DOPart: Competitive-Ratio Optimal Online Offloading of Sequentially Dependent Tasks

Shiva Saxena and Ben Liang
Department of Electrical and Computer Engineering
University of Toronto, Canada

***Abstract*—This work is motivated by the need for real-time partitioning of a deep neural network inference job and offloading part of it from a resource-constrained device to a resource-rich server. Since both the processing delay of different job stages and the communication delay of offloading usually are unknown ahead of time, this problem is naturally modelled as online offloading of sequentially dependent computational tasks from a local processor to a remote processor. We aim to minimize the total delay of all tasks by utilizing local and remote computational resources, while considering the impact of the offloading delay. We propose Delay-optimal Online Partitioning (DOPart), a lightweight online algorithm that uses an adaptive thresholding strategy to solve the offloading problem. We derive the competitive ratio for DOPart and show that it is optimal in the sense that no other deterministic online algorithm can achieve a lower competitive ratio. Furthermore, it provides bounded performance guarantees even under imprecise parameter estimation. Through experimenting with AlexNet, VGG-19, and ResNet-34, we demonstrate that DOPart substantially outperforms state-of-the-art online solutions.**

I. Introduction

Computation offloading refers to uploading and executing computationally intensive parts of a computing job on a more powerful remote server. It is a common strategy to overcome the restrictions of resource-constrained mobile devices. For example, mobile cloud/edge computing facilitates computation offloading by providing access to the vast amount of resources hosted by cloud/edge servers. In order to fully exploit the advantages of computational offloading, we need to make intelligent offloading decisions and efficiently utilize the computational resources available in the network.

In this work, we are motivated by a recently emergent class of computation offloading problems concerning the partitioning of inference jobs with deep neural networks (DNNs) between a local mobile device and a remote server. The approximately layer-by-layer structure of an DNN (e.g., ResNet-34 shown in Fig. 1) provides an opportunity to partition an inference job between layers, process the earlier layers at the local device, and offload the later layers to the remote server. Clearly, the total delay in completing the job is reduced if and only if the speed-up due to faster processing at the remote server is greater than the communication delay incurred during offloading.

Since the computation resources on the local device is shared among multiple applications, the local processing delay is time varying and unknown in advance. Additionally, the offloading delay depends on the state of the communication channel, which also is time varying and unknown in advance especially in the wireless environment. This calls for an *online* algorithm that can dynamically adapt to the uncertainties in the computation and communication environment, in order to minimize the total job completion delay.

There is an extensive body of works in the literature on online partitioning and offloading of computing jobs comprising *independent* tasks (e.g., [1]-[3]). These works exploit the parallel processing of tasks to minimize the total delay. However, in DNNs and other similar applications, the jobs comprise a *dependent* sequence of tasks, or stages, where the execution of each stage requires the completion of the previous stage.

For online offloading of dependent tasks, as detailed further in Section II, most existing works assume the local processing delay and the offloading delay are revealed, possibly estimated by some profiler (e.g., [4] and [5]), when a task arrives. However, in many real-world systems, the available computing resource can vary during the processing of a task, so the processing delay may not be known until after the local execution of a task. Furthermore, reinforcement learning based online solutions (e.g., [6] and [7]) require system specific parameter tuning and generally do not provide performance guarantee. Instead, in this work we propose a lightweight online solution that does not require offline profiling, while still providing provable performance guarantees.

The main contributions of this work are as follows:

- We propose DOPart, an adaptive online algorithm for the partitioning and offloading of a computing job comprising sequentially dependent tasks. It does not require a priori knowledge of communication or local processing latencies. It uses only simple decision rules with low computational complexity, and yet it is highly effective.
- We show that DOPart is optimal in the sense that it achieves the best possible competitive ratio among all deterministic algorithms. In particular, let $\alpha_{\min}$ and $\alpha_{\max}$ be the minimum and maximum ratios between local and remote processing latencies, and consider the non-trivial scenario where $\alpha_{\max} > 1$. If $\alpha_{\min}\alpha_{\max} \geq 1$, DOPart is $\alpha_{\max}$-competitive, and if $\alpha_{\min}\alpha_{\max} < 1$, it is $\sqrt{\alpha_{\max}/\gamma}$ -competitive where γ is as defined in (15). We prove that these competitive ratios are tight and optimal because they meet the corresponding lower bounds. We also show

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada.

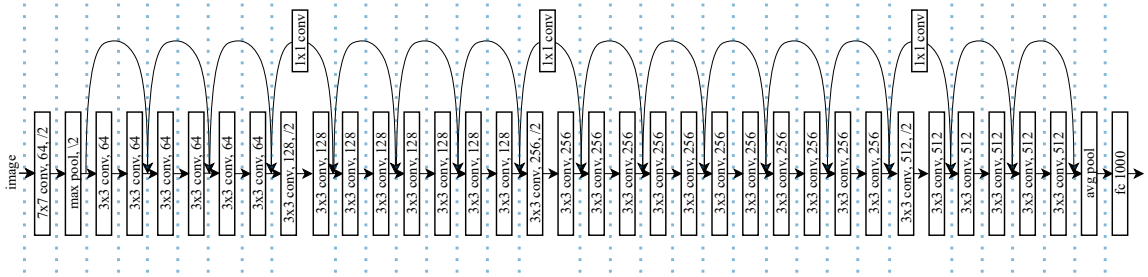

Fig. 1. Architecture of ResNet-34, with sequentially dependent stages indicated by blue dashed lines.

that DOPart offers bounded performance guarantees even when α_{min} and α_{max} are not precisely known.

- We experiment with online offloading of DNN inference jobs using AlexNet, VGG-19, and ResNet-34 on the ImageNet dataset over a simulated wireless channel to demonstrate the performance of DOPart and compare it against Neurosurgeon [4], Autodidactic Neurosurgeon [8], and other alternatives. Our numerical results indicate up to 17% reduction in inference time over these state-of-the-art methods.

