\bar{x}_{k_1} < \frac{1}{2}$ and $\bar{x}_{k_2} < \frac{1}{2}$, and the fact that $a_{k_1} - a_{i_1}$ and $a_{k_2} - a_{i_2}$ are upper bounded by $a_{m+1} - a_1$. □

From Theorem 2, the accuracy ratio between AMR^2 and the optimal schedule is bounded as follows:

$$\frac{A^\dagger}{A^*} \le 1 + \frac{a_{m+1} - a_1}{A^*}.$$

Note that $a_{m+1} - a_1 < 1$ and A^* grows as $O(n)$, where n is the number of jobs. To see the latter, note that $A^* \ge n a_1$ since the total accuracy for n jobs cannot be lower than $n a_1$. Thus, $\frac{A^\dagger}{A^*}$ goes to 1 as n goes to infinity. Thus, AMR^2 is asymptotically optimal.

Corollary 1. *If the processing times of all jobs on the ES are at most T, then $A^* \le A^\dagger$.*

Proof. Considering AMR^2 when all jobs have processing time less than T on all the models, we consider three cases. In the first case, there is no job assigned as fractional in the LP-relaxed solution which implies that we obtain $A^* = A^*_{LP} = A^\dagger$. In the second case, there is a single fractional job and we have $A^*_{LP} \le A^*_{LP,I} + a_{m+1}$. In this case, AMR^2 schedules the fractional job on the ES and we obtain $A^\dagger = A^*_{LP,I} + a_{m+1}$. Thus, $A^*_{LP} \le A^\dagger$. In the third case, the number of fractional jobs is two. AMR^2 schedules or either both jobs on the ES or one on the ES and the other on model m of the ED, achieving a total accuracy that is at least $A^\dagger = A^*_{LP,I} + a_{m+1} + a_m$. The solution of AMR^2 will be always greater or at most equal to the solution of A^*_{LP}, as it will have T seconds to schedule the two jobs, meanwhile, the LP-relaxation will have a time that is less or equal to T to schedule both of them. In all the three cases $A^\dagger \ge A^*_{LP} \ge A^*$. □

Remark 1: The schedule $\mathbf{x}^\dagger$ given by AMR^2 may result in a makespan greater than T. In our experimental results, we show that the percentage of violation on an average is at most 40% for the considered application. As noted before, a special case of our problem is GAP for which the best-known approximation algorithm, proposed in [14], has the makespan bound $2T$ and produces a schedule that may exceed T. The algorithm in [14] achieves the optimal cost for GAP. However, it requires a bisection search to find this optimal cost and each step in the bisection search requires solving an LP-relaxed feasibility problem. In contrast, in AMR^2 we solve an LP-relaxed problem only once and thus it has a lower computational complexity, which is important because, as we will see later in Section VIII, computing the schedule itself cannot take significant time when T is small.

VI. Identical Jobs

In this section, we consider the problem $\mathcal{P}_\text{I}$, a special case of $\mathcal{P}$ where the jobs are identical, i.e., $p_{ij} = p_i$, for all models $i \in M$. We present Accuracy Maximization using Dynamic Programming (AMDP) for $\mathcal{P}_\text{I}$. The formulation for $\mathcal{P}_\text{I}$ is given below.

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \sum_{i=1}^{m+1} a_i \sum_{j=1}^{n} x_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{m} p_i \sum_{j=1}^{n} x_{ij} \le T && (40) \\ & p_{m+1} \sum_{j \in 1}^{n} x_{(m+1)j} \le T && (41) \\ & \sum_{i=1}^{m+1} x_{ij} = 1, \quad \forall j \in J && (42) \\ & x_{ij} \in \{0,1\}, \quad \forall i \in M, \forall j \in J. && (43) \end{aligned}$$

Next, we exploit the structure of $\mathcal{P}_\text{I}$ and reduce it to solving a Cardinality Constrained Knapsack Problem (CCKP).

A. CCKP

Our first observation is that the number of jobs assigned to the ES under an optimal schedule is given by $n_c = \left\lfloor \frac{T}{p_{m+1}} \right\rfloor$. To see this, assigning number of jobs less than n_c can only reduce the accuracy as the ES provides highest accuracy, and no more than n_c can be assigned due to constraint (41). We present this observation in the following lemma.

Lemma 2. *Under an optimal schedule, the number of jobs assigned to the ES is given by* $n_c = \left\lfloor \frac{T}{p_{m+1}} \right\rfloor$.

We define $n_l = n - n_c$. Since the jobs are identical, without loss of generality, we assign the last n_c jobs to the ES. We are now only required to compute the optimal assignment for jobs $j \in \{1, \ldots, n_l\}$ to the models 1 to m on the edge device. Thus, given Lemma 2, solving $\mathcal{P}_{\text{I}}$ is reduced to solving the following problem $\mathcal{P}_{\text{I}}{}'$:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \sum_{i=1}^{m} a_i \sum_{j=1}^{n_l} x_{ij} \\ \text{subject to} \quad & \sum_{i=1}^{m} p_i \sum_{j=1}^{n_l} x_{ij} \leq T, && (44) \\ & \sum_{i=1}^{m} x_{ij} = 1, \quad \forall j \in \{1, \ldots, n_l\} && (45) \\ & x_{ij} \in \{0,1\}, \, \forall i \in M \backslash \{m+1\}, \forall j \in \{1, \ldots, n_l\}. \end{aligned}$$

We do a variable change to formulate the CCKP. Let r denote an index taking values from $\{1, \ldots, mn_l\}$. We define new variables z_r, accuracies $\bar{a}_r$, and processing times $\bar{p}_r$ as follows: for $i \in M \backslash \{m+1\}$ and $j \in \{1, \ldots, n_l\}$,

$$\begin{aligned} z_r &= \{x_{ij} : r = j + (i-1)n_l\}, \\ \bar{a}_r &= a_i, \, (i-1)n_l \leq r < in_l, \\ \bar{p}_r &= p_i, \, (i-1)n_l \leq r < in_l. \end{aligned}$$

The CCKP using $\{z_r : 1 \leq r \leq mn_l\}$ as the decision variables is stated below.

