Scheduling Machine Learning Compressible Inference Tasks with Limited Energy Budget

Tiago da Silva Barros, Davide Ferre, Frederic Giroire, Ramon Aparicio-Pardo, Stephane Perennes

Scheduling Machine Learning Compressible Inference Tasks with Limited Energy Budget

Tiago da Silva Barros
Université Côte d'Azur/I3S/Inria
Sophia Antipolis, France
dasilva@i3s.unice.fr

Davide Ferré
CNRS/Université Côte d'Azur/I3S/Inria
Sophia Antipolis, France
davide.ferre@inria.fr

Frédéric Giroire
CNRS/Université Côte d'Azur/I3S/Inria
Sophia Antipolis, France
frederic.giroire@inria.fr

Ramon Aparicio-Pardo
Université Côte d'Azur/CNRS/I3S
Sophia Antipolis, France
raparicio@i3s.unice.fr

Stéphane Pérennes
CNRS/Université Côte d'Azur/I3S/Inria
Sophia Antipolis, France
stephane.perennes@inria.fr

ABSTRACT

Advancements in cloud computing have boosted Machine Learning as a Service (MLaaS), highlighting the challenge of scheduling tasks under latency and deadline constraints. Neural network compression offers the latency and energy consumption reduction in data centers, aligning with efforts to minimize cloud computing's carbon footprint, despite some accuracy loss.

This paper investigates the Deadline Scheduling with Compressible Tasks - Energy Aware (DSCT-EA) problem, which addresses the scheduling of compressible machine learning tasks on several machines, with different speeds and energy efficiencies, under an energy budget constraint. Solving DSCT-EA involves determining both the machine on which each task will be processed and its processing time, a problem that has been proven to be NP-Hard. We formulate DSCT-EA as a Mixed-Integer Programming (MIP) problem and also provide an approximation algorithm for solving it. The efficacy of our approach is demonstrated through extensive experimentation, revealing its superiority over traditional scheduling techniques. It allows to save up to 70% of the energy budget of image classification tasks, while only losing 2% of accuracy compared to when not using compression.

KEYWORDS

scheduling, energy budget, neural network compression, deadlines

ACM Reference Format:
Tiago da Silva Barros, Davide Ferré, Frédéric Giroire, Ramon Aparicio-Pardo, and Stéphane Pérennes. 2024. Scheduling Machine Learning Compressible Inference Tasks with Limited Energy Budget. In *The 53rd International Conference on Parallel Processing (ICPP '24), August 12–15, 2024, Gotland, Sweden.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3673038.3673106

ICPP '24, August 12–15, 2024, Gotland, Sweden

ACM ISBN 979-8-4007-1793-2/24/08
https://doi.org/10.1145/3673038.3673106

1 INTRODUCTION

In recent years, there has been a growing usage of Machine Learning (ML) models in cloud computing, contributing to the adoption of Machine Learning as a Service (MLaaS) [18], studied in several applications such as image recognition and self-driving cars[17].

Cloud and network operators have faced challenges in developing efficient strategies for utilizing computational resources to support machine learning tasks. Among these challenges, scheduling is an important one. A scheduler must determine on which machine each task is executed and its processing order. This becomes especially critical when tasks must adhere to deadline constraints.

In the context of saving computational resources, researchers have investigated neural network compression techniques, including pruning [13] and quantization [26]. However, these approaches typically involve compressing the model during the training phase, necessitating re-training the model after compression.

Recent approaches compress neural networks at inference time [3, 28], reducing network size to varying degrees. Greater compression yields lower latency (i.e., the processing time of a task) but at the expense of accuracy. In [5], we introduced a scheduling system using compressible neural networks for image classification tasks, in which several heterogeneous machines could be used. Wedeveloped an approximation algorithm with proven guarantees for maximizing the average accuracy while respecting deadlines constraints.

Cloud and network operators are compelled to mitigate their cloud carbon footprint, driving researchers and scientists to investigate novel methods for conducting ML inference with greater energy efficiency.

The adoption of MLaaS and the expanding size of neural network models has resulted in increased energy consumption, particularly during the inference stage [15]. According to reports from NVIDIA [9], 80-90% of Artificial Intelligence (AI) costs stem from inference. Indeed, processing millions of requests, such as those encountered in social networks, can lead to a significant number of inferences in deep learning models, resulting in elevated energy consumption and a sizable carbon footprint [7].

Consequently, new energy-aware scheduling solutions are urgently needed, even though the introduction of energy consumption

considerations further complicates the scheduling problem. Neural network compression could be used to reduce the energy consumption of a model by reducing its size during inference. Therefore, we propose constraining cloud inference services to operate within an energy budget, which sets a maximum energy threshold for executing a set of inference tasks. The energy consumption of a task is influenced by both its processing time and the specific machine that executes it, given that power consumption can vary across different machines. In this paper, we extend our previous work to tackle the challenge of limiting cloud energy consumption when scheduling compressible tasks.

Our objective is to maximize accuracy while adhering to an energy budget constraint. The scheduling system determines task-to-machine allocations and processing times for each job, ensuring compliance with deadlines and energy budget limitations.

We formulate the problem as a non-linear program and develop an approximation algorithm with proven guarantees. Subsequently, we evaluate our approach against traditional scheduling techniques.

The contributions of this paper can be summarized as follows:

- We investigate the scheduling problem in which tasks can be compressed, and the energy consumption must adhere to an energy budget. The tasks under consideration are inference deep learning tasks, which present a trade-off between latency and accuracy. We believe our approach could be extended to other tasks exhibiting diminishing marginal gain.
- We analyze the problem complexity and propose (i) an exact algorithm to solve the problem using fractional relaxation, which serves as the basis for (ii) an approximation algorithm with proven guarantees for various heterogeneous machines (in terms of speed and energy efficiency) that satisfy the energy budget constraint.
- Finally, we validate and compare our approach with state-of-the-art solutions.

The rest of this paper is organized as follows. In Section 2, we provide a detailed review of related works, before introducing the problem formulation in Section 3. Sections 4 and 5 are dedicated to the exact and the approximation scheduling algorithms that we propose, respectively. In Section 6, we assess the algorithm's performance through experiments. Finally, we delve into the main discussions and conclusions in Section 7, alongside potential future works.

2 RELATED WORKS

NN Compression. Several papers have tackled the model compression problem during the training phase, employing various techniques such as pruning and knowledge distillation. For an extensive review, see [4].

Subsequently, some researchers [3, 28] have proposed solutions to compress models during the inference phase without requiring retraining. For instance, Cai et al. [3] introduced the Once-For-All (OFA) method to compress Convolutional Neural Networks (CNNs) across four dimensions: width, kernel size, depth, and image resolution, aiming to reduce the neural network's execution time while minimizing the impact on model accuracy, particularly for image classification tasks. In fact, the number of possible sub networks is so high (for MobileNet, more than 10^{19}) that we can compress the model to any compression size. In this way, we argue we have a continuous processing time (related to the compression levels) for the inference tasks.

Additionally, in [5], the authors expanded upon the initial parameters utilized in OFA and observed an exponential relationship between accuracy and inference latency tradeoff.

Scheduling DL inference tasks Numerous papers have been dedicated to addressing the scheduling problem for classical tasks. A seminal example is the work of Leyland and Liu [12], where they introduced the Earliest Deadlines First (EDF) algorithm.

The application of scheduling techniques to deep learning tasks has received significant attention in recent years. Various solutions [8, 16, 19] have been proposed for scheduling inference request tasks in cloud and edge environments. Notably, the work by Nigade et al. [16] closely aligns with our research objectives. They investigated methods to maximize the accuracy of a set of deep learning model inference requests while meeting network and processing latency constraints.

On top of that, several papers have addressed the scheduling problem with the aim of minimizing energy consumption.

Xu et al. [25] investigated a Mobile Edge Computing (MEC) environment to minimize energy consumption by determining the number of tasks offloaded to the edge server. Ma et al. [14] aimed to maximize a weighted function balancing reliability and accuracy within an energy budget, optimizing task allocation. Wang et al. [23] analyzed GPU energy consumption with spatial multitasking (simultaneous execution of multiple kernels). They proposed a scheduling system to balance energy efficiency and performance.