II. Related Work

A. DNN Partitioning

In the seminar work on DNN partitioning [4], an algorithm named Neurosurgeon was proposed based on system level analysis of the partition points that minimizes the total delay. Subsequently, [9]-[11] considered resource constraints and minimized the total delay in an offline setting. In [5] and [12], offline communication and computation profilers are used to obtain the partitioning points for DNN inference. To reduce the overhead of profiling and parameter storage, [8] proposed a variant of LinUCB for DNN partitioning through online learning.

All of these works utilize some offline profiling or prediction to calculate the optimal partitioning point. The predicted model can become outdated quickly in a dynamic environment where the computation and communication resources fluctuate over time. In our work, we seek an online solution that can automatically adapt to the system dynamics, and we provide performance guarantee even when the future processing and communication latencies are unknown and time varying.

Furthermore, there are also interest studies on dynamic modification of the DNN architecture to minimize the inference delay [13]-[16]. All of these studies substantially differ from our work, since we assume no control over the DNN model's architecture or the job's structure in general, and we cannot compromise on the inference accuracy or the job's intended computational goal.

B. Online Dependent Task Offloading

There is rich literature on the online offloading of dependent tasks to reduce latency. Offloading in single-user single-server systems were considered in [17]-[19], all with general directed acyclic graphs for task dependency. However, all of these works still require profiling and prediction, which as explained before, do not apply to our problem where the computation and communication latencies can fluctuate unpredictably over time.

Deep reinforcement learning (DRL) techniques have also been proposed for single-user dependent task offloading (e.g., [6] and [7]). DRL techniques require specific hyper parameter tuning tailored to each system setting, and they generally do not provide performance guarantees. Instead, in our work we propose a light-weight solution that is also competitive-ratio optimal.

There are further studies on task offloading in multi-user or multi-server systems, such as [20]-[22]. They mainly address resource contention among devices or applications. Therefore, their solutions are beyond the scope of our problem. Similar to the strategy in [20], our single-user solution may be used as a sub-module in the extension to more complex systems.

III. Online Offloading Problem Formulation

A. Sequentially Dependent Stages and Offloading

We consider a job with L sequential stages, with $\mathcal{L} = \{1, 2, \ldots, L\}$ denoting the indices of the stages. A local machine and a remote machine collaborate to process this job. The job is initiated at the local machine and may be offloaded to the remote machine at any stage. As explained above, DNN inference offloading with L dependent layers may be viewed as a prime example of this model.

Let P_i^l and P_i^r be the processing delays of stage i on the local and remote machines, respectively. The remote machine usually corresponds to a dedicated processor or virtual machine in the cloud with known speed. Additionally, when the jobs are of DNN inference type, the amount of computation required for each DNN layer solely depends on the number of neurons and weights. Therefore, it is reasonable to assume that the sequence $\{P_i^r\}_{i=1}^L$ is known.

However, due to fluctuation in the available computing power and shared applications at the local machine, $\{P_i^l\}_{i=1}^L$ is random. The only knowledge about P_i^l is its range defined with respect to P_i^r:

$$\alpha_{\text{min}} P_i^r \leq P_i^l \leq \alpha_{\text{max}} P_i^r \ , \tag{1}$$

where $0 < \alpha_{\text{min}} \leq \alpha_{\text{max}}$. The bounds α_{min} and α_{max} can estimated by computation profiling [4], [10], and in Section V and Section VI we will study the impact of their estimation errors on the competitive ratio and average performance of our solution.

When the job is offloaded at stage i, this means that stages 1 to $i-1$ have been processed locally, and stages i to L will be processed remotely. For mathematical convenience, we further define offloading at stage $L+1$ as not offloading at all. We assume that the remote machine has all programming for the job, so that offloading at stage i involves only transmitting the input data for stage i from the local machine to the remote machine. Note that the input data for stage i is available only after the processing of stage $i-1$ is completed, and the remote machine can start processing stage i only after the input data for this stage is received.

Let d_i be the input data size for stage i. Let $\mathcal{D} = \{d_1, d_2, \ldots, d_L, d_{L+1}\}$. Here, d_1 is the size of the input data (e.g., image for classification), and d_{L+1} is the size of the final output (e.g., inferred image label), which in practice is negligibly small and can be returned to the local machine instantly. The communication delay for offloading stage i is $C_i = \frac{d_i}{B_i}$, where B_i is the data rate at the time of offloading. We do not assume any advance knowledge of $\{B_i\}_{i=1}^L$, so that B_i and thus C_i become known only at stage i.

B. Online Delay Minimization Problem

We define an online delay minimization problem instance I as $\{P_i^l, P_i^r, d_i, B_i\}_{i=1}^L$. Given a problem instance, we consider the *total delay*, which is the time to finish all job stages $\mathcal{L}$. If the job is offloaded at stage i, then the total delay, denoted by T_i, includes the processing delay of the first $i-1$ stages on the local machine, the communication delay for offloading, and the processing delay of the last $L-i+1$ stages on the remote machine. Thus, we have

$$T_i = \sum_{k=1}^{i-1} P_k^l + C_i + \sum_{k=i}^{L} P_k^r. \tag{2}$$

Note that this formula applies also to the case $i = L+1$, which represents no offloading, by defining $C_{L+1} = 0$.

Our optimization objective is to find an optimal offloading point to minimize the total delay. In the offline setting where one has future information of the entire problem instance I, this problem can be easily solved by searching over all possible offloading stages. For problem instance I, the offline optimizer is given by

$$p_{\text{OPT}}(I) = \arg \min_{i \in \mathcal{L} \bigcup \{L+1\}} T_i \ . \tag{3}$$

Let $T_{\text{OPT}}(I)$ be the corresponding offline minimum. In practice, one does not have knowledge of the future processing delays P_i^l or communication delays C_i. Despite such uncertainty, we aim to design an online algorithm that provides performance guarantee.