$$\begin{aligned} \underset{\{z_r\}}{\text{maximize}} \quad & \sum_{i=1}^{mn_l} \bar{a}_r z_r \\ \text{subject to} \quad & \sum_{r=1}^{mn_l} \bar{p}_r z_r \leq T, && (46) \\ & \sum_{r=1}^{mn_l} z_r = n_l, && (47) \\ & z_r \in \{0,1\}, \, \forall r \in \{1, \ldots, mn_l\}. && (48) \end{aligned}$$

Let $\{z_r^* : 1 \leq r \leq mn_l\}$ denote an optimal solution for CCKP. The CCKP can be interpreted as follows. We have mn_l items, where each item represents a model and there are n_l copies of the same model. Since the jobs are identical, the problem reduces to the number of times a model is selected, equivalent to the number of jobs assigned to it, such that all jobs are assigned. In the following lemma we state that solving CCKP results in an optimal solution for $\mathcal{P}_{\text{I}}{}'$.

Lemma 3. *The solution* $x_{ij}^* = \{z_r^* : r = j + (i-1)n_l\}$ *is an optimal solution for* $\mathcal{P}_I{}'$.

Proof. By construction, $\mathcal{P}_{\text{I}}{}'$ and CCKP have one-to-one mapping between the decision variables, have equivalent objective functions and constraints in (44) and (46). They only differ in the constraints (45) and (47). We note that (47) is equivalent to

$$\sum_{j=1}^{n_l} \sum_{i=1}^{m} x_{ij} = n_l. \tag{49}$$

Let $\mathcal{P}_{\text{I}}{}^{\ddagger}$ denote the problem $\mathcal{P}_{\text{I}}{}'$ with the constraint (45) replaced by (49). From the above observations, $\mathcal{P}_{\text{I}}{}^{\ddagger}$ is equivalent to CCKP, and thus it is sufficient to show that an optimal solution $\{x_{ij}^{\ddagger}\}$ for $\mathcal{P}_{\text{I}}{}^{\ddagger}$ is optimal for $\mathcal{P}_{\text{I}}{}'$. Since (49) is a relaxation of the constraint in (45), the optimal objective value of $\mathcal{P}_{\text{I}}{}^{\ddagger}$ should be at least the optimal objective value of $\mathcal{P}_{\text{I}}{}'$. On the other hand, given $\{x_{ij}^{\ddagger}\}$, consider the assignment where for each model i, we assign $\sum_{j=1}^{n_l} x_{ij}^{\ddagger}$ jobs to it. Given that the jobs are identical, and from (49), all the n_l jobs will be assigned exactly once to some model. Thus, this assignment is feasible for $\mathcal{P}_{\text{I}}{}'$ and objective value under this assignment will be equal to the optimal objective value of $\mathcal{P}_{\text{I}}{}^{\ddagger}$. Thus, $\{x_{ij}^{\ddagger}\}$ is also an optimal solution for $\mathcal{P}_{\text{I}}{}'$. □

In Algorithm 2 we present AMDP for solving $\mathcal{P}_{\text{I}}$. The optimality of AMDP is a direct consequence of Lemmas 2 and 3 and is stated in the following theorem.

Theorem 3. *AMDP is an optimal algorithm for* $\mathcal{P}_I$.

Algorithm 2: AMDP

```
1: $n_l = n - \left\lfloor \frac{T}{p_{m+1}} \right\rfloor$
2: Assign the jobs $j \in \{n_l + 1, \ldots, n\}$ to the ES
3: Solve the CCKP for $\{z_r^*\}$ using the DP algorithm
4: Assignment for remaining jobs:
   $x_{ij}^* = \{z_r^* : r = j + (i-1)n_l\}$ for all $i \in M \backslash \{m+1\}$,
   and $j \in \{1, \ldots, n_l\}$.
```

B. The DP Algorithm

The main step in AMDP is to solve the CCKP for which one can leverage existing branch-and-bound or Dynamic Programming (DP) algorithms [12]. We use the DP algorithm since it has pseudo-polynomial runtime for computing the optimal solution. The summarize the main steps of the algorithm. Let s, k, and τ denote positive integers. We define $y_s(\tau, k)$ as the maximum accuracy that can be achieved by selecting items from the set $\{1, \ldots, s\}$, where $s \leq mn_l$, given a time constraint τ $(\leq T)$ and the number of items to be selected are k $(\leq n_l)$.

$$y_s(\tau, k) = \max \left\{ \sum_{r=1}^{s} \bar{a}_r z_r \middle| \sum_{r=1}^{s} \bar{p}_r z_r \leq \tau, \sum_{r=1}^{j} z_r = k, z_r \in \{0,1\} \right\} \tag{50}$$

The DP iterations are given below:

$$y_s(\tau, k) = \begin{cases} y_{s-1}(\tau, k) & \text{if } \bar{p}_s \geq \tau \\ \max\{y_s(\tau - \bar{p}_s, k-1) + \bar{a}_s, y_{s-1}(\tau, k)\} & \text{otherwise.} \end{cases}$$

We compute the solution for $y_s(T, n_l)$, where $s = mn_l$.

The computational complexity of the DP algorithm is $O(mnT)$ and AMDP has the same computational complexity.

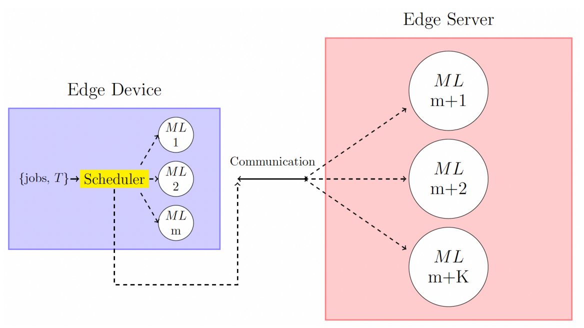


Fig. 2: Scheduling between an Edge Device and K Edge Servers.