However, none of the aforementioned works consider compressible jobs or involve a small number of models with different sizes to fulfill tasks.

Scheduling compressible jobs Some studies have explored scheduling tasks with compressible processing times. For example, Vickson et al. [22], and Shabtay et al. [20] have proposed task compression techniques. However, these studies primarily focus on scenarios involving a single machine or a single deadline.

Regarding scheduling under energy constraints, some works have addressed the single-machine scenario [21, 27]. Yin et al. [27] aimed to minimize tardiness and energy consumption on a single machine, considering discrete levels of compression. Tsao et al. [21] sought to minimize energy costs using a genetic algorithm to optimize task processing times.

On the other hand, Wu et al. [24] addressed the problem of scheduling in multiple machines with controllable processing times. They aimed to minimize makespan and energy consumption, considering variables such as job processing time (discrete levels) and the machines where tasks are assigned. They proposed an evolutionary metaheuristic for solving the problem.

However, the above works never consider both deadlines and multiple machines, and they often employ discrete levels of task compression. In contrast, our solution utilizes the Once-For-All approach [3] to accommodate compressible tasks. Additionally, we are the first to propose an approximation algorithm with proven guarantees for scheduling such compressible tasks under energy constraints.

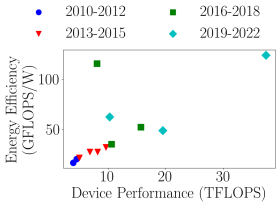


Figure 1: Energy efficiency vs speed for various GPUs [7].

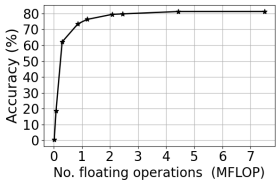


Figure 2: Once-for-All: Accuracy vs number of floating operations [5]

3 PROBLEM FORMULATION

In this section, we define the Deadline Scheduling with Compressible Tasks - Energy Aware (DSCT-EA) problem, and formulate it as a Mixed-Integer Program (MIP). Let us consider a set J of n tasks, which can be scheduled on a set M of m machines.

Each machine $r \in M$ is characterized by its speed s_r, quantified in FLOPS (Floating-point Operations per Second), power consumption P_r, measured in Watts (W), and energy efficiency $E_r = s_r/P_r$, computed as the ratio of speed to power consumption. In this regard, the energy efficiency of servers can exhibit significant heterogeneity, largely depending on the hardware architecture specifications. Figure 1, derived from a comprehensive evaluation conducted by Desislavov et al. [7], illustrates the relationship between energy efficiency and speed for across various NVIDIA GPUs designed for servers . The general observed trend is that devices exhibit linear improvement in energy efficiency with the advancement of hardware speed. We order the machines by non decreasing energy efficiency, i.e., $r < r'$ if $E_r < Er'$.

Each job $j \in J$ represents a compressible inference task that requires f_j^{max} floating-point operations for full execution (i.e., without compression) and must be completed before a deadline d_j (measured in seconds). We order the tasks by non decreasing deadlines, i.e., $i < j$ if $d_i < d_j$. An accuracy function $a_j(f)$ specifies the accuracy reached by job j when dedicating $f < f_j^{max}$ floating operations to it. Let $a_j^{\max} = a_j(f_j^{max})$ be the maximum accuracy reached by a task. This setting naturally describes scenarios involving slimmable networks, such as OFA and AutoSlim [3, 28], where the accuracy of a task depends on its level of compression. Further details regarding accuracy functions are provided in Section 3.1.

A solution for the DSCT-EA problem specifies, for each job j, (i) the machine on which it is executed and (ii) its compression level, i.e, the number f_j of floating-point operations allocated to it.

Below, we formally model DSCT-EA as a minimization problem of an objective function, subject to several constraints.

Variables. We have two sets of decision variables. x_{jr} represents a set of binary variables with a value of 1 if task j is processed on machine r. t_{jr} denotes the set of variables indicating the processing time of task j on machine r. The number of operations dedicated to task j is given by $f_j = \sum_{r \in M} s_r \cdot t_{jr}$.

Model Formulation

$$\min \quad \sum_{j=1}^{n} \left[1 - a_j \left(\sum_{r \in M} s_r \cdot t_{jr} \right) \right] \tag{1a}$$

$$\text{s.t.} \quad \sum_{i=1}^{j} t_{i,r} \le d_j, \qquad j \in J, r \in M \tag{1b}$$

$$t_{jr} \cdot s_r \le f_j^{max}, \qquad j \in J, r \in M \tag{1c}$$

$$t_{jr} \le x_{jr} \cdot d_j, \qquad j \in J, r \in M \tag{1d}$$

$$\sum_{r \in M} x_{jr} = 1, \qquad j \in J \tag{1e}$$

$$\sum_{r \in M} \sum_{j \in J} \frac{s_r}{E_r} \cdot t_{jr} \le B, \tag{1f}$$

$$t_{jr} \in \mathbb{R}^+, \; x_{jr} \in \{0,1\} \qquad j \in J, r \in M \tag{1g}$$

Equation (1a) outlines the objective function, which aims to minimize the accuracy error across all jobs (or equivalently, to maximize the accuracy). Constraint (1b) ensures that we do not exceed the deadline of a job, accounting for the processing time of previously executed jobs on the same machine. Constraint (1c) restricts the total number of floating-point operations dedicated to job j to be no more than f_j^{max}. Constraints (1d) and (1e) guarantee that each job is allocated processing time on a single machine. Finally, constraint (1f) states that the total energy consumed by all machines must not exceed the energy budget B for the system. Here, we consider, in order to evaluate the energy derived from tasks, when some task is processed.

NP Hardness of the problem. DSCT-EA is NP-Complete, as it is an extension of DSCT [5]. This can be shown with a reduction from Sum Partition Problem.

3.1 Accuracy function

We consider accuracy functions that stem from slimmable neural networks [3, 28]. Figure 2 provides an example of such a function, where the accuracy achieved in a ML inference task depends on the model's size processing it, directly linked to the total number of floating-point operations performed. Such accuracy functions are concave, meaning that the derivative with respect to the number of floating-point operations decreases as the number of operations increases. For a task j where f floating operations have been executed, we define the *marginal gain* as the right-hand derivative of the accuracy function $a'^{+}_j(f)$, and the *marginal loss* as the left-hand derivative $a'^{-}_j(f)$. In this work, we will consider piecewise linear functions, where these two derivatives may not always be equal.

Piecewise linear function We consider the special case of linear piecewise functions in the following, as (i) they are a good model for the accuracy function of slimmable models (see Figure 2) and (ii) they can be used to approximate any accuracy functions, for a sufficiently large number of pieces. We use the following notation:

$$a(f) = \left\{ \alpha_k \cdot f + b_k, \quad \text{if } p_k \le f \le p_{k+1}, \qquad \forall k \in \mathcal{K} \right. \tag{2}$$

Here, $\mathcal{K} = \{1, \ldots, K\}$, where K is the number of segments, and α_k and b_k, for $k \in \mathcal{K}$, represent, respectively, the slope and y-intercept point for a linear segment k. The values of p_k and p_{k+1}, called breakpoints, represent the start and the end point for each linear segment k. We have $a(p_{K+1}) = a^{\max}$, and $a(0) = b_1 = a^{\min}$.

3.2 Fractional Relaxation

Here, we introduce a fractional relaxation of the DSCT-EA problem, named Deadline Scheduling with Compressible Tasks - Energy

Aware - Fractional Relaxation (DSCT-EA-FR), in which a task can be executed on multiple machines. We will utilize a solution to this problem as a starting point for our approximation algorithm for DSCT-EA. We give the formulation for the case in which the accuracy functions are piecewise linear.