Let $T_{\text{ALG}}(I)$ be the total delay achieved by an online algorithm for problem instance I. The standard performance measurement for an online algorithm is the *competitive ratio*:

$$\rho \triangleq \max_{I \in \mathcal{I}} \frac{T_{\text{ALG}}(I)}{T_{\text{OPT}}(I)}, \tag{4}$$

where $\mathcal{I}$ is the set of all problem instances. When such ρ exists, the online algorithm is called ρ-competitive. The competitive ratio is the worst-case multiplicative guarantee on the output of the online algorithm when compared with an optimal offline algorithm.

IV. Delay-optimal Online Partitioning

A. Algorithm Description

Our algorithm is motivated by an adversarial analysis of the possible outcomes of any action. On the one hand, if an online algorithm offloads at any stage $i \in \mathcal{L}$, the adversary will reduce the communication delays of stages $j > i$ and the local processing delays of stages $j \geq i$ to the minimum, to incur a worst possible outcome for the algorithm's decision. On the other hand, if the online algorithm does not offload at stage i, the adversary will increase the communication delays of the stages $j > i$ and the local processing delays of stages $j \geq i$ to the maximum. An effective solution needs to balance these two extremes.

Suppose an online algorithm is at stage i deciding whether to offload the remainder of the job starting from this stage. This means that all stages $j \leq i-1$ have already been processed on the local machine, incurring known delay $\sum_{k=1}^{i-1} P_k^l$. If the job is offloaded at stage i, then the total processing delay will be

$$\bar{T}_i = \sum_{k=1}^{i-1} P_k^l + \sum_{k=i}^{L} P_k^r \ , \tag{5}$$

which is a known quantity. If instead the job is offloaded at any later stage $j > i$, the total processing delay will be $\sum_{k=1}^{i-1} P_k^l + \sum_{k=i}^{j-1} P_k^l + \sum_{k=j}^{L} P_k^r$. We note that $\sum_{k=i}^{j-1} P_k^l$ is unknown at this time, but it is lower bounded by $\alpha_{\min} \sum_{k=i}^{j-1} P_k^r$. Considering all possible $i+1 \leq j \leq L+1$, it is easy to see that, at stage i, the minimum possible total processing delay is

$$T_i^{\min} = \sum_{k=1}^{i-1} P_k^l + \alpha_{\min} P_i^r + \min\{\alpha_{\min}, 1\} \sum_{k=i+1}^{L} P_k^r \ . \tag{6}$$

Similarly, while at stage i, the maximum possible total processing delay is

$$T_i^{\max} = \sum_{k=1}^{i-1} P_k^l + \max\{\alpha_{\max}, 1\} \sum_{k=i}^{L} P_k^r \ . \tag{7}$$

We also need to account for the offloading communication delay. In DOPart, we use the geometric mean of the lower and upper bounds above as the threshold for our offloading decision. At stage i, we observe the current communication data rate B_i to obtain the communication delay C_i. We offload the job at this stage if and only if

$$C_i + \bar{T}_i \leq \sqrt{T_i^{\min} T_i^{\max}} \ . \tag{8}$$

We remark here that (8) is similar in form to the optimal time-series search [23]. However, there are two main differences between our design and the standard time-series search. First, the lower and upper bounds of the processing time in our problem varies over time. Second, in addition to the sequence

Algorithm 1: Delay-optimal Online Partitioning (DOPart)

Input: $\mathcal{L}$, $\{d_i\}_{i=1}^{L}$, $\{P_i^r\}_{i=1}^{L}$, $\alpha_{\min}$, $\alpha_{\max}$

$\mathcal{F} \leftarrow \emptyset$
for $i \leftarrow 1$ *to* L **do**
  Compute $T_i^{\min}$, $T_i^{\max}$, and $\bar{T}_i$
  $B_i \leftarrow$ Observe current data rate
  $C_i \leftarrow d_i / B_i$
  if $C_i + \bar{T}_i \leq \sqrt{T_i^{min} T_i^{max}}$ **then**
    Transmit input data for stage i
    Process stages $\mathcal{L} \backslash \mathcal{F}$ on the remote machine
    Exit
  end
  Process stage i on the local machine
  $P_i^l \leftarrow$ Observe local processing delay of stage i
  $\mathcal{F} \leftarrow \mathcal{F} \cup i$
end

of processing times, here we need to take an additional factor, the communication delay, into consideration. Note that only the left-hand side of (8) contains the communication delay, while the right-hand side is based on the processing delay only. We will show in Section IV-B that this unique design leads to the best possible competitive ratio for our optimization problem. The pseudocode for the proposed DOPart algorithm is given in Algorithm 1.

Complexity: Since DOPart processes the stages sequentially, the time complexity of DOPart is $\mathcal{O}(L)$, i.e., linear in the number of stages. We note that this is a substantial reduction from existing methods that use deep reinforcement learning, which do not provide any performance guarantee.

B. Competitiveness Analysis

We derive the competitive ratio for DOPart under three different cases: (1) $\alpha_{\max} \leq 1$, (2) $\alpha_{\max} > 1$ and $\alpha_{\min}\alpha_{\max} < 1$, and (3) $\alpha_{\max} > 1$ and $\alpha_{\min}\alpha_{\max} \geq 1$. For all cases, we will additionally show that DOPart is competitive-ratio optimal, i.e., there does not exist another deterministic online algorithm that achieves a lower competitive ratio. This also directly implies that our derived competitive ratios are tight.