VII. Multiple Edge Servers

In this section, we extend AMR^2 for the case of multiple ESs. As before, the ED is equipped with m models and it is connected through multiple communication links to K ESs which are physically collocated. Each ES is equipped with a powerful ML model and on average they give more accurate inferences compared with those executed in the ML models on the ED. The system is shown in Fig. 2. The motivation for this system comes from the scenario where there is a single server which hosts multiple virtual machines each equipped with a powerful ML model or a factory floor where for reliability reasons there are multiple servers.

Inference jobs can be executed locally on one of the m ML models, or remotely on one of the K ESs. Again, our objective is to maximize the total inference accuracy subject to a constraint T on makespan. Let the index set for the ML models is $M' = \{1,..,m,m+1,\dots,m+K\}$, where $m+1$ to $m+K$ denote the models on the ESs. We formulate the following constraints:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} \leq T \tag{51}$$

$$\sum_{j=1}^{n} p_{ij}x_{ij} \leq T, \, \forall i = \{m+1,...,m+K\} \tag{52}$$

$$\sum_{i=1}^{m+K} x_{ij} = 1, \, \forall j \in J \tag{53}$$

$$x_{ij} \in \{0,1\}, \, \forall i \in M', \forall j \in J, \tag{54}$$

where constraints (51) and (52) ensure that the total processing times on the ED and all the ESs, respectively, are within T, and thus the makespan is within T. Constraints in (53) imply that each job is assigned to only one model and no job should be left unassigned, and (54) imposes the integer constraints. We are interested in the following accuracy maximization problem $\mathcal{P}_K$:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & A = \sum_{i=1}^{m+K}\sum_{j=1}^{n} a_i x_{ij} \\ \text{subject to} \quad & (51), (52), (53), \text{ and } (54). \end{aligned}$$

Clearly, $\mathcal{P}_K$ is an extension of $\mathcal{P}$ and is thus NP-hard. With a slight abuse in the notation, we use A^* to denote the optimal total accuracy of $\mathcal{P}_K$.

We solve $\mathcal{P}_K$ by first formulating the LP-relaxation by relaxing the constraints in (54) as follows:

$$x_{ij} \geq 0, \, \forall i \in M', \forall j \in J. \tag{55}$$

In the following lemma we state that, in the case of K ESs, the number of fractional jobs in the LP-relaxed solution can be at most $K+1$.

Lemma 4. *For the LP-relaxation of $\mathcal{P}_K$, there exists an optimal basic solution with at most $K+1$ fractional jobs.*

Proof. Since the LP-relaxation of $\mathcal{P}_K$ has $n+K+1$ constraints apart from the non-negative constraint in (55), there exists an optimal basic solution with $n+K+1$ basic variables, where all the basic variables take positive value and all the non-basic variables take value zero. In this basic solution, for the n constraints in (53) to be satisfied, at least one positive basic variable should belong to each of those n constraints. The remaining $K+1$ basic variables may belong to at most $K+1$ equations. This implies that at most $n-K-1$ equations should have exactly one positive basic variable whose value should be 1 in order to satisfy the constraint. Remaining $K+1$ jobs will have multiple basic variables that takes value in the interval $(0,1)$. Thus, at most $K+1$ jobs in the LP-relaxed solution of $\mathcal{P}_K$ are fractional. □

We re-index the fractional jobs such that they are assigned indices from $F = \{1,2,\dots,K+1\}$. We find the assignment for F. We use the same notation presented in the case of a single ES: given a generic job $j \in F$ we refer to i and k as the model which job j is divided. The details of the algorithm are presented in Algorithm 3.

Algorithm 3: AMR^2 for K ESs

```
 1: Input: $p_{ij}$, for all $i \in M'$ and $j \in J$.
 2: Solve the LP-relaxation of $\mathcal{P}_K$.
 3: The fractional job are collected in $F$.
 4: for all $j \in F$ do
 5:     if $\bar{x}_{ij} > \bar{x}_{kj}$ then
 6:         $x^{\dagger}_{ij} = 1$
 7:     else
 8:         $x^{\dagger}_{kj} = 1$
 9:     end if
10: end for
11: Output: Assignment matrix $\mathbf{x}^{\dagger}$ and total accuracy $A^{\dagger}$
```

Next, we present the performance analysis for AMR^2, presented in Algorithm 3, for solving $\mathcal{P}_k$.

Theorem 4. *For the case of K edge servers, AMR^2 provides a makespan of at most $2T$.*

Proof. In the case of K ESs, enumerating all possible cases for different fractional jobs scheduled between the models of the ED and the ESs and proving the $2T$ bound is tedious because as K increases the number of cases increases exponentially.

Nevertheless, the idea behind the proof for each case is similar to as that in Theorem 1 and therefore, we present the proof step for one case. The proof for all other cases follows using similar steps. In this proof, we refer to the set of jobs to be scheduled $\{1, \dots, N\}$ as J.

Suppose that $\bar{x}_{(m+K)j} \geq \frac{1}{2}, \forall j \in F$, this means that we are in the case where AMR^2 schedules all fractional jobs on model $m+K$. The completion time of the ED, when considering only the jobs that are assigned as integer in the LP solution, is:

$$P_1 = \sum_{i=1}^{m} \sum_{j \in J} p_{ij} \bar{x}_{ij},$$

meanwhile, on each of the K ESs is:

$$\begin{aligned} P_K &= \sum_{j \in J} p_{(m+K)j} \bar{x}_{(m+K)j} \\ &= \sum_{j \in J \setminus F} p_{(m+K)j} \bar{x}_{(m+K)j} + \sum_{j \in F} p_{(m+K)j} \bar{x}_{(m+K)j} \leq T. \end{aligned} \tag{56}$$

We want to prove that

$$\sum_{j \in J \setminus F} p_{(m+k)j} \bar{x}_{(m+k)j} + \sum_{j \in F} p_{(m+k)j} \leq 2T. \tag{57}$$

Assume (57) is false, i.e.,:

$$\sum_{j \in J \setminus F} p_{(m+k)j} \bar{x}_{(m+k)j} + \sum_{j \in F} p_{(m+k)j} > 2T. \tag{58}$$