Piecewise linear formulation. DSCT-EA-FR could be directly reformulated as a linear program by simply altering the type of the variable x_{jr} in the MIP of Section 3 to a real number within the range $[0, 1]$. However, we can further simplify the fractional relaxation by directly omitting the variables x_{jr}, as shown below:

$$\min \quad \sum_{j=1}^{n} -z_j \qquad (Dual) \quad (3a)$$

$$\text{s.t.} \quad z_j \leq \alpha_{jk} \sum_{r \in M} s_r \cdot t_{jr} + b_{jk}, \quad j \in J, k \in K \qquad (\gamma_{jk}) \quad (3b)$$

$$\sum_{i=1}^{j} t_{i,r} \leq d_j, \qquad j \in J, r \in M \qquad (\mu_{jr}) \quad (3c)$$

$$\sum_{r \in M} t_{jr} \cdot s_r \leq f_j^{max}, \qquad j \in J \qquad (\lambda_j) \quad (3d)$$

$$\sum_{r \in M} \sum_{j \in J} \frac{s_r}{E_r} \leq B, \qquad (\beta) \quad (3e)$$

$$t_{jr} \in \mathbb{R}^+, \qquad j \in J, r \in M \qquad (\nu_{jr}) \quad (3f)$$

A new variable z_j is introduced for each task $j \in J$. Constraints (3b) ensures that it is lower than the value of the accuracy function a_j over each segment $k \in K$ for any values of the execution times t_{jr}, $r \in M$. The minimization makes its equal for an optimal solution.

Karush - Kuhn - Tucker Conditions We first discuss the KKT (Karush-Kuhn-Tucker) conditions for the optimization problem and then, we present an algorithm to solve it with polynomial time complexity.

In our convex problem, the KKT conditions are necessary and sufficient conditions for optimality [2] (Chap. 5 p. 226 and 244). First, we write the Lagrangian of the above formulation:

$$L(\boldsymbol{t}, \boldsymbol{\mu}) = \sum_{j=1}^{n} -z_j + \boldsymbol{\gamma}^T \boldsymbol{l}(\boldsymbol{t}) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{t}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{t}) + \boldsymbol{\nu}^T \boldsymbol{i}(\boldsymbol{t}) + \beta j(t),$$

where $\boldsymbol{\gamma} = [\gamma_1 ... \gamma_{nk}]$, $\boldsymbol{l}(\boldsymbol{t}) = [z_j - \alpha_{jk} \sum_{r=1}^{m} s_r \cdot t_{jr} - \beta_{jk}]_{jk} \boldsymbol{\mu} = [\mu_1 ... \mu_{nm}]$, $\boldsymbol{g}(\boldsymbol{t}) = [\sum_{i \leq j} t_{ir} - d_j]_{j,r}$, $\boldsymbol{\lambda} = [\lambda_1 ... \lambda_n]$, $\boldsymbol{h}(\boldsymbol{t}) = [\sum_{r \in M} t_{jr} \cdot s_r - f_j^{max}]_j$, $\boldsymbol{\nu} = [\nu_1 ... \nu_{nm}]$, $\boldsymbol{i}(\boldsymbol{t}) = [-t_{jr}]_{j,r}$ and $j(t) = \sum_{j \in J} \sum_{r \in M} t_{jr} \cdot \frac{s_r}{E_r} - B$.

From the *zero-gradient condition* (i.e., the gradient at the optimum $(\boldsymbol{t^*}, \boldsymbol{\mu^*}, \boldsymbol{\lambda^*}, \boldsymbol{\nu^*}, \boldsymbol{\beta^*})$ should be $\boldsymbol{0}$), we have:

$$\frac{\partial L}{\partial t_{jr}} = -\sum_{k \in K} \alpha_{jk} s_r \gamma_{jk} + \sum_{i=j}^{n} \mu_{ir} + \lambda_j s_r - \nu_{jr} + \beta \frac{s_r}{E_r} = 0, \quad \forall j \in J, r \in M. \tag{4}$$

$$\frac{\partial L}{\partial z_j} = -1 + \sum_{k \in K} \gamma_{jk} = 0, \qquad \forall j \in J. \tag{5}$$

Discussion on Constraints (3b). For each job j, we encounter two scenarios for this set of constraints.

Either the execution time t_j lies *between 2 breakpoints* of the piecewise linear accuracy function or it lies *at a breakpoint*. In the former case, a single constraint (3b) is tight, and its corresponding dual variable is positive ($\gamma_{jl} \geq 0$), while all the dual variables of the other constraints are null. Equation 5 implies that $\gamma_{jl} = 1$. In the latter case, two constraints 3b are tight, and the two consecutive corresponding dual variables are positive ($\gamma_{jl} \geq 0$ and $\gamma_{j,l+1} \geq 0$). The dual variables of all the other constraints are null. Equation 5 implies that $\gamma_{jl} + \gamma_{j,l+1} = 1$. Hereafter, we express the equations using a single dual variable γ_{jl} for a job j (which can be set to 1 when j is not at a breakpoint).

Conditions for job execution times on the same machine r. Consider two consecutive jobs $j < n$ and $j + 1$ on a machine r. Applying Equation 4 on job j gives:

$$\mu_{jr} + \sum_{i=j+1}^{n} \mu_{ir} = \alpha_{jl} s_r \gamma_{jl} + \alpha_{j(l+1)} s_r (1 - \gamma_{jl}) - \lambda_j s_r + \nu_{jr} - \beta \frac{s_r}{E_r}$$

For task $j + 1$:

$$\sum_{i=j+1}^{n} \mu_{ir} = \alpha_{(j+1)l'} s_r \gamma_{j'l'} + \alpha_{(j+1)l'} s_r \gamma j'(l'+1) - \lambda_{j+1} s_r + \nu_{(j+1)r} - \beta \frac{s_r}{E_r}$$

Taking the difference, we obtain:

$$\begin{aligned} \mu_{jr} = &(\alpha_{jl} s_r \gamma_{jl} + \alpha_{j(l+1)} s_r (1 - \gamma_{jl})) - (\alpha_{(j+1)l'} s_r \gamma_{j'l'} \\ &+ \alpha_{(j+1)l'} s_r \gamma j'(l'+1)) - \lambda_j s_r + \lambda_{j+1} s_r + \nu_{jr} - \nu_{(j+1)r}. \end{aligned} \tag{6}$$

If both jobs j and $j + 1$ are not constrained (by f^{max}, by 0, and by their deadlines), the complementary slackness conditions state that $\lambda_j = \lambda_j = \nu_{jr} = \nu_{(j+1)r} = 0$, giving

$$\alpha_{jl} \gamma_{jl} + \alpha_{j(l+1)} (1 - \gamma_{jl}) = \alpha_{(j+1)l'} \gamma_{j'l'} + \alpha_{(j+1)l'} \gamma_{j'(l'+1)} \tag{7}$$

If both jobs are not at a breakpoints (i.e., $\gamma_{j(l+1)} = \gamma_{(j+1)(l'+1)} = 0$), we get

$$\alpha_{jl} = \alpha_{(j+1)l'} \tag{8}$$

It means that two unconstrained consecutive jobs should have the same marginal gains (same slopes). Otherwise, it would have been better to execute the one with the larger slope till its deadline or next breakpoint.

If j and/or $j + 1$ jobs are at a breakpoint, using that $\alpha_{jl} \geq \alpha_{j(l+1)}$ and $\alpha_{(j+1)l} \geq \alpha_{(j+1)(l+1)}$ (concave accuracy functions), it gives that

$$\alpha_{(j+1)(l')} \geq \alpha_{jl} \geq \alpha_{(j+1)(l'+1)} \quad \text{for } j' \text{ at a breakpoint} \tag{9}$$

$$\alpha_{(j+1)(l')} \geq \alpha_{j(l+1)} \text{ and } \alpha_{jl} \geq \alpha_{(j+1)(l'+1)} \text{ when both j and j+1 at a breakpoint} \tag{10}$$

The first inequality ensures that it is not better to decrease t_j and increase t_{j+1}, and the second one that is not better to increase t_j and decrease t_{j+1}. We say here that the marginal gains of j and j' are comparable.