For notational convenience, for $i \in \mathcal{L} \bigcup \{L+1\}$ we define $T_{-i}^l = \sum_{k=1}^{i-1} P_k^l$, $T_{+i}^l = \sum_{k=i}^{L} P_k^l$, and $T^l = T_{-i}^l + T_{+i}^l$, with the last being the total local processing delay if all stages are processed locally. We also define T_{-i}^r, T_{+i}^r, and T^r similarly, for the remote processing delays.

Before presenting the main results, we require the following two lemmas, whose proofs are omitted due to the page limit.

Lemma 1. *For any $\alpha_{min} \leq \alpha_{max}$, we have $T_{-i}^l + \alpha_{max} T_{+i}^r \leq T_{-k}^l + \alpha_{max} T_{+k}^r$ for all $i > k$.*

Lemma 2. *For any $\alpha_{min} \leq \alpha_{max}$, we have $T_{-i}^l + \alpha_{min} P_i^r + \min\{\alpha_{min}, 1\} T_{+(i+1)}^r \leq T_{-k}^l + \min\{\alpha_{min}, 1\} T_{+k}^r$ for all $i < k$.*

Furthermore, since $\alpha_{\min} \geq \min\{\alpha_{\min}, 1\}$, the corollary below directly follows from Lemma 2:

Corollary 2.1. *For any α_{min} and α_{max}, we have $T_{-i}^l + \alpha_{min} P_i^r + \min\{\alpha_{min}, 1\} T_{+(i+1)}^r \leq T_{-k}^l + \alpha_{min} P_k^r + \min\{\alpha_{min}, 1\} T_{+(k+1)}^r$ for all $i < k$.*

1) Case (1): $\alpha_{max} \leq 1$: In the simplest case of $\alpha_{\max} \leq 1$, i.e., the processing delay of the local machine is always less than that on the remote machine, DOPart is 1-competitive. This means that it performs as well as the optimal offline algorithm, and clearly no other algorithm can do better.

Theorem 1. *If $\alpha_{max} \leq 1$, DOPart is 1-competitive.*

Proof. When $\alpha_{\max} < 1$, since $\alpha_{\min} \leq \alpha_{\max}$, we have $\alpha_{\min}\alpha_{\max} < 1$. Based on (8), DOPart never offloads i.e., it processes all the stages locally. Meanwhile, the best offline strategy in hindsight is to never offload because $C_i \geq 0$, $\forall i \in \mathcal{L}$, and $P_i^l < P_i^r$, $\forall i \in \mathcal{L}$. Therefore, DOPart behaves the same as the optimal offline algorithm in this case. □

2) Case (2): $\alpha_{max} > 1$ and $\alpha_{min}\alpha_{max} < 1$: Unlike the previous trivial case, when $\alpha_{\max} > 1$, there can be potential benefit in offloading the job to the remote machine. In this case, since we also have $\alpha_{\min} < 1/\alpha_{\max} < 1$, the local machine can be faster than the remote machine. Therefore, DOPart is required to be more conservative with the offloading decision.

First, since in this case $\min\{\alpha_{\min}, 1\} = \alpha_{\min}$, we have a further corollary from Lemma 2:

Corollary 2.2. *When $\alpha_{min} < 1$, $T_{-i}^l + \alpha_{min} T_i^r \leq T_{-k}^l + \alpha_{min} T_{+k}^r$ for all $i < k$.*

We also require the following lemma, whose proof is omitted due to the page limit.

Lemma 3. *When $\alpha_{max} > 1$ and $\alpha_{min}\alpha_{max} < 1$, if DOPart offloads at stage i then $\alpha_{min} + \alpha_{max} > 2$ and $T_{-i}^l \geq \phi T_{+i}^r$, where*

$$\phi = \left(\frac{1 - \alpha_{min}\alpha_{max}}{\alpha_{min} + \alpha_{max} - 2} \right). \tag{9}$$

Theorem 2. *If $\alpha_{max} > 1$ and $\alpha_{min}\alpha_{max} < 1$,*

(i) No deterministic online algorithm can do better than α_{max}-competitive, and
(ii) DOPart is α_{max}-competitive.

Proof. First we prove part (i). To show this we take an adversarial approach. The adversary first sets $C_1 = 0$. If any deterministic online algorithm ALG offloads at stage 1, the total delay $T_{\text{ALG}} = T^r$. The adversary then sets the local processing latencies $P_k^l = \alpha_{\min} P_k^r \ \forall \ k \in \{1, \ldots, L\}$. Since $\alpha_{\min} < 1$, the total delay of optimal offline algorithm $T_{\text{OPT}} = \alpha_{\min} T^r$. Thus in this case $T_{\text{ALG}}/T_{\text{OPT}} \geq 1/\alpha_{\min}$. If ALG does not offload at stage 1, the adversary sets the local processing latencies $P_k^l = \alpha_{\max} P_k^r$, $\forall \ k \in \{1, \ldots, L\}$, and $C_k = (\alpha_{\max} - 1) T_{+k}^r$, $\forall \ k \in \{2, \ldots, L\}$. In this case the total delay $T_{\text{ALG}} = \alpha_{\max} T^r$. Since $\alpha_{\max} > 1$,

$T_{\text{OPT}} = C_1 + T^r = T^r$. Then in this case $T_{\text{ALG}}/T_{\text{OPT}} \geq \alpha_{\max}$. Because $1/\alpha_{\min} > \alpha_{\max}$, we have

$$\rho_{\text{ALG}} \geq \min\left\{\alpha_{\max}, \frac{1}{\alpha_{\min}}\right\} = \alpha_{\max} \ .$$