We use (56) in (58) and obtain:

$$\begin{aligned} &- \sum_{j \in F} p_{(m+K)j} \bar{x}_{(m+K)j} + \sum_{j \in F} p_{(m+K)j} > T \\ \implies &\sum_{j \in F} p_{(m+K)j} (-\bar{x}_{(m+K)j} + 1) > T \\ \implies &\sum_{j \in F} p_{(m+K)j} (-\bar{x}_{(m+K)j} + 1) > T. \end{aligned} \tag{59}$$

Recalling $\bar{x}_{(m+K)j} \geq \frac{1}{2}$, $1 - \bar{x}_{(m+K)j} < \frac{1}{2}$, for all j. Because of this, (59) is false, as LHS of (56) is larger than LHS of (59), and this means that (57) is true. ☐

Theorem 5. *Assuming a_{m+K} and a_1 are the highest and the lowest accuracy, respectively, the total accuracy achieved by an optimal schedule is at most $(K+1)\frac{(a_{m+K} - a_1)}{2}$ higher than the total accuracy achieved by the extended AMR^2 (Algorithm 3), i.e., $A^* - A^\dagger \leq (K+1)\frac{(a_{m+K} - a_1)}{2}$.*

Proof. Given the solution $\bar{\mathrm{x}}$ from the LP-relaxation of $\mathcal{P}_K$ we divide the fractional decision variables in:

$$b_{ij} = \begin{cases} \bar{x}_{ij} & \text{if } \bar{x}_{ij} \leq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}$$

$$c_{ij} = \begin{cases} \bar{x}_{ij} & \text{if } \frac{1}{2} < \bar{x}_{ij} < 1 \\ 0 & \text{otherwise.} \end{cases}$$

Then, $A^*_{LP} - A^\dagger$ is given by the followig expression:

$$\sum_{i=1}^{m+K} \sum_{j \in F} a_i b_{ij} + \sum_{i=1}^{m+K} \sum_{j \in F} a_i c_{ij} - \sum_{i=1}^{m+K} \sum_{j \in F} a_i x^\dagger_{ij}. \tag{60}$$

The expression in (60) takes maximum value for the problem instance, which we call worst-case problem instance, where $b_{ij} = b_{(m+K)j}$ and $c_{ij} = c_{1j}$, for all $j \in F$. Therefore, we obtain

$$A^*_{LP} - A^\dagger \leq a_{m+K} \sum_{j \in F} b_{(m+K)j} + a_1 \sum_{j \in F} c_{1j} - \sum_{j \in F} a_1, \tag{61}$$

and, to be precise, the RHS of (61) is maximized when $c_{1j} = \frac{1}{2} + \epsilon$, for all $j \in F$, where $\epsilon > 0$ approaches zero. To see this, note that in the worst-case problem instance, a fractional job j^* contributes to A^*_{LP} the value $a_1 c_{1j} + a_{m+K} b_{(m+K)j}$, while in $A^\dagger$ it contributes a_1. Therefore, the accuracy difference contributed by j^* is given by

$$a_1 c_{1j^*} + a_{m+K} b_{(m+K)j^*} - a_1. \tag{62}$$

Now consider a modified problem instance where the assignment is the same as the worst-case problem instance for all jobs $j \in F$, except for job j^* for which c_{2j^*} is equal to c_{1j^*} of the worst-case problem instance. For this modified instance, AMR^2 assign j^* to model 2 with accuracy a_2, and the accuracy difference contributed by j^* is given by

$$a_2 c_{2j^*} + a_{m+K} b_{(m+K)j^*} - a_2. \tag{63}$$

Clearly, the difference in (62) is bigger than that in (63), as $c_{1j^*} = c_{2j^*} < 1$ and $a_2 > a_1$. Using similar arguments, it can be verified that (61) is the upper bound for $A^*_{LP} - A^\dagger$ over all problem instances. Therefore, we have

$$\begin{aligned} A^*_{LP} - A^\dagger &\leq a_{m+K} \sum_{j \in F} b_{(m+K)j} + a_1 \sum_{j \in F} c_{1j} - \sum_{j \in F} a_1 \\ &= a_{m+K} \sum_{j \in F} b_{(m+K)j} + a_1 \sum_{j \in F} (1 - b_{(m+K)j}) - \sum_{j \in F} a_1 \\ &= (a_{m+K} - a_1) \sum_{j \in F} b_{(m+K)j} \\ &\leq \frac{K+1}{2} (a_{m+K} - a_1). \end{aligned} \tag{64}$$

In the last step above, we have used $b_{(m+K)j} \leq \frac{1}{2}$, for all $j \in F$, and that, according to Lemma 4, the number of fractional jobs from the LP solution can be at most $K+1$. ☐

From Theorem 5, we have

$$\frac{A^\dagger}{A^*} \leq 1 + \frac{(K+1)(a_{m+K} - a_1)}{2A^*}.$$

Noting that K is a constant, $a_{m+K} - a_1 < 1$, and A^* grows as $O(n)$, we see that AMR^2 is asymptotically optimal for the case of K ESs.

VIII. Experimental Results

In this section, we first present the experimental setup. We then present the implementation details for estimating the processing and communication times. As explained in Section II, the aspect of multiple models on the ED has not been considered in computation offloading literature and there are no existing algorithms that are applicable for the problem at hand for a performance comparison. Therefore, we present the performance comparison between AMR^2 and a baseline

Greedy Round Robin Algorithm (Greedy-RRA). Given the list of jobs, Greedy-RRA offloads them from the start of the list to the ES until the constraint T is met. The remaining jobs are assigned in a round robin fashion to the models on the ED until the constraint T is met. Any further remaining jobs are assigned to model 1. Note that Greedy-RRA solution may violate the time constraint T and its runtime is $O(n)$. Finally, we also demonstrate the performance of the extended AMR^2 for the case of multiple ESs.

For the experimental setup, we chose image classification as the ML application due to its prevalence. Nevertheless, we emphasize that our system model and the proposed algorithm apply to other ML applications.