If job j is constrained by its deadline, in this case $\mu_{jr} \geq 0$, because of the non-negativity of dual variables. We obtain

$$\alpha_{jl} \geq \alpha_{(j+1)l'} \quad \text{when } j+1 \text{ is not at a breakpoint and} \tag{11}$$

$$\alpha_{jl} \geq \alpha_{(j+1)(l'+1)} \quad \text{otherwise.} \tag{12}$$

We thus have decreasing marginal gains (decreasing slopes) of jobs (not constrained by their maximum number of operations) on a given machine in an optimal solution. Indeed, otherwise, it would have been better to increase the execution time of job $j + 1$.

Note that, when $j+1$ is constrained by its maximum number of operations ($f_{j+1} = f_{j+1}^{max}$), it may happen that the marginal gain and loss of $j+1$ are higher than the marginal gain of j. Indeed, in this case, the dual variable $\lambda_{j+1} \geq 0$. Equation (6) becomes $\alpha_{(j+1)l'} = \alpha_{jl} s_r + \lambda_{j+1}$, when j and j' are not at breakpoint and j is not constrained by its deadline nor by f_j^{max}. Other cases are dealt with similarly and omitted here due to lack of space.

The Energy Profiles Let us introduce the concept of energy profile, which will be used extensively throughout the rest of the paper. The *energy profile* p_r of a machine $r \in M$ indicates the maximum amount of work (in seconds) that can be done on that machine to respect some energy constraints. In particular, the following conditions must be satisfied:

$$\sum_{j \in J} t_{jr} \leq p_r, \forall r \in M \quad \text{and} \quad \sum_{r \in M} p_r \cdot P_r \leq B.$$

These two conditions state that (i) the amount of work on a machine cannot exceed its profile, and (ii) when all machines are fully used up to their profile, the total energy consumption is at most the energy budget. The energy profile for our system is the collection $p = \{p_1, \cdots, p_m\}$ of the energy profiles of all machines.

We also define the *energy marginal gain and loss* of a task j executed on machine r, whose number of operation f_j is in segment l, as

$$\text{Energy Marginal Gain} = \frac{E_r}{s_r} \frac{\partial^+ a_j}{\partial t_{jr}} = \begin{cases} E_r \alpha_{jl+1} & \text{if } j \text{ at breakpoint} \\ E_r \alpha_{j(l)} & \text{otherwise} \end{cases}$$

$$\text{Energy Marginal Loss} = \frac{E_r}{s_r} \frac{\partial^- a_j}{\partial t_{jr}} = E_r \alpha_{j(l)}.$$

Characteristics of the Energy Profiles. Using again Equation (4) and similar technics, we can show that:

- For machines r and r' such that $E_{r'} \leq E_r$, the *energy profile of the most energy efficient machine should be the longest one.*
- The last jobs executed on machines with not full energy profile should have *equal energy marginal gains* when between breakpoint or *comparable* gain when at breakpoint, meaning that the energy marginal loss of j' (first slope) should be lower than the one of j with the first slope (otherwise, it would have been better to decrease the execution time of j and increase the one of j'), while its energy marginal gain (second slope) should be higher than the one of j' (otherwise, it would have been better to decrease the execution time of j' and increase the one of j).
- The *energy slope product of the last job should be lower on machines with a non full energy profile than on machines with a full energy profile.* The proof is omitted due to lack of space.

4 OPTIMAL ALGORITHM FOR DSCT-EA-FR

In this section, we present an optimal algorithmic solution for DSCT-EA-FR, considering piecewise linear accuracy functions. In particular, we first present an algorithm for solving DSCT-EA-FR when a single machine is used, which is than used as a building block to tackle the multiple-machines scenario.

4.1 Algorithm for DSCT-EA-FR, one machine

Algorithm 1 can be used to optimally solve DSCT-EA-FR on one machine. Its inputs are the deadlines of tasks and a list *listSegments* that stores information about the linear segments of the accuracy functions of our tasks. For each segment we know its *slope*, the *task* to which it is associated, its *position* within the accuracy function (e.g., segment number 1 or segment number 2), the number of operations *totalFlops* needed to fully process it, and *usedFlops*, the number of operations currently dedicated to it by the scheduler.

Note that, although Algorithm 1 does not explicitly take energy into consideration, incorporating an energy budget is feasible by treating it as an additional deadline (plus some additional steps). Further insights into this process will be provided Algorithm 2, where this transformation is explicitly outlined.

The algorithm starts by sorting the segments in non-increasing order of slopes (line 1). Then, for each segment, the algorithms first computes its total processing time based on the value of *totalFlops* and the speed of the machine (line 5), and then adjusts it according to the deadlines of the following tasks (lines 6 - 7). Indeed, increasing too much the processing time for a segment could cause a following task to violate its deadline, which is prevented by line 7. The process is repeated for all the segments. Note that Algorithm 1 always prioritizes segments with higher slope and, for a task $j \in J$, a segment k is considered only if segment $k-1$ (which has a higher slope) was already processed.

Algorithm 1 Exact algorithm for scheduling on one machine using piecewise linear accuracy functions

Require: List of deadlines $[d_1, ..., d_n]$, speed s of the machine, and a list of segments *listSegments*, storing information about the linear segments of the accuracy functions of tasks.
Ensure: List of task execution times $[t_1, ..., t_n]$.

1: *listSegments.order*() ▷ Non-increasing order of slopes
2: **for** $seg \in listSegments$ **do**
3: $j \leftarrow seg.task$
4: $k \leftarrow seg.position$
5: $contribution \leftarrow \frac{seg.totalFlops}{s}$
6: **for** $i > j$ **do** ▷ For all jobs after
7: $contribution \leftarrow min(contribution, d_i - \sum_{k \leq i} t_k)$
8: $t_j \leftarrow t_j + contribution$ ▷ Update processing time of job j
9: **return** $[t_1, ..., t_n]$

Theorem 1. *Algorithm 1 presents a time complexity* $O(n^2)$.

Proof. At line 1, the algorithm sorts the list of segments in $O(n \log(n))$. Assuming a constant number of segments for each job, the algorithm loops over all n jobs at line 2 and, for each of them, it loops over the following ones (at most n) at line 6. This gives a complexity of $O(n^2)$. □

The optimality of Algorithm 1 is shown in the next section, contextually with the proof of optimality of the algorithm for multiple machines (Theorem 2).

4.2 Algorithm for DSCT-EA-FR, multiple machines

After dealing with a single machine, we now propose an algorithm for solving DSCT-EA-FR in the multiple-machines scenario, where energy budget is also taken into consideration.

Intuitively, more energy efficient machines should be preferred when we must respect some energy budget; indeed, for a unit of energy, an efficient machine can execute more floating operations than a less efficient one.

Following this intuition, let us define the *naive energy profile* p^{naive} of a system as the energy profile in which we use the most efficient machines until the energy budget is met.

Although intuition might suggest otherwise, the naive energy profile does not always lead to an optimal solution.

We propose an algorithm which computes the optimal energy profile and solution for DSCT-EA-FR when using multiple machines. It comprises two main steps:

- *ComputeNaiveSolution*: It computes the naive profile and the optimal solution for this profile. This procedure is described in Algorithm 2.
- *RefineProfile*: Check if changing the energy profile could lead to a better solution. If so, update accordingly the energy profile and the solution. Repeat this step until no better solution can be found. This procedure is described in Algorithm 3.

ComputeNaiveSolution Algorithm 2 computes the optimal solution for the naive energy profile.

Firstly, the naive energy profile is derived from the energy budget E_{budget}. This is accomplished by sorting the set of machines M in non-increasing order of energy efficiency E and computing, for each of them, the energy profile by selecting the minimum between the maximal execution time for a machine (d^{max}) and the time needed to reach the energy budget B.