We now prove part (ii). If $\alpha_{\min} + \alpha_{\max} \leq 2$, from Lemma 3, DOPart does not offload at any stage. Then the total delay is $T_{\text{DOPart}} = T^l$. Suppose the optimal offline algorithm OPT that offloads at layer k. Its total delay is $T_{\text{OPT}} = T^l_{-k} + C_k + T^r_{+k} \geq T^l_{-k} + T^r_{+k}$, with equality when $C_k = 0$. The competitive ratio is then given by

$$\begin{aligned}\rho_{\text{DOPart}} &= \max_{I\in\mathcal{I}}\left\{\frac{T^l}{T^l_{-k}+T^r_{+k}}\right\} = \max_{I\in\mathcal{I}}\left\{\frac{T^l_{-k}+T^l_{+k}}{T^l_{-k}+T^r_{+k}}\right\} \\ &\overset{(a)}{=} \max_{I\in\mathcal{I}}\left\{\frac{T^l_{-k}+\alpha_{\max}T^r_{+k}}{T^l_{-k}+T^r_{+k}}\right\} \overset{(b)}{=} \alpha_{\max}. \end{aligned} \tag{10}$$

Here equality (a) is because $T^l_{+k} \leq \alpha_{\max} T^r_{+k}$, and equality (b) is because $\alpha_{\max} > 1$.

The case where $\alpha_{\min}+\alpha_{\max} > 2$ is more complicated. Define $q = \min\{s \in \mathcal{L}\bigcup\{L+1\} | T^l_{-s} \geq \phi T^r_{+s}\}$. From Lemma 3 we know DOPart offloads at stage i such that $T^l_{-i} \geq \phi T^r_{+i}$, so we have $i \geq q$. The total delay of DOPart is given by

$$\begin{aligned} T_{\text{DOPart}} &= T^l_{-i} + C_i + T^r_{+i} \\ &\overset{(a)}{\leq} T^l_{-i} + \sqrt{T_i^{\min} T_i^{\max}} - \bar{T}_i + T^r_{+i} \\ &\overset{(b)}{=} \sqrt{\left(T^l_{-i} + \alpha_{\min}T^r_{+i}\right)\left(T^l_{-i} + \alpha_{\max}T^r_{+i}\right)}. \end{aligned} \tag{11}$$

Here, inequality (a) is due to the offloading threshold in (8). Equality (b) is because $\min\{\alpha_{\min}, 1\} = \alpha_{\min}$, so that $\alpha_{\min}P^r_i + \min\{\alpha_{\min}, 1\}T^r_{+(i+1)} = \alpha_{\min}T^r_{+i}$.

Suppose OPT offloads at stage $k \in \mathcal{L}\bigcup\{L+1\}$. Its latency is

$$T_{\text{OPT}} = T^l_{-k} + C_k + T^r_{+k} \geq T^l_{-k} + T^r_{+k}. \tag{12}$$

Then from (11) and (12) we have

$$\frac{T^2_{\text{DOPart}}}{T^2_{\text{OPT}}} \leq \max_{I\in\mathcal{I}}\left\{\frac{\left(T^l_{-i} + \alpha_{\min}T^r_{+i}\right)\left(T^l_{-i} + \alpha_{\max}T^r_{+i}\right)}{\left(T^l_{-k}+T^r_{+k}\right)^2}\right\} \tag{13}$$

If $k < q$, from (13) we have

$$\begin{aligned}\frac{T^2_{\text{DOPart}}}{T^2_{\text{OPT}}} &\overset{(a)}{\leq} \max_{I\in\mathcal{I}}\left\{\frac{T^l_{-k}+T^l_{-i}-T^l_{-k}+\alpha_{\min}T^r_{+i}}{T^l_{-k}+T^r_{+k}}\right. \\ &\qquad \left.\frac{T^l_{-k}+T^l_{-i}-T^l_{-k}+\alpha_{\max}T^r_{+i}}{T^l_{-k}+T^r_{+k}}\right\} \\ &\overset{(b)}{\leq} \max_{I\in\mathcal{I}}\left\{\frac{T^l_{-k}+\alpha_{\max}\left(T^r_{+k}-T^r_{+i}\right)+\alpha_{\min}T^r_{+i}}{T^l_{-k}+T^r_{+k}}\right. \\ &\qquad \left.\frac{T^l_{-k}+\alpha_{\max}\left(T^r_{+k}-T^r_{+i}\right)+\alpha_{\max}T^r_{+i}}{T^l_{-k}+T^r_{+k}}\right\} \\ &\overset{(c)}{\leq} \max_{I\in\mathcal{I}}\left\{\left(\frac{T^l_{-k}+\alpha_{\max}T^r_{+k}}{T^l_{-k}+T^r_{+k}}\right)^2\right\} \overset{(d)}{\leq} \alpha^2_{\max}\end{aligned}$$

Here, equality (a) is because T^l_{-i} can be split into T^l_{-k} and $T^l_{-i} - T^l_{-k}$ since $k \leq i$. Inequality (b) is because $T^l_{-i} - T^l_{-k} \leq \alpha_{\max}(T^r_{+k} - T^r_{+i})$ from Lemma 1. Inequality (c) is because $(\alpha_{\min} - \alpha_{\max})T^r_{+i} \leq 0$. Inequality (d) is because $\alpha_{\max} > 1$.