A. Experimental Setup

Our experimental setup comprises a Raspberry Pi device (the ED) and a local server (the ES) that are connected and located in the same LAN. Raspberry Pi has 4 cores, 1.5 GHz CPU frequency, and 4 GB RAM, with the operating system Raspbian 10, while the server has 512 cores, 1.4 GHz CPU frequency, and 504 GB RAM, with the operating system Debian 11. All the functions on Raspberry Pi and on the server are implemented using Python 3. We used HTTP protocol to offload images from Raspberry Pi to the ES and implemented HTTP Client and Server using Requests and Flask, respectively. The LP-relaxation problem of P is solved using PuLP library on the ED. The data samples are images from the ImageNet dataset for which we use DNN models for inference. On Raspberry Pi we import, from the TensorFlow Lite library, two pre-trained MobileNets corresponding to two values 0.25 and 0.75 for the hyperparameter α, which is a width multiplier for the DNN [38]. Both the models are quantized and require input images of dimensions 128×128. Quantization describes the process of reducing the precision of the weights. Thus, it is reduced the size of the DNN model. On the ES, we import a pre-trained ResNet50 model [6] from the Tensorflow library. The ResNet50 model requires input images of dimensions 224×224. Images of different dimensions need to be reshaped to the respective dimensions on the ES and the ED. The top-1 accuracies for the three models are presented in Table I.

Model	**Top-1 Accuracy**
MobileNet $\alpha = 0.25$ (model 1)	0.395
MobileNet $\alpha = 0.75$ (model 2)	0.559
ResNet50 (model 3)	0.771

TABLE I: Test accuracies of the considered DNN models [4].

We implemented both AMR^2 and Greedy-RRA on Raspberry Pi in Python 3. AMR^2 takes up to 50 ms for computing a schedule for 40 jobs. The runtime of AMR^2 is dominated by the runtime of the solver from the Python library for solving the LP-relaxation. In future, we plan to reduce this runtime by implementing AMR^2 in C.

B. Estimation of Processing and Communication Times

In our experiments, we consider images of dimensions 333×500, 375×500, and 480×640, for which we estimate the processing and communication times using the following procedure. On Raspberry Pi, we run 30 samples of same image dimensions and use the median processing times as our estimate. Note that median is an unbiased estimate, and unlike the mean, it is not affected by cold start. We note that the estimates for the processing times include the reshape times.

In order to estimate the total time on the ES, we use the HTTP client/server connection to send 30 images of same image dimensions from Raspberry Pi to the server. For each image we measure the time till the reception of an inference for the image from the ES, and finally use the median. At the server we also measure the reshape time and the processing time, and the estimate for the communication time is obtained by subtracting the reshape time and the processing time from the total time. Since Raspberry Pi and the dedicated local server are in the same LAN, the observed communication times are almost constant with negligible variance. This is also true for the observed processing times, and we will later verify this when implementing the schedules using these estimates.

The estimates for the processing times are presented in Table II. Observe that the processing times increase with the model size. On Raspberry Pi, the variance in the processing times on a model is small. In contrast, the total times on the ES vary with the dimensions of the image and are an order of magnitude higher than the processing times on Raspberry Pi. In Figure 3, we present the communication, reshape, and processing times on the ES. It is worth noting that, as the dimensions of the image increases, both communication time and the reshape times increase. Thus, it is more advantageous to offload images with smaller dimensions. The bandwidth of the communication of the system has been studied using iPerf3 with 30 experiments of 1 minutes each. Note that the measured bandwidth varies based on the chosen Tranport layer. We use HTTP to send images from ED to ES. Thus, we measured TCP bandwidth. The average bandwidth is 987 Mbit/s. However, using Requests and Flask we achieved an average bandwidth of 144 Mbit/s. The most important think that we realized is that the bandwidth is constant within the experiments. Given as input a JPEG image of 1024×1024 to the network, we experienced an average communication time of 0.0007 sec with a variance of 2.88×10^{-5} sec.

Model	**Location**	333×500	375×500	480×640
MobileNet $\alpha = 0.25$	ED	0.01	0.011	0.011
MobileNet $\alpha = 0.75$	ED	0.04	0.04	0.043
ResNet50	ES	0.28	0.32	0.38

TABLE II: Estimated processing times (in seconds).

C. Performance of AMR^2

In Figure 4, we examine the number of jobs assigned to different models under AMR^2. Observe that as T increases the number of jobs assigned to larger models increases. Also, note that MobileNet $\alpha = 0.25$ is only being used when T is small. In all the subsequent figures, for each point, we run 30 experiments and compute the average. Recall that the total accuracy $A^\dagger$ is based on the top-1 accuracy of the models.

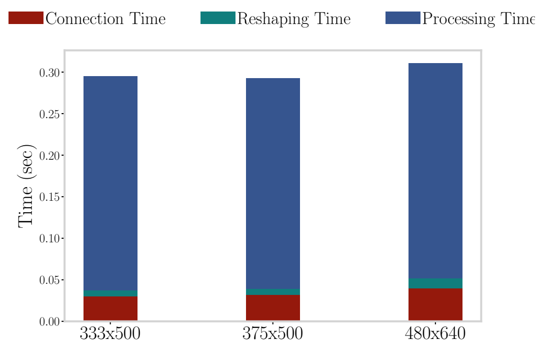


Fig. 3: Estimated total time for inference on the ES.

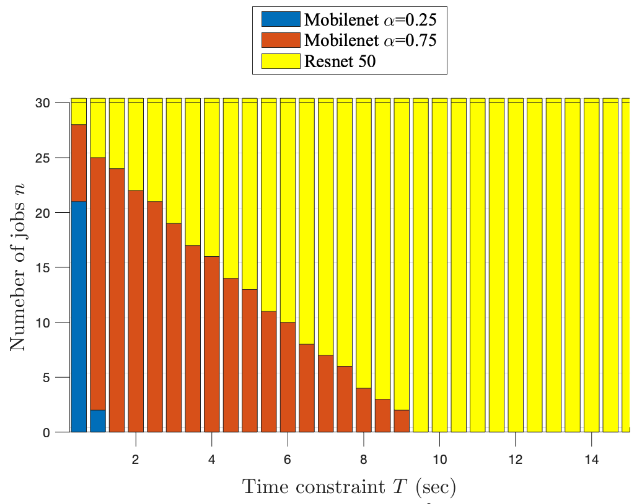


Fig. 4: Job assignment under AMR^2 for varying T.