Then, the algorithm transforms the problem into an "equivalent" one on a single-machine problem. This is done by checking for each task and for each machine whether the deadline falls inside the machine profile, in which case the machine can be fully used for the task. Otherwise, only a fraction of the machine can be used to process the job. Next, the algorithm uses Algorithm 1 to compute an optimal solution for this single-machine scenario.

We then need to go back to the multiple-machines scenario. It iterates through the tasks in non-decreasing order of deadlines and checks if it's possible to evenly distribute the tasks across the machines, scheduling the same amount of time for each machine. If energy profile limitations prevent this, the algorithm schedules the maximum allowed time on the least energy-efficient machine and repeats the process with the remaining machines. This procedure is repeated for all tasks.

RefineProfile Once we have the optimal solution for the naive energy profile, Algorithm 3 checks whether a better solution could be found by tweaking the energy profile.

Let us define the *accuracy-per-Joule* $\psi_{seg,r}$ as the accuracy gained per unit of energy by increasing the processing time of a linear segment seg in machine r. Note that this is equivalent to the energy marginal gain discussed in Section 3, as accuracy-per-Joule is also computed as $\psi_{seg,r} = seg.slope \cdot E_r$.

Firstly, Algorithm 3 generates a list of all pairs $(segment, machine)$, sorted in non-increasing order of accuracy-per-Joule. The pairs at the top of the list offer the highest potential for increasing accuracy.

Next, the algorithm iterates over the list and, for each pair (seg, r), it computes how much the execution time of seg could be increased on machine r, keeping track of the additional energy E_{add} that it would consume. To do this, it considers the minimum between the energy required for the remaining floating operations of seg and the energy needed to fill the gap between the tasks that were already scheduled and the deadline of $seg.task$. For each pair analyzed, the algorithm iterates over the list of pairs in reverse order, searching for tasks to be reduced. If a pair (seg', r') is found that presents a smaller accuracy-per-Joule value than (seg, r), it means that it would be beneficial to decrease the execution time of seg' in machine r' and increase that of seg on r. The amount of energy E_{sub} that can be saved by reducing the execution time of seg' is computed, and finally the minimum value between E_{add} and E_{sub} is the actual amount of energy that is "allocated" for seg in machine r, and "deallocated" for seg' in machine r'.

Algorithm 2 `ComputeNaiveSolution`: Computes an optimal scheduling for the naive energy profile

Require: List of segments $listSegments$, the naive energy profile $p^{naive} = [p_1, ..., p_m]$, the speeds $[s_1, ..., s_m]$ and the energy efficiencies $[E_1, ..., E_m]$ of the machines. Deadlines $[d_1, ..., d_n]$ are in non-decreasing order.

$M.order()$ ▷ Order M in non-increasing order of energy efficiency E
$E_{temp} = 0$
for $r \in M$ **do** ▷ Compute naive energy profile
  $p_r \leftarrow \min(\frac{B - E_{temp}}{\frac{s_r}{E_r}}, d^{max})$
  $E_{temp} \leftarrow E_{temp} + p_r \cdot \frac{s_r}{E_r}$
for $j \in J$ **do** ▷ Compute temporary deadlines
  for $r \in M$ **do**
    $d_j^{temp} \leftarrow d_j^{temp} + min(s_r, \frac{p_r \cdot s_r}{d_j}) \cdot d_j$
$t \leftarrow$ Algorithm 1 $(d = d^{temp}, s = 1.0)$ ▷ Computes solution in single machine
$K \leftarrow M$
for $j \in J$ **do** ▷ Distribute the task along machines
  while $t_j \geq 0$ **do**
    $k_{\min} \leftarrow \arg\min_{k \in K}(E_k)$
    if $\sum_{j' \leq j} t_{j'} + \frac{t_j}{\sum_{k \in K} s_k} > p_{k_{min}}$ **then** ▷ Schedule until the machine energy profile
      $t_j \leftarrow t_j - s_{k_{\min}}(p_{k_{\min}} - \sum_{j' \leq j} t_{j'})$
      $t_{jk_{min}} \leftarrow p_{k_{\min}} - \sum_{j' \leq j} t_{j'}$
      $K \leftarrow K \setminus \{k_{\min}\}$
    else ▷ Same time for all machines
      for $k \in K$ **do**
        $t_{jk} \leftarrow \frac{t_j}{\sum_{k \in K} s_k}$
      $t_j \leftarrow 0$
return $[t_{jr} : \forall j \in J, \forall r \in M]$

Theorem 2. *Algorithm 4 is optimal and has a time complexity* $O(n^2m^2)$.

Proof. **Optimality.** At the end of the first phase, computeNaiveSolution, the execution times on all machines satisfy the first set of KKT conditions given by Equations (11), (12) (non increasing

Algorithm 3 `RefineProfile`: Adapts the naive profile in order to find the optimal solution

Require: List of Segments $listSegments$; List of machines speed $\boldsymbol{s}$ and energy efficiency $\boldsymbol{E}$; List of task deadlines $\boldsymbol{d}$

1: $P \leftarrow \{\}$ ▷ Set containing the pairs (segment, machine)
2: **for** $seg \in listSegments$ **do**
3: **for** $r \in M$ **do**
4: $P \leftarrow P + \{(seg, r)\}$
5: $P.order()$ ▷ Order P in non-increasing order of accuracy per Joule
6: **for** $(seg, r) \in P$ **do**
7: $j \leftarrow seg.task$
8: $E_{add} \leftarrow \min(\frac{seg.totalFlops - seg.usedFlops}{E_r}, (d_j - \sum_{k \le j} t_{kr})\frac{s_r}{E_r})$
9: **for** $(seg', r') \in P.reverse()$ **do**
10: **if** $r > r'$ and $\psi_{seg,r} > \psi_{seg',r'}$ **then** ▷ Check accuracy-per-Joule for refining Profile
11: $E_{sub} \leftarrow min(\frac{seg'.flops}{E_{r'}}, t_{j'r'}\frac{s_{r'}}{E_{r'}})$
12: $E_{transf} \leftarrow min(E_{add}, E_{sub})$ ▷ Energy transferred between machines
13: $t_{jr} \leftarrow t_{jr} + \frac{E_{transf} \cdot E_r}{s_r}$
14: $seg.usedFlops \leftarrow seg.usedFlops + E_r \cdot E_{transf}$
15: $t_{j'r'} \leftarrow t_{j'r'} - \frac{E_{transf} \cdot E_{r'}}{s_{r'}}$
16: $seg'.usedFlops \leftarrow seg'.usedFlops + E_{r'} \cdot E_{transf}$
17: $E_{add} \leftarrow E_{add} - E_{transf}$
18: **return** $[t_{jr} : \forall j \in J, \forall r \in M]$

marginal gains on a machine) and Equations (8) and (10) (comparable marginal gain for jobs not constrained by their deadline and maximum execution time). Indeed, the procedure considers the linear pieces of accuracy functions by decreasing slopes. It is thus not possible to have a job with a lower marginal gain and an earlier deadline, which is processed before one with a higher marginal gain and a later deadline.

At the end of the second phase RefineProfile, they satisfy the second set of KKT Conditions stated in the part *Characteristics of the Energy Profiles* of Section 3.2: higher energy marginal gains on more energy efficient machines and comparable energy marginal gain for machines with non full energy profiles. We now argue that the conditions of non-increasing marginal gains are maintained following the RefineProfile phase, specifically after each iteration of the for loop starting at line 5 of RefineProfile. Let's first consider machine r', where we decrease the processing time of a job j' (line 14 of the RefineProfile Algorithm). It's important to note that j' will always be the job on machine r' with the lowest marginal loss. Consider that the number of operations if j', $f_{j'}$, is in the linear segment $k > 1$ of its accuracy function. If we reduce its number of operations, but not up to the breakpoint $p_{j'k}$, it does not change its marginal gain $\alpha_{j'k}$ and loss $\alpha_{j'k}$, so the condition are still satisfied. Now, if we pass the breakpoint $p_{j'k}$, the marginal gain increases (by concavity of the accuracy function) and is equal to $\alpha_{j'(k-1)}$. However, we are insured that it won't become higher than the marginal loss of a later job, say α_{il}, with $i > j'$. Indeed, if we decreased the number of floating operations of j', when it was at $p_{j'k}$, it means that it had the lowest marginal loss $\alpha_{j'(k-1)} \le \alpha_{il}$. So, the KKT conditions (11) and (12) are still satisfied. A similar argument shows that, when increasing the number of floating operations of job j on machine r, we cannot decrease its marginal gain lower than the one the marginal loss of a later job (not constrained by its maximum number of floating operations).