If $q \leq k < i$, we have

$$\begin{aligned} T_{\text{OPT}} &= T^l_{-k} + C_k + T^r_{+k} \\ &\overset{(a)}{\geq} T^l_{-k} + \sqrt{T_k^{\min}T_k^{\max}} - \bar{T}_k + T^r_{+k} \\ &\overset{(b)}{=} \sqrt{\left(T^l_{-k} + \alpha_{\min}T^r_{+k}\right)\left(T^l_{-k} + \alpha_{\max}T^r_{+k}\right)}. \end{aligned} \tag{14}$$

Here, inequality (a) is because $k < i$, which means that DOPart does not offload at k, so from (8) we have $C_k \geq \sqrt{T_k^{\min}T_k^{\max}} - \bar{T}_k$. Equality (b) is because of (5). From (11) and (14) we have

$$\begin{aligned}\frac{T^2_{\text{DOPart}}}{T^2_{\text{OPT}}} &\leq \max_{I\in\mathcal{I}}\left\{\frac{\left(T^l_{-i} + \alpha_{\min}T^r_{+i}\right)\left(T^l_{-i} + \alpha_{\max}T^r_{+i}\right)}{\left(T^l_{-k} + \alpha_{\min}T^r_{+k}\right)\left(T^l_{-k} + \alpha_{\max}T^r_{+k}\right)}\right\} \\ &\overset{(a)}{\leq} \max_{I\in\mathcal{I}}\left\{\frac{T^l_{-k}+\alpha_{\max}\left(T^r_{+k}-T^r_{+i}\right)+\alpha_{\min}T^r_{+i}}{T^l_{-k}+\alpha_{\min}T^r_{+k}}\right. \\ &\qquad \left.\frac{T^l_{-k}+\alpha_{\max}\left(T^r_{+k}-T^r_{+i}\right)+\alpha_{\max}T^r_{+i}}{T^l_{-k}+\alpha_{\max}T^r_{+k}}\right\} \\ &\overset{(b)}{\leq} \max_{I\in\mathcal{I}}\left\{\frac{T^l_{-k}+\alpha_{\max}T^r_{+k}}{T^l_{-k}+\alpha_{\min}T^r_{+k}}\right\} \\ &\overset{(c)}{\leq} \max_{I\in\mathcal{I}}\left\{\frac{\phi T^r_{+k}+\alpha_{\max}T^r_{+k}}{\phi T^r_{+k}+\alpha_{\min}T^r_{+k}}\right\} \\ &= \left(\frac{\alpha_{\max}-1}{1-\alpha_{\min}}\right)^2 \overset{(d)}{<} \alpha^2_{\max}\end{aligned}$$

Here, inequality (a) is because T^l_{-i} can be split into T^l_{-k} and $T^l_{-i} - T^l_{-k}$, and $T^l_{-i} - T^l_{-k} \leq \alpha_{\max}(T^r_{+k} - T^r_{+i})$ from Lemma 1. Inequality (b) is because $(\alpha_{\min} - \alpha_{\max})T^r_{+i} \leq 0$ since $\alpha_{\min} \leq \alpha_{\max}$. Inequality (c) is because of Lemma 3 and $\alpha_{\max} > \alpha_{\min}$. Inequality (d) is because $\alpha_{\min} < 1/\alpha_{\max}$.

If $k = i$, $T_{\text{DOPart}} = T_{\text{OPT}}$. Then we have $\frac{T_{\text{DOPart}}}{T_{\text{OPT}}} = 1 < \alpha_{\max}$.

If $k > i$, then from (13) we have

$$\begin{aligned}\frac{T^2_{\text{DOPart}}}{T^2_{\text{OPT}}} &\overset{(a)}{<} \max_{I\in\mathcal{I}}\left\{\frac{\left(T^l_{-i} + \alpha_{\min}T^r_{+i}\right)\left(T^l_{-i} + \alpha_{\max}T^r_{+i}\right)}{\left(T^l_{-i} + \alpha_{\min}T^r_{+i}\right)^2}\right\} \\ &\overset{(b)}{=} \frac{\phi T^r_{+i} + \alpha_{\max}T^r_{+i}}{\phi T^r_{+i} + \alpha_{\min}T^r_{+i}} = \left(\frac{\alpha_{\max}-1}{1-\alpha_{\min}}\right)^2 \overset{(c)}{<} \alpha^2_{\max}\end{aligned}$$

Here, inequality (a) is from $T^l_{-k} + T^r_{+k} > T^l_{-k} + \alpha_{\min}T^r_{+k}$ and Corollary 2.2. Equality (b) is from Lemma 3 and $\alpha_{\max} > \alpha_{\min}$. Inequality (c) is because $\alpha_{\min} < 1/\alpha_{\max}$.

Summarizing the above, we conclude that we also have $\rho_{\text{DOPart}} = \alpha_{\max}$ for the case $\alpha_{\min} + \alpha_{\max} > 2$. □

3) Case (3): $\alpha_{max} > 1$ and $\alpha_{min}\alpha_{max} \geq 1$: This case represents an environment where offloading to the remote machine generally offers a higher speedup in comparison with case (2). Since there is a higher chance for delay reduction in offloading, our algorithm is allowed to be more liberal with the offloading threshold.

Theorem 3. *If $\alpha_{max} > 1$ and $\alpha_{max}\alpha_{min} \geq 1$,*

(i) No deterministic online algorithm can do better than $\sqrt{\alpha_{max}/\gamma}$-competitive where

$$\gamma = \min\{\alpha_{min}, 1\} + (\alpha_{min} - \min\{\alpha_{min}, 1\})\,(P_1^r/T^r)\,, \quad (15)$$

(ii) DOPart is $\sqrt{\alpha_{max}/\gamma}$-competitive, and

(iii) A simplified form follows that DOPart is at least $\sqrt{\alpha_{max}/\min\{\alpha_{min}, 1\}}$-competitive.

Proof. We proof parts (i) and (ii) similarly to the corresponding parts of Theorem 2, except in this case $\alpha_{\min} \geq 1/\alpha_{\max}$ and we also take into account $\alpha_{\min} > 1$. Part (iii) follows directly from part (ii) since $(\alpha_{\min} - \min\{\alpha_{\min}, 1\})(P_1^r/T^r) \geq 0$. □

V. Competitive Ratio with Estimation Errors

We extend our competitive analysis to the scenario where we only have access to estimates of $\alpha_{\min}$ and $\alpha_{\max}$, defined as $\hat{\alpha}_{\min} = \alpha_{\min}(1+\epsilon_1)$ and $\hat{\alpha}_{\max} = \alpha_{\max}(1+\epsilon_2)$, with errors $\epsilon_1, \epsilon_2 \in \mathbb{R}$. Without loss of generality, we assume that the estimation errors are small enough so that $\hat{\alpha}_{\min} \leq \hat{\alpha}_{\max}$.