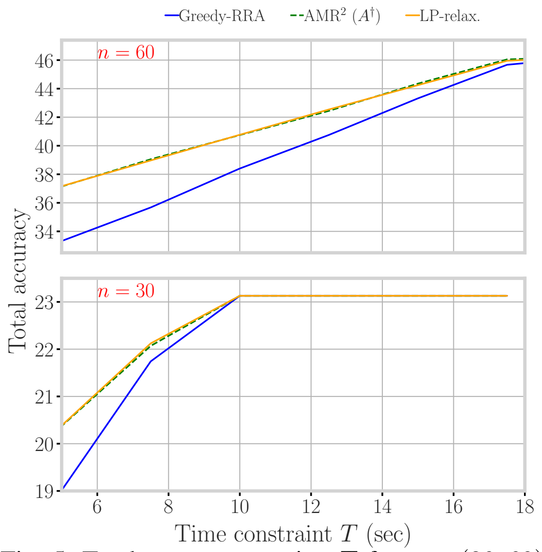


Fig. 5: Total accuracy varying T for $n = (30, 60)$.

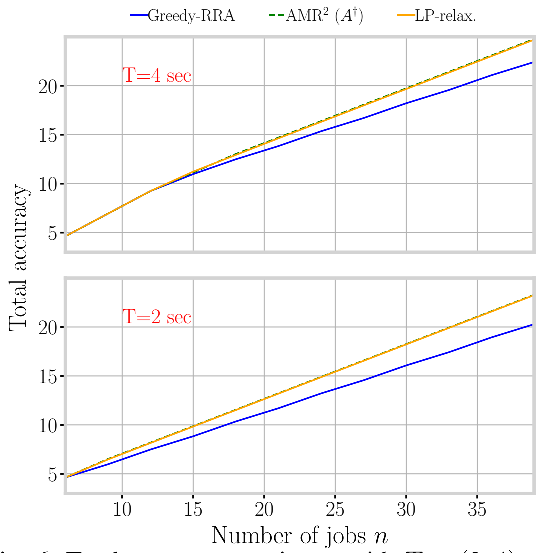


Fig. 6: Total accuracy varying n with $T = (2, 4)$ sec.

In Figures 5 and 6, we compare total accuracy achieved under different schedules, by varying T and n, respectively. For $n = 60$, no LP-relaxed solution exists for $T = 2$ sec. From both figures, we observe that $A^\dagger$ overlaps with, and in some cases exceeds, the total accuracy of the LP-relaxed solution A^*_{LP}. This is because all the processing times (cf. Table II) are less than 2 sec, the minimum value used for T, and therefore, from Corollary 1, $A^\dagger$ exceeds A^*. Furthermore, in some cases, where T is large enough, AMR^2 may assign both the fractional jobs to the server and $A^\dagger$ exceeds A^*_{LP}. In the above cases, however, the makespan under AMR^2 exceeds T.

From Figure 5, we observe that AMR^2 always has higher total accuracy than Greedy-RRA with a percentage gain between 20–60% averaging at 40%, but the percentage gains are lower at smaller T. The latter fact is also confirmed in Figure 6 when $T = 2$ sec. This is expected, because for $T = 2$ sec, not many jobs can be offloaded to the server as the processing times are around 0.3 seconds. For $T = 4$ sec we see significant gains of around 40–50%.

In Figure 7, we present the makespan achieved by AMR^2 and Greedy-RRA for varying n. The actual makespan, i.e., the time elapsed at Raspberry Pi from the start of scheduling the jobs till the finishing time of the last job is indicated by AMR^2 in the legend. The estimated makespan that is numerically computed using the schedule $\mathbf{x}^\dagger$ and the estimated processing and communication times is indicated by AMR^2 (estd. proc. time). Observe that both these makespans have negligible difference asserting that the variances in our estimates for both communication and processing times are small. For $T = 4$, AMR^2 violates T for $n \geq 17$, but then it saturates at a makespan with a maximum percentage of violation of 15%. This is expected because from Lemma 1 there cannot be more than two fractional jobs irrespective of n value and thus, the constraint violation due to the reassignment of the fractional jobs do not increase beyond $n = 30$. This saturation effect can also be observed for $T = 2$. In this case, the percentage of violation under AMR^2 is higher because the processing times on the server are comparable to $T = 2$ sec and reassigning a fractional job to the server results in a higher percentage of violation.

D. Experimental results for multiple ESs

As a proof of concept, we study the performances of AMR^2 for the case of multiple ESs. We use the same experimental testbed as before. On the server, we create multiple containers. On each container, we implemented an HTTP server that hosts a DNN. To make containers independent of one from another

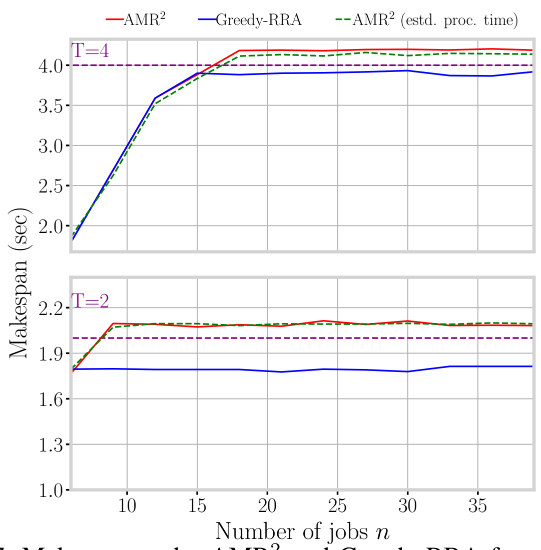


Fig. 7: Makespan under AMR^2 and Greedy-RRA for varying n, and $T = 2$ sec and $T = 4$ sec.