The KKT conditions are necessary and sufficient conditions for optimality as the problem is convex. Thus, the algorithm provides on optimal solution.

Complexity. In `computeNaiveSolution` (Algorithm 2), from lines 6-8, the temporary deadlines are computed, which is done in $O(nm)$. In line 9, according to Theorem 1, it is done in $O(n^2)$. In lines 11-21, the algorithm iterates over every job and each machine.

In `refineProfile`(Algorithm 3), it iterates over each job and each machine and orders the pair list, which is done in $O(nm \cdot log(nm))$. Then, in lines 6-17, we iterate for each pair $(segment, machine)$ in a nested loop. Subsequently, Algorithm 3 has a time complexity $O(n^2m^2)$. Therefore, DSCT-EA-FR-Opt has a time complexity $O(n^2m^2)$. □

Algorithm 4 DSCT-EA-FR-Opt : Computes the optimal solution for DSCT-EA-FR using Piecewise Linear function as accuracy function

Require: List of Segments $listSegments$; List of machines speed $\boldsymbol{s}$ and energy efficiency $\boldsymbol{E}$; List of task deadlines $\boldsymbol{d}$

1: $\boldsymbol{t} \leftarrow$ `computeNaiveSolution()` ▷ Algorithm 2
2: $\boldsymbol{t} \leftarrow$ `refineProfile()` ▷ Algorithm 3
3: **return** $[t_{jr} : \forall j \in J, \forall r \in M]$

5 APPROXIMATION ALGORITHM FOR DSCT-EA

Here, we propose an approximation algorithm Deadline Scheduling with Compressible Tasks - Energy Aware - Approximation Algorithm (DSCT-EA-approx for short), which addresses the DSCT-EA problem. We adapted the algorithm developed in [5]. For addressing the energy budget constraints added to the problem, we set that, for each machine, the energy profile found in the fractional relaxation solution works as an upper bound of the load scheduled in the machine. If the work in the machine is equal to the energy profile, i.e. $\sum_{j \in J} t_j = p_r$, the machine is no longer considered in the scheduling. The pseudocode is described in Algorithm 5.

According to the Theorem 3 in [5], DSCT-EA-approx has an absolute performance guarantee G:

$$OPT - G \le SOL \le OPT \tag{13}$$

Where OPT is the total accuracy for the optimal solution in DSCT-EA-FR and SOL is the total accuracy for the the solution provided by DSCT-EA-approx and the performance guarantee is given by $G = m\int_0^\infty \max_{j,r} \frac{\partial a_j(s_r t)}{\partial t} dt$. . For piecewise linear accuracy function, we have

$$G = m(a^{\max} - a^{\min})\left(1 + \ln\left(\frac{\theta_{\max}}{\theta_{\min}}\right)\right). \tag{14}$$

Where $\theta_{\min} = \min_{j \in J} \alpha_{j0}$ and $\theta_{\max} = \max_{j \in J} \alpha_{j|K|}$.

Algorithm 5 DSCT-EA-APPROX: Approximation algorithm for the scheduling problem on several machines

Input:$[d_1, \ldots, d_n]$, $[t_1^{\max}, ..., t_n^{\max}]$, $[s_1, \ldots, s_m]$, $[E_1, ..., E_m]$
Output: List of task processing times $[t_{jr} : \forall j \in J, r \in M]$

Sort the segments by non increasing order deadlines.
$\boldsymbol{t^f, w_r^{max}}$ = DSCT-EA-FR-OPT($[d_1, \ldots, d_n]$, $[t_1^{\max}, ..., t_n^{\max}]$, $[s_1, \ldots, s_m]$, $[E_1, ..., E_m]$) ▷ Compute the optimal fractional solution $\boldsymbol{t^f}$, e.g. for piecewise, Algorithm 4.
for $r \in M$ **do**
  $S_r \leftarrow []$ ▷ List of tasks scheduled on machine r
  $w_r \leftarrow 0$ ▷ w_r amount of work on r
$F \leftarrow \{\}$ ▷ Set with fully completed machines
for $j \in J$ **do** ▷ Schedule each task on the machine with the least amount of work, $r_{\min}$
  $r_{best} \leftarrow \arg\min_{r \in M \setminus F} w_r$
  $t_{j,r_{best}} \leftarrow min(\sum_{r=1}^{m} t_{j,r}^f, w_r^{max} - w_r)$
  $S_{r_{best}} \leftarrow S_{r_{best}} + [j]$
  if $w_{r_{best}} = w_{r_{best}}^{max}$ **then**
    $F \leftarrow F + [r_{best}]$
for $r \in M$ **do** ▷ Cut tasks violating their deadlines and shift the following ones.
  for $j \in S_r$ **do**
    if $t_{j,r}^{(i)} + t_{jr} \geq d_j$ **then**
      $v_j \leftarrow t_{j,r}^{(i)} + t_{jr} - d_j$
      $t_{j,r} \leftarrow t_{j,r} - v_j$ ▷ Cut task
      for $i > j \in S_r$ **do** ▷ Shift the following tasks
        $t_{i,r}^{(i)} \leftarrow t_{i,r}^{(i)} - v_j$
return $[t_{jr} : \forall j \in J, r \in M]$

6 RESULTS

In this section, we evaluate the approximation algorithm DSCT-EA-APPROX (see Algorithm 5). We first show that the lower bound of Eq. (13) is only reached in very specific scenarios: in most cases, DSCT-EA-APPROX achieves solutions that are almost optimal. We then discuss the execution time of our algorithm and demonstrate that DSCT-EA-APPROX provides results within a reasonable time frame. Next, we compare the performance of our algorithm with state-of-the-art methods that either do not employ compression or utilize only limited discrete levels of compression. We show that it can achieve large energy gains with only a small impact on the task accuracy. Lastly, we consider a scenario with 2 heterogeneous machines to investigate the energy profile and the computational contribution of each machine, i.e., the number of floating operations performed.

Hardware Settings. For the experiments, we used a laptop equipped with an Intel Core i9-12900H CPU and an NVIDIA RTX A2000 GPU.

Experiments. We generate synthetic sets of tasks using the Once-For-All [3] approach applied to ResNet [10], with 1000 classification classes.

We set the minimum task accuracy to $a^{\min} = 1/1000$, the accuracy of a random guess, and the maximum accuracy to $a^{\max} = 0.82$. These values were chosen based on our tests of `ofa-resnet` on the `ImageNet-1k` [6] dataset. We denote as θ_j the "task efficiency" of a task $j \in J$, representing the slope of its first segment. We fix the minimum value of task efficiency to $\theta_{\min} = 0.1$. In our experiments, we modeled the accuracy function of a task j as piecewise linear function, constructed by performing a linear regression with 5 segments over an exponential accuracy function of parameter θ_j (Figure 2 provides an example, although with more than 5 segments).

We considered machine speeds that are uniformly distributed between 1 TFLOPS an 20 TFLOPS, and energy efficiencies uniformly distributed between 5 GFLOPS/W and 60 GFLOPS/W. These values were selected based on research findings presented in [7].

We define the *deadline tolerance level*, denoted by ρ, as $\rho = \frac{m^2 \cdot d_{\max}}{\sum_{j \in J}(f_j^{max} \cdot \sum_{r \in M} s_r)}$. The higher the value of ρ, the more time is allocated for the tasks.