The following theorems indicate that even with imprecise estimation of $\alpha_{\min}$ and $\alpha_{\max}$, DOPart provides bounded performance guarantees. Our competitive ratio analysis in this more complex scenario uses similar techniques as in Section IV-B, with additional finesse required to account for the estimation errors and handle different cases where ϵ_1 and ϵ_2 are positive or negative. The proof details are omitted due to the page limit.

Theorem 4. *If $\hat{\alpha}_{max} \leq 1$, $\rho_{DOPart} = \max\{1, \alpha_{max}\}$.*

Theorem 5. *If $\hat{\alpha}_{max} > 1$ and $\hat{\alpha}_{min}\hat{\alpha}_{max} < 1$,*

$$\rho_{DOPart}(\epsilon_1, \epsilon_2) = \alpha_{max} \max\{1, 1+\epsilon_2, (1+\epsilon_1)(1+\epsilon_2)\}.$$

Theorem 6. *If $\hat{\alpha}_{max} > 1$ and $\hat{\alpha}_{min}\hat{\alpha}_{max} \geq 1$,*

$$\rho_{DOPart}(\epsilon_1, \epsilon_2) = \sqrt{\frac{\alpha_{max}}{\min\{\alpha_{min}, 1\}} \frac{1}{1-|\epsilon_1|} \frac{1}{1-|\epsilon_2|}}.$$

The above results give insight into how the performance of DOPart is impacted by ϵ_1 and ϵ_2. Comparing Theorems 5 and 6 with their counter parts Theorems 2 and 3, and applying the Taylor series expansion on the extra factors, we observe that the competitiveness of DOPart deteriorates approximately linearly in ϵ_1 and ϵ_2. Therefore, DOPart is robust against small to moderate estimation errors.

VI. Numerical Evaluation

We study the numerical performance of DOPart, with experiments on AlexNet, VGG16, and ResNet34. The stages of AlexNet and VGG16 directly correspond to their layers of neurons, while the stages of ResNet34 are more complex as shown in Fig. 1.

The remote processing latencies P_k^r for each stage are obtained by executing 5000 samples each of AlexNet, VGG19, and ResNet34 inference jobs on a dedicated Digital Research Alliance of Canada cluster with Nvidia-T4 GPUs. The local processing latency for stage k is set to $P_k^l = 2^\mu P_k^r$ where μ is drawn from the uniform distribution between $\log_2 \alpha_{\min}$ and $\log_2 \alpha_{\max}$, to emulate the fluctuation in the local machine's processing speed due to random demands from parallel applications on the same machine. We simulate a random communication channel for offloading, where the bandwidth B_i for stage i is drawn from the uniform distribution between 5Mbps and 60Mbps, which emulates the wireless fading channel of a mobile device.

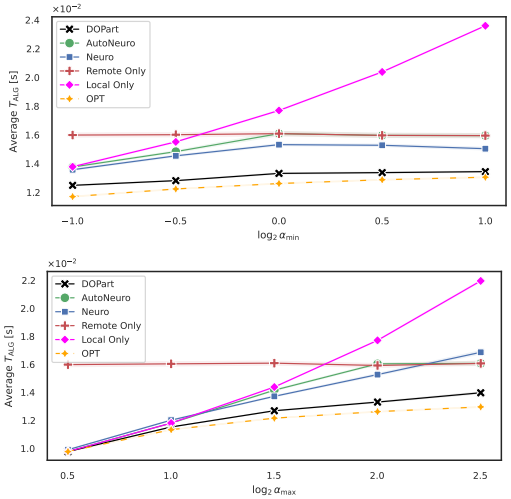

Fig. 2. Average delay for ResNet34, with 95% confidence intervals.

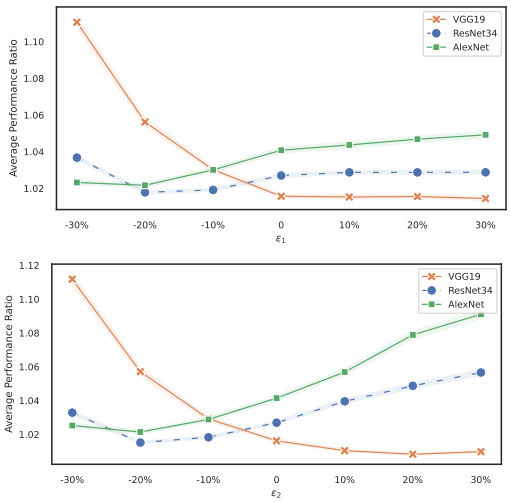

Fig. 3. Performance ratio vs. parameter estimation errors, with 95% confidence intervals.

We compare DOPart against the following benchmarks:

- **Neuro:** Neurosurgeon [4] predicts the future processing and communication delays based on a pre-trained model. We implement the best version of Neurosurgeon by always using the value that minimizes the prediction error for the future processing latencies of all the layers. It uses only the network bandwidth at the beginning of the execution of the inference job. We remark that the method in [10] behaves the same way as Neurosurgeon for sequentially dependent tasks, so this benchmark also represents [10].
- **AutoNeuro:** Autodidactic Neurosurgeon [8] uses an online learning approach to solve the sequential dependent task partitioning problem. To compare with [8] in its

most favourable performance, we implement the oracle from [8], which is the idealized best prediction that one can achieve. For each experimental setting we do so by finding the partition point that minimizes the average delay over 5000 executions of the inference job.