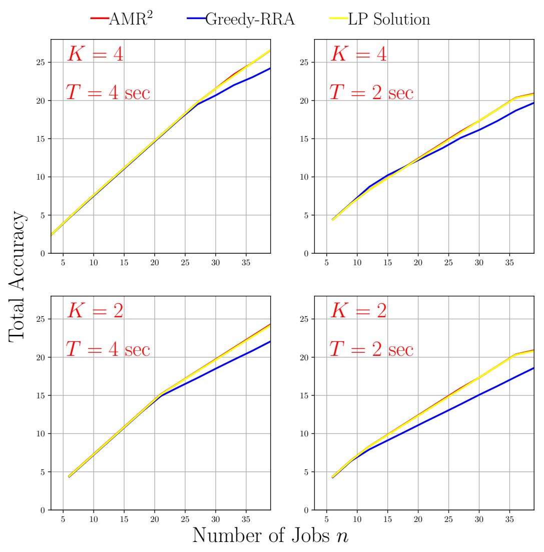


Fig. 8: Total Accuracy under AMR^2 and Greedy-RRA for varying n, and $T = 2$ sec and $T = 4$ sec. On the left in the case of 2 ESs, on the right with 4 ESs.

and from the processes of the Operative System on the physical machine, we limited the resources of containers to a maximum of 8 GB of RAM and 4 CPUs each. In Table IV, we present the DNN models from which we choose.

Model	**Average test Accuracy**
MobileNetV2	0.713
ResNet50	0.749
Xception	0.790
InceptionResNetV2	0.803

TABLE III: Test accuracy of the considered DNN models on the ESs [4].

Model	333×500	375×500	480×640
MobileNetV2	0.2050	0.2102	0.2101
ResNet50	0.87	0.90	0.92
Xception	1.79	1.79	1.80
InceptionResNetV2	2.51	2.51	2.53

TABLE IV: Processing time (sec) for image of dimensions 128×128, 512×512, 1024×1024 when using models from [39].

We ran experiments using 2 ESs (using MobileNetV2 and InceptionResNetV2) and 4 ESs (using MobileNetV2, ResNet50, Xcpetion, InceptionResNetV2). In Figure 8, we study the total accuracy for time constraints $T = 2$ sec and $T = 4$ sec. The two subplots on the left represents the experiments for $k = 2$, and the two subplots on the right represents the experiments for $k = 4$. Greedy-RRA works as described before doing a round-robin assignment by including k ESs. From Figure 8, we observe that as the time constraint increases, the difference between the total accuracy achieved by AMR^2 and Greedy-RRA decreases. This is expected because Greedy-RRA has more ESs to use for offloading images without exceeding the maximum completion time T on them. Also, as the number of edge servers increases the total accuracy increases. For example, for $T = 2.0$ sec, the total accuracy achieved by AMR^2 increases by 20% for $K = 4$ when compared to $K = 2$. Furthermore, observe that for $K = 2$ we achieve a total accuracy of 21 for 30 jobs, but for $K = 4$ we reach the same accuracy with 27 jobs. Thus, the average accuracy per job increases as k increases, we observe similar trends as in the case of $K = 1$ for increasing T. The figures are not presented due to redundancy.

E. Experimental results for AMDP

In this subsection, we present the results of AMDP using the same setup for the single ES described above, except for considering identical inference jobs, i.e., images with equal data size 480×640 pixels. The top-1 accuracies of the models are presented in Table I, and their processing times are presented in Tabel II. Recall that AMDP is a Dynamic Programming algorithm that guarantees optimal solution (Theorem 3). Its optimality is verified by comparing its solution with the solution given by an ILP Solver for $\mathcal{P}_{\text{I}}$. Also, we compare its runtime with the runtimes of the ILP solver and AMR^2. In Figure 9, we present this comparison by varying the number of jobs between 10 to 200. Observe from the upper sub-figure that AMDP reaches the same solution as the ILP solver. From the lower sub-figure, observe that the runtime of AMDP is much lower (by more than an order of magnitude) when compared to the runtime of the ILP solver, which increases approximately linearly (note that the figure is in log scale) with the number of jobs. Interestingly, we found that AMR^2 also provides an optimal solution for these problem instances as there are no fractional jobs to be rounded. However, the advantage of using AMDP is that it has a lower runtime (an order of magnitude less) than that AMR^2.

IX. Conclusion

We have studied the offloading decision for inference jobs between an ED and an ES, where the ED has m models and

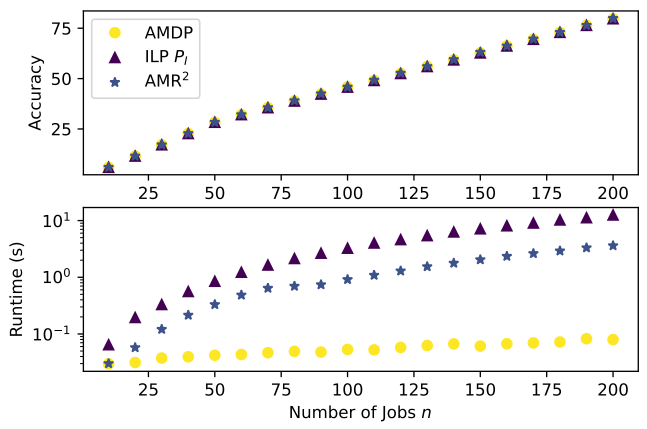


Fig. 9: Comparison between the solution of AMDP, an ILP solver which solves $\mathcal{P}_{\mathrm{I}}$, and AMR^2.

the ES has a state-of-the-art model. Given n data samples at the ED, we proposed an approximation algorithm AMR^2 for maximizing the total accuracy for the inference jobs subject to a time constraint T on the makespan. We proved that the makespan under AMR^2 is at most $2T$, and its total accuracy is lower than the optimal total accuracy by at most 2, and for typical problem instances its total accuracy is at least the optimal total accuracy. When the data samples are identical, we have proposed AMDP, a pseudo-polynomial time algorithm to compute the optimal schedule. We have implemented AMR^2 on Raspberry Pi and demonstrated its efficacy in improving the inference accuracy for classifying images within a time constraint T. Also, under the considered scenarios AMR^2 provides, on average, 40% higher total accuracy than that of Greedy-RRA. We also extended AMR^2 for the case of multiple edge servers and observed an increase of 20% in the total accuracy when the number of ESs increase from $k = 2$ to $k = 4$.