We build scenarios with varying (i) task heterogeneity and (ii) energy budget ratio levels. To this end, we define the *task heterogeneity ratio* μ as $\mu \stackrel{\text{def}}{=} \frac{\theta_{\max}}{\theta_{\min}}$, which reflects the similarity between the accuracy functions of the different tasks. Task efficiencies are uniformly distributed between $\theta_{\min}$ an $\theta_{\max}$, and the value f_j^{max} for each task is computed so to have $a_j(f_j^{max}) = a^{\max}$.

We also define an energy budget ratio β, which describes how strict is the energy budget. Formally, we have $\beta = \frac{B}{\sum_{r \in M} d^{max} \cdot s_r}$, where $d^{max} = \max_{j \in J}(d_j)$. The closest β is to zero, the stricter the energy budget constraint becomes.

Baselines. We benchmark the performance of DSCT-EA-APPROX against 3 different solutions:

- DSCT-EA-UB: an upper bound provided by DSCT-EA-FR-OPT, which solves the fractional relaxation DSCT-EA-FR.
- EDF-NOCOMPRESSION: here, no compression is applied, i.e., tasks are always fully processed, performing f^{max} floating operations. We use the EDF (Earliest Deadline First) strategy combined with scheduling on the machine with the least amount of work [29] to determine where tasks should be scheduled. Scheduling is performed until the energy budget is reached, at which point no further tasks are scheduled.
- EDF-3COMPRESSIONLEVELS: this algorithm considers a discrete number of compression levels for neural networks. Three compression levels are used, corresponding to accuracy levels of 27%, 55%, and 82%. A strategy based on prior research [11] is implemented. Like the previous approach, tasks are scheduled until the energy budget is reached.

Performance guarantees To assess the performance of our algorithm, we analyze the optimality gap for solutions provided by DSCT-EA-APPROX, which indicates the deviation from optimality. We consider a set of $n = 100$ tasks and $m = 5$ machines, with deadline tolerance $\rho = 0.35$ and energy budget ratio $\beta = 0.5$.

We vary the task heterogeneity ratio μ between 5.0 and 20.0, conducting 100 experiments for each value of μ, and collect the mean, maximum, and minimum accuracy values.

Figure 3 illustrates the results of these experiments. It can observed that, on average, we are quite far from reaching the pessimistic lower bound of Eq. 13, which may only be achieved in very specific and rare scenarios.

Execution time analysis We compare the execution time of DSCT-EA-APPROX with that of an optimal solution for DSCT-EA, denoted

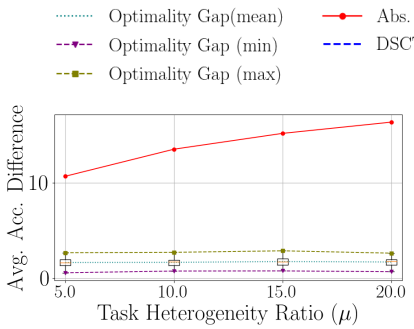
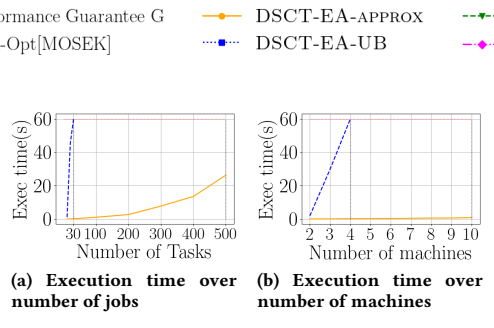


Figure 3: Optimality gap (average accuracy difference between DSCT-EA-UB **and** DSCT-EA-APPROX**) over 100 experiments with varying task heterogeneity.**



(a) Execution time over number of jobs

(b) Execution time over number of machines

Figure 4: Execution times of DSCT-EA-APPROX **vs** DSCT-EA-**Opt [1] for instances with increasing (a) numbers of tasks, with $m = 5$ and (b) number of machines, with $n = 50$.**

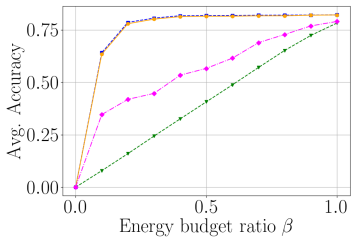


Figure 5: Average task accuracy for DSCT-EA-APPROX **and the baselines over energy budget ratio β, for $m = 2$ and $n = 100$.**

by DSCT-EA-Opt, which is computed using the cvx-MOSEK software [1], a widely used commercial solver for solving Mixed-Integer Programs (MIP). We considered 2 distinct scenarios: (i) when increasing the number of tasks from $n = 10$ to $n = 500$, keeping fixed $m = 5$ machines, and (ii) when increasing the number of machines from $m = 2$ to $m = 10$, fixing $n = 50$ tasks. We considered an average over 10 instances for each experiment analyzed and a time limit of 60s.

The results are shown in Figs. 4a and 4b respectively. We can observe that the solver (DSCT-EA-UB) could handle for only small instances before reaching the time limit (for experiments 1 and 2, respectively, $n = 30$ and $m = 4$). On other hand, our approach, DSCT-EA-APPROX, was capable of managing large instances with hundreds of tasks and several machines.

We also evaluated the execution time of DSCT-EA-FR-OPT against the Mosek solver applied to DSCT-EA-FR, varying the number the tasks from 100 to 500, fixing $m = 5$. The results are described in Table 1. We observe that the developed algorithm DSCT-EA-FR-OPT provides faster results for all instances tested, even with a non-optimized python implementation.

Table 1: Comparison of Execution Times for DSCT-EA-FR-OPT **and** DSCT-EA-FR **[Mosek]**

Number of tasks	**100**	**200**	**300**	**400**	**500**
DSCT-EA-FR-OPT (s)	**1.05**	**2.66**	**7.75**	**13.52**	**26.2**
DSCT-EA-FR [Mosek] (s)	1.11	4.26	11.01	21.23	38.07

Comparison with the State-of-the-Art In this section, we compare the performance of DSCT-EA-APPROX against several state-of-the-art approaches under different energy constraints. We tested DSCT-EA-APPROX against EDF-NOCOMPRESSION, EDF-3COMPRESSIONLEVELS, as well as the optimal solution DSCT-EA-UB.

We varied the energy budget ratio from $\beta = 0.1$ to $\beta = 1.0$ to explore a wide range of energy constraints. Additionally, we considered $n = 100$ tasks, $m = 2$ machines, and a deadline tolerance value $\rho = 1.0$. All tasks were assumed to be uniform with $\theta = 0.1$.

Fig. 5 illustrates the average accuracy under varying energy budget ratios. We can observe that, for β close to zero, the average accuracy is low due to the stringent energy budget. In general, DSCT-EA-APPROX presents a near optimal average accuracy, which almost matches DSCT-EA-UB and clearly outperforms other solutions. For $\beta = 1.0$, the average accuracy for all methods converges to $a^{\max}$, since the energy budget allows the tasks be fully processed.

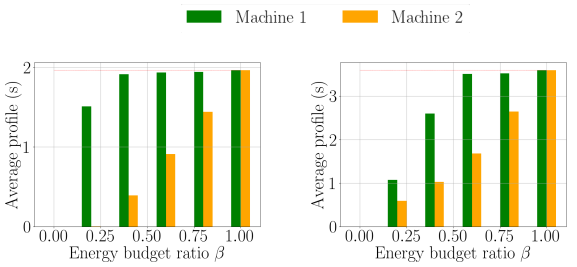
Energy Gain. Note, that, using our solution, DSCT-EA-APPROX, 70% of the energy can be saved up while only reducing by 2% the average task accuracy, compared to a scenario without compression, achieving a high energy sobriety.

Workload Balancing with Heterogeneous Machines and Tasks. Lastly, we explore how DSCT-EA-APPROX balanced the workload across machines with different energy efficiencies and speeds. We consider a simple scenario with $m = 2$, where machine 1 has speed $s_1 = 2$ TFLOPS and energy efficiency $E_1 = 80$ GFLOPS/W, while for machine 2 we have $s_2 = 5$ TFLOPS and $E_2 = 70$ GFLOPS/W. These values were chosen according to [7]. Essentially, machine 1 is slower but more energy efficient than machine 2.