- **Local Only:** This is a trivial strategy that execute the inference job locally for T^l total delay.
- **Remote Only:** This is a trivial strategy that offloads the complete inference job to the remote device, which incurs $C_1 + T^r$ total delay.

In Fig. 2, we present the average delay of DOPart and all benchmarks over 5000 instances of inference jobs on ResNet34. We observe that DOPart can substantially outperform both Neuro and AutoNeuro. Consider an example of a typical scenario where $\alpha_{\min} = 1$ and $\alpha_{\max} = 4$. DOPart provides 17% improvement over Neuro and 13% improvement over AutoNeuro. Furthermore, we observe that the average delay increases with $\alpha_{\min}$ and $\alpha_{\max}$. This is because the expected local processing latencies increase with increasing $\alpha_{\min}$ and $\alpha_{\max}$. Similar observation are made with AlexNet and VGG19. We omit those figures to avoid redundancy.

We are also interested in the impact of the estimation accuracy of $\alpha_{\min}$ and $\alpha_{\max}$ on the performance of DOPart. Fig. 3 shows how the average empirical performance ratio between of DOPart and the optimal offline algorithm responds to changes in the estimation errors ϵ_1 and ϵ_2 on $\alpha_{\min}$ and $\alpha_{\max}$, respectively. We set $\alpha_{\min} = 1$ and $\alpha_{\max} = 3$. Our main observation is that DOPart still achieves excellent performance ratios even for up to $20\% \sim 30\%$ estimation error. Furthermore, it is interesting to see that for AlexNet and ResNet34 the performance initially improves, but for VGG19 it degrades, with underestimation of both $\alpha_{\min}$ and $\alpha_{\max}$. The results are reversed for overestimation of $\alpha_{\min}$ and $\alpha_{\max}$. This suggests that there is no general trend on whether one should favour underestimation or overestimation.

VII. Conclusion

In this paper, we propose a lightweight online Delay-optimal Online Partitioning (DOPart) algorithm for the offloading of computation jobs with sequentially dependent tasks. Through dynamic adaptation to the unpredictable fluctuation in the system state, it encourages optimal sharing between local and remote computational resources while considering the impact of offloading delay. We obtain the competitive ratio of DOPart and show that it is optimal, by providing a matching lower bound for the delay minimization problem. We further observe that DOPart is robust against estimation errors on $\alpha_{\min}$ and $\alpha_{\max}$, in terms of both the competitive ratio and numerical average performance.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[2] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Proc. International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2010, pp. 49–62.

[3] 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, 2016.

[4] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[5] Y. Huang, F. Wang, F. Wang, and J. Liu, "Deepar: A hybrid device-edge-cloud execution framework for mobile deep learning applications," in *Proc. IEEE INFOCOM WKSHPS*, 2019, pp. 892–897.

[6] J. Yan, S. Bi, and Y. J. A. Zhang, "Offloading and resource allocation with general task graph in mobile edge computing: A deep reinforcement learning approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 8, pp. 5404–5419, 2020.

[7] J. Wang, J. Hu, G. Min, W. Zhan, A. Y. Zomaya, and N. Georgalas, "Dependent task offloading for edge computing based on deep reinforcement learning," *IEEE Trans. Comp.*, vol. 71, no. 10, pp. 2449–2461, 2021.

[8] L. Zhang, L. Chen, and J. Xu, "Autodidactic neurosurgeon: Collaborative deep inference for mobile edge intelligence via online learning," in *Proc. the Web Conference 2021*, 2021, pp. 3111–3123.

[9] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE Network*, vol. 32, no. 1, pp. 96–101, 2018.

[10] C. Hu, W. Bao, D. Wang, and F. Liu, "Dynamic adaptive dnn surgery for inference acceleration on the edge," in *Proc. IEEE Conference on Computer Communications (INFOCOM)*, 2019, pp. 1423–1431.

[11] S. Zhang, Y. Li, X. Liu, S. Guo, W. Wang, J. Wang, B. Ding, and D. Wu, "Towards real-time cooperative deep inference over the cloud and edge end devices," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 2, pp. 1–24, 2020.

[12] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, "Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services," *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, 2019.

[13] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *IEEE 37th Iinternational Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 328–339.

[14] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge AI: On-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2019.

[15] H. Li, C. Hu, J. Jiang, Z. Wang, Y. Wen, and W. Zhu, "Jalad: Joint accuracy-and latency-aware deep structure decoupling for edge-cloud execution," in *Proc. IEEE ICPADS*. IEEE, 2018, pp. 671–678.

[16] L. Wang, L. Xiang, J. Xu, J. Chen, X. Zhao, D. Yao, X. Wang, and B. Li, "Context-aware deep model compression for edge cloud computing," in *Proc. IEEE ICDCS*. IEEE, 2020, pp. 787–797.

[17] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 4, pp. 23–32, 2013.

[18] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE INFOCOM WKSHPS*, 2014, pp. 352–357.

[19] H. Wu, W. J. Knottenbelt, and K. Wolter, "An efficient application partitioning algorithm in mobile environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1464–1480, 2019.

[20] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, 2014.

[21] S. Liu, Y. Yu, X. Lian, Y. Feng, C. She, P. L. Yeoh, L. Guo, B. Vucetic, and Y. Li, "Dependent task scheduling and offloading for minimizing deadline violation ratio in mobile edge computing networks," *IEEE J. Selected Areas in Commun.*, vol. 41, no. 2, pp. 538–554, 2023.

[22] F. Wang, S. Cai, and V. K. Lau, "Sequential offloading for distributed DNN computation in multiuser MEC systems," *IEEE Internet of Things Journal*, vol. 10, no. 20, pp. 18 315–18 329, 2023.

[23] R. El-Yaniv, A. Fiat, R. M. Karp, and G. Turpin, "Optimal search and one-way trading online algorithms," *Algorithmica*, vol. 30, pp. 101–139, 2001.