In our problem model, we considered that the communication times are deterministic and in our testbed we used an architecture where the ED and the ES are connected via Ethernet. If the ED and ES are connected over wireless channels where the communication times are random, AMR^2 can still be used by using the estimated mean communication times. The performance analysis of AMR^2 for this case is left for future work.

References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[2] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[3] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model compression and hardware acceleration for neural networks: A comprehensive survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020.

[4] "Image classification using TensorFlow Lite," https://www.tensorflow.org/lite/guide/hosted_models.

[5] "Pytorch mobile," https://pytorch.org/mobile/home/.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE CVPR*, 2016, pp. 770–778.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE CVPR*, 2009, pp. 248–255.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE CVPR*, 2018, pp. 4510–4520.

[9] S. Teerapittayanon, B. McDanel, and H. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *Proc. ICPR*, 2016, pp. 2464–2469.

[10] H. Cai, C. Gan, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," *CoRR*, vol. abs/1908.09791, 2019.

[11] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer Publishing Company, Incorporated, 2008.

[12] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*. Springer, Berlin, Germany, 2004.

[13] G. Ross and R. Soland, "A branch and bound algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 8, pp. 91–103, 1975.

[14] D. Shmoys and E. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical Programming*, no. 62, pp. 461–474, 1993.

[15] A. Fresa and J. P. Champati, "Offloading algorithms for maximizing inference accuracy on edge device in an edge intelligence system," in *Proc. ACM MSWIM (to appear)*, Oct 2022.

[16] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009.

[17] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. IEEE ISIT*, 2016, pp. 1451–1455.

[18] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3590–3605, 2016.

[19] J. P. Champati and B. Liang, "Semi-online algorithms for computational task offloading with communication delay," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1189–1201, 2017.

[20] ——, "Single restart with time stamps for parallel task processing with known and unknown processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 1, pp. 187–200, 2020.

[21] M.-H. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proc. IEEE SPAWC Worskhop*, 2015, pp. 186–190.

[22] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *Proc. IEEE ICC*, 2015, pp. 5529–5534.

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

[24] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 421–432, 2023.

[25] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "Qos driven task offloading with statistical guarantee in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 1, pp. 278–290, 2022.

[26] Z. Wang, W. Bao, D. Yuan, L. Ge, N. H. Tran, and A. Y. Zomaya, "See: Scheduling early exit for mobile dnn inference during service outage," in *in Proc. MSWIM*, 2019, p. 279–288.

[27] S. S. Ogden and T. Guo, "Mdinference: Balancing inference accuracy and latency for mobile applications," in *Proc. IEEE IC2E*, 2020, pp. 28–39.

[28] I. Nikoloska and N. Zlatanov, "Data selection scheme for energy efficient supervised learning at iot nodes," *IEEE Communications Letters*, vol. 25, no. 3, pp. 859–863, 2021.

[29] Z. Xu, L. Zhao, W. Liang, O. F. Rana, P. Zhou, Q. Xia, W. Xu, and G. Wu, "Energy-aware inference offloading for dnn-driven applications in mobile edge clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 4, pp. 799–814, 2021.

[30] S. Wang, C. Ding, N. Zhang, X. Liu, A. Zhou, J. Cao, and X. Shen, "A cloud-guided feature extraction approach for image retrieval in mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 292–305, 2021.

[31] H. Zhou, M. Li, N. Wang, G. Min, and J. Wu, "Accelerating deep learning inference via model parallelism and partial computation offloading,"

IEEE Transactions on Parallel and Distributed Systems, vol. 34, no. 2, pp. 475–488, 2023.

[32] K. Dudzinski, "On a cardinality constrained linear programming knapsack problem," *Operations Research Letters*, vol. 8, no. 4, pp. 215–218, 1989.

[33] D. G. Cattrysse and L. N. Van Wassenhove, "A survey of algorithms for the generalized assignment problem," *European Journal of Operational Research*, vol. 60, no. 3, pp. 260–272, 1992.

[34] C. Chekuri and S. Khanna, "A ptas for the multiple knapsack problem," in *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '00. USA: Society for Industrial and Applied Mathematics, 2000, p. 213–222.

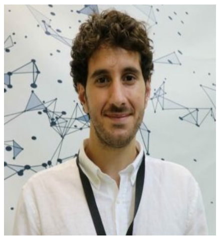
[35] P. Ross and A. Luckow, "Edgeinsight: Characterizing and modeling the performance of machine learning inference on the edge and cloud," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 1897–1906.

[36] C. Potts, "Analysis of a linear programming heuristic for scheduling unrelated parallel machines," *Discrete Applied Mathematics*, vol. 10, no. 2, pp. 155–164, 1985.

[37] J. van den Brand, "A deterministic linear program solver in current matrix multiplication time," in *Proc. ACM SODA*. Society for Industrial and Applied Mathematics, 2020, p. 259–278.

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[39] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.



Andrea Fresa is a PhD student at IMDEA Networks Institute and Universidad Carlos III, Madrid, Spain. He obtained his Bachelors and Masters degree in Computer Engineering at Universita' di Napoli Federico II, Naples, Italy. Prior to joining PhD, he was part of the IoT group in Ericsson Research, Helsinki, Finland, where he focused on the development of a cross-communication platform for heterogeneous IoT devices. His general research interest is in the design and analysis of algorithms for Edge Computing Systems.

Jaya Prakash Champati received his bachelor of technology degree from the National Institute of Technology Warangal, India in 2008, and master of technology degree from the Indian Institute of Technology (IIT) Bombay, India in 2010. He received his PhD in Electrical and Computer Engineering from the University of Toronto, Canada in 2017. From 2017-2020, he was a post-doctoral researcher with the division of Information Science and Engineering, EECS, KTH Royal Institute of Technology, Sweden. He is currently a Research Assistant Professor at IMDEA Networks Institute, Madrid, Spain where he leads the Edge Networks Group. His general research interest is in the design and analysis of algorithms for scheduling problems that arise in networking and information systems. Prior to joining PhD he worked at Broadcom Communications, where he was involved in developing the LTE MAC layer. He is a Marie Skłodowska-Curie Actions (MSCA) postdoctoral fellow, 2021, and a recipient of the best paper award at IEEE National Conference on Communications, India, 2011.