We used $n = 100$ tasks and looked at two 2 distinct scenarios. In the first one (Uniform Tasks), we used a batch of sets with efficiency value θ uniformly distributed between 0.1 and 4.9. In the second scenario (Earliest High Efficient Tasks), we divide tasks into two sets: the earliest 30% of tasks (according to their deadline) have high efficiency ($4.0 \leq \theta \leq 4.9$), while the remaining ones are not very efficient ($0.1 \leq \theta \leq 1.0$). All the experiments used a deadline tolerance value $\rho = 0.01$, indicating very strict deadlines.

Figs. 6a and 6b display the final energy profile computed by DSCT-EA-APPROX for both machines, with varying energy budget ratios β, for scenarios Uniform Tasks and Earliest High Efficient Tasks, respectively. Recall that by energy profile we mean how the workload (in seconds) is distributed across the two machines by DSCT-EA-APPROX, which considers energy constraints during scheduling. Fig. 6a shows that, unsurprisingly, the computed profile for scenario Uniform Tasks is very close to the naive one. The same cannot be said about the Earliest High Efficient Tasks scenario, as Fig. 6b shows a behavior that sensibly deviates from the naive profile.

For $\beta \leq 0.4$, there's a notable difference between the energy profiles of machines 1 and 2 compared to the last deadline d^{max}. This



(a) Uniform Tasks **(b) Earliest High Efficient Tasks**

Figure 6: Energy profile of 2 machines when varying the energy budget ratio. Machine 1 is more energy efficient than machine 2.

difference arises because there's a significant refinement in the profile caused by tasks with high efficiency values θ being constrained by the deadline on machine 1. Consequently, the refinement in the profile increases the workload on machine 2 to accommodate these tasks. For example, for $\beta = 0.4$, the naive profile indicates $p_1^{naive} = 1.96$s and $p_2^{naive} = 0.0$. However, the final profile yields $p_1 = 1.26$s and $p_2 = 0.25$s. Thus, the profile was refined during the optimal algorithm, leading to an increase of the workload of machine 2.

7 CONCLUSION

In this paper, we introduced a novel scheduling system for deep learning inference tasks, incorporating neural network compression, deadlines, and energy constraints. We formulated the problem as a Mixed-Integer Program and developed an exact algorithm for its fractional relaxation and an approximation algorithm for the original problem.

Our evaluations demonstrate that our algorithm outperforms traditional scheduling methods under various energy budgets and nearly achieves optimal performance. Looking ahead, we identify the integration of renewable power sources into the scheduling problem as promising avenues for future research. Moreover, we intend to consider in the problem model the energy consumption resulted from communication of devices.

ACKNOWLEDGMENTS

This work has been supported the French government National Research Agency (ANR) through the UCA JEDI (ANR-15-IDEX-01) EUR DS4H (ANR-17-EURE-004), and ARTIC (ANR-19-CE-25-0001-01) projects, by the France 2030 program under grant agreements No. (ANR-22-PECL-0003 and ANR-22-PEFT-0002), by SmartNet, and by the European Network of Excellence dAIEDGE under Grant Agreement Nr. 101120726.

REFERENCES

[1] MOSEK ApS. 2022. *The MOSEK optimization toolbox for Python manual. Version 10.0.* https://docs.mosek.com/latest/pythonapi/index.html
[2] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization.* Cambridge university press.
[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2020. Once-for-All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations.*
[4] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2018. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine* 35, 1 (2018).
[5] Tiago Da Silva Barros, Frederic Giroire, Ramon Aparicio-Pardo, Stephane Perennes, and Emanuele Natale. 2024. Scheduling with Fully Compressible Tasks: Application to Deep Learning Inference with Neural Network Compression. In *IEEE/ACM International Symposium on Cluster, Cloud, and Internet Computing.*
[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 248–255.
[7] Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. 2023. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems* 38 (2023), 100857.
[8] Biyi Fang, Xiao Zeng, and Mi Zhang. 2018. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking.*
[9] Karl Freund. 2019. Google cloud doubles down on nvidia gpus for inference, 2019. *URL https://www. forbes. com/sites/moorinsights/2019/05/09/google-cloud-doubles-down-on-nvidia-gpus-for-inference* (2019).
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 770–778.
[11] Dayoung Lee and Minseok Song. 2021. Quality-Oriented Task Allocation and Scheduling in Transcoding Servers With Heterogeneous Processors. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 3 (2021), 1667–1680.
[12] C. L. Liu and J. W Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)* 20, 1 (1973), 46–61.
[13] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270* (2018).
[14] Huirong Ma, Rui Li, Xiaoxi Zhang, Zhi Zhou, and Xu Chen. 2023. Reliability-aware online scheduling for dnn inference tasks in mobile edge computing. *IEEE Internet of Things Journal* (2023).
[15] J. McDonald, B. Li, N. Frey, D. Tiwari, V. Gadepally, and S. Samsi. 2022. Great power, great responsibility: Recommendations for reducing energy for training language models. *arXiv preprint arXiv:2205.09646* (2022).
[16] Vinod Nigade, Pablo Bauszat, Henri Bal, and Lin Wang. 2022. Jellyfish: Timely Inference Serving for Dynamic Edge Networks. In *2022 IEEE Real-Time Systems Symposium (RTSS).* IEEE, 277–290.
[17] Suman Raj, Harshil Gupta, and Yogesh Simmhan. 2023. Scheduling dnn inferencing on edge and cloud for personalized uav fleets. In *IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid).*
[18] Mauro Ribeiro, Katarina Grolinger, and Miriam AM Capretz. 2015. Mlaas: Machine learning as a service. In *2015 IEEE 14th international conference on machine learning and applications (ICMLA).* IEEE, 896–902.
[19] T. Si Salem, G. Castellano, G. Neglia, F. Pianese, and A. Araldo. 2021. Towards inference delivery networks: Distributing machine learning with optimality guarantees. In *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet).* IEEE, 1–8.
[20] Dvir Shabtay and Moshe Kaspi. 2006. Parallel machine scheduling with a convex resource consumption function. *European Journal of Operational Research* (2006).
[21] Yu-Chung Tsao, Vo-Van Thanh, and Feng-Jang Hwang. 2020. Energy-efficient single-machine scheduling problem with controllable job processing times under differential electricity pricing. *Resources, Conservation and Recycling* 161 (2020).
[22] RG Vickson. 1980. Two single machine sequencing problems involving controllable job processing times. *AIIE transactions* (1980).
[23] Yidi Wang, Mohsen Karimi, Yecheng Xiang, and Hyoseung Kim. 2021. Balancing energy efficiency and real-time performance in GPU scheduling. In *2021 IEEE Real-Time Systems Symposium (RTSS).* IEEE, 110–122.
[24] Xueqi Wu and Ada Che. 2019. A memetic differential evolution algorithm for energy-efficient parallel machine scheduling. *Omega* 82 (2019), 155–165.
[25] Zichuan Xu, Liqian Zhao, Weifa Liang, Omer F Rana, Pan Zhou, Qiufen Xia, Wenzheng Xu, and Guowei Wu. 2020. Energy-aware inference offloading for DNN-driven applications in mobile edge clouds. *IEEE Transactions on Parallel and Distributed Systems* 32, 4 (2020).
[26] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. 2019. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 7308–7316.
[27] Lvjiang Yin, Xinyu Li, Chao Lu, and Liang Gao. 2016. Energy-efficient scheduling problem using an effective hybrid multi-objective evolutionary algorithm. *Sustainability* 8, 12 (2016), 1268.
[28] Jiahui Yu and Thomas Huang. 2019. AutoSlim: Towards One-Shot Architecture Search for Channel Numbers. https://doi.org/10.48550/ARXIV.1903.11728
[29] Yi-Wen Zhang, Rong-Kun Chen, and Zonghua Gu. 2023. Energy-aware partitioned scheduling of imprecise mixed-criticality systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2023).