Scheduling Compressible Tasks: Application Neural Network Inference

Tiago da Silva Barros, Frédéric Giroire, Ramon Aparicio-Pardo, Stéphane Perennes, Emanuele Natale

Ordonnancement avec des tâches compressibles : Application à l'inférence de réseaux neuronaux[†]

T. S. Barros, F. Giroire, R. Aparicio-Pardo, S. Pérennes, and E. Natale

Université Côte d'Azur/CNRS/I3S/Inria, Sophia Antipolis, France

Avec l'avènement et l'utilisation croissante du Machine Learning as a Service, les systèmes de cloud et de réseau offrent désormais la possibilité de déployer des tâches de ML sur des clusters hétérogènes. Les opérateurs de réseaux et de clouds doivent ensuite programmer ces tâches, en déterminant à la fois quand et sur quels appareils les exécuter. Parallèlement, plusieurs solutions, telles que la compression de réseaux neuronaux, ont été proposées pour construire de petits modèles pouvant fonctionner sur un matériel avec des ressources limitées. Ces solutions permettent de choisir la taille du modèle au moment de l'inférence en fonction du temps de traitement désiré sans avoir à ré-entraîner le réseau. Dans ce travail, nous considérons le problème DSCT (Deadline Scheduling with Compressible Tasks), un *nouveau problème d'ordonnancement* avec deadlines dans lequel les *tâches peuvent être compressées*. Chaque tâche présente un compromis entre son niveau de compression (et donc son temps de traitement) et son utilité. L'objectif est de maximiser l'utilité des tâches. Nous proposons un algorithme d'approximation *avec des garanties prouvées* pour résoudre le problème. Nous validons son efficacité par des simulations approfondies, en obtenant des résultats presque optimaux.

Mots-clefs : scheduling, neural network compression, approximation algorithms, convex programming.

1 Introduction

Nowadays, with the rise of 5G, IoT (Internet of Things), and new hardware capabilities, traditional approaches for deploying cloud services, e.g. Machine Learning (ML) models, have changed. This shift has led to services being deployed along the cloud-to-edge continuum. Then, due to hardware limitations, new model compression techniques have appeared in order to reduce memory and storage usage with a minimal accuracy drop. Cai et al. [CGW+20] introduce an adaptable Deep Neural Network (DNN), which can be adjusted during inference without the need for retraining. Their method involves training DNNs under multiples configurations, e.g. varying numbers of layers. Then, at inference time, the configuration offering the best accuracy while meeting latency requirements can be identified.

In this work, we explore using adjustable neural networks to deploy Deep Learning models in a networked system. Tasks, treated as inference requests, must be scheduled along the cloud-edge continuum before deadlines. Previous research (e.g., [NBBW22]) suggests using pretrained models of varying sizes to optimize latency and accuracy. However, we propose a more advanced approach: allowing machines to compress models to *any compression level*. This flexibility, enabled by the large number of possible configurations (e.g., over 10^{19} for MobileNet), allows for models closely matching targeted processing times. This introduces an optimization challenge.

In this paper, we study the new problem of scheduling a set of *fully compressible tasks* with deadlines. The goal is to decide when, on which machine, and with which compression level, each task should be executed in order to maximize the global utility. The contributions can be summarized as follows:

– We introduce and study the scheduling problem DEADLINE SCHEDULING WITH COMPRESSIBLE TASKS (DSCT) in which tasks can be compressed. The tasks have a utility function expressing their trade-off between processing time and utility.

[†] A long version of this paper has been accepted for publication [dSBGAP+24]

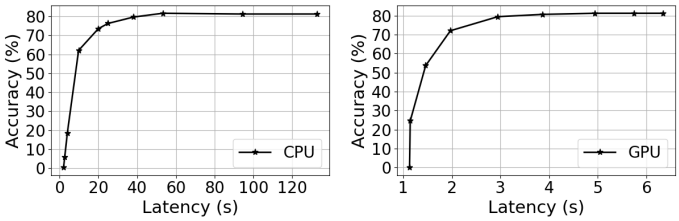


FIGURE 1: Accuracy vs. Latency using the OFA framework for classifying a set of 1000 images from `imagenet-1k`. Experiments on a CPU (left) and GPU (right).

- We carry out experiments to study this trade-off for the OFA [CGW+20] solution.
- We model the accuracy function of large families of compressible tasks, as concave differentiable functions, and we propose convex optimization models and exact algorithms to solve the problem exactly on a single machine.
- We discuss the problem complexity and propose an approximation algorithm with proved guarantees for several heterogeneous machines.
- Finally, the solutions are validated and compared to state-of-the-art solutions with experimentation.

2 Experiments and Latency-accuracy tradeoff analysis

In this section, we derive the latency-accuracy trade off of families of compressible neural networks such as [CGW+20] experimentally. For this, we evaluated the updated `OFA-resnet` in two different devices, one equipped with a CPU (Intel Core i9-12900H), and the other one with a GPU (NVIDIA RTX A2000). Results are shown in Fig. 1 (left) and (right), respectively. The latency and accuracy of the different configurations are plotted. Each point represents the average accuracy reached when classifying a set with 1000 images for a given maximum latency. First, we observe that we obtained a full range for the trade-off. Then, for both CPU and GPU cases, we see that the accuracy function is a concave non decreasing function with a very strong accuracy gains for small latency times and an almost flat curve for large latency times.

3 Modeling and Algorithms

We define the DEADLINE SCHEDULING WITH COMPRESSIBLE TASKS (DSCT) problem, which consists of scheduling a set of n tasks which can be fully compressed on m machines. Formally, we have a set J of tasks. Each task $j \in J$ has a deadline d_j, a maximum processing time $t_j^{\max}$, and a utility function $u_j(t)$ giving its utility when executed for a time t. The goal is to choose the processing time $t_j \in [0, t_j^{\max}]$ for each task $j \in J$ in order to maximize the global utility, i.e., $\sum_{j=0}^{n} u_j(t_j)$, the sum of the utility of all executed tasks. We consider a scenario in which machines have *different speeds* (e.g. servers, laptops, cellphones, IoT devices, etc.). The accuracy function of a task j on a machine r of speed s_r is: $a_{j,r}(t) = a_j(s_r \cdot t)$, with a_j a concave function. We use an exponential accuracy function $a_{j,r}(t) = 1 - e^{-s_r \theta_j t}$, where θ_j is the *task efficiency* , but we believe we can extend for other concave functions family.

We exhibit that the DSCT on a single machine can be modeled by a *convex linear program*, as, for a task, the decision isn't whether to execute it, but rather the selection of its processing time. Using the Karush–Kuhn– Tucker (KKT) conditions, we propose an *exact algorithm* to solve it on a single machine in polynomial time.

We then show that the *problem is NP-hard on multiple machines*. We propose an *approximation algorithm*, DSCT-APPROX, to solve DSCT with a proved guarantee, see Theorem 1. DSCT-APPROX consists in three main steps:

— Step 1 (Fractional relaxation). We first solve the fractional relaxation. In this step, a single task can be assigned to multiple machines. It gives a schedule with maximum accuracy value $OPT_f \geq OPT$ in polynomial time, where OPT is the maximum accuracy of DSCT.

— Step 2 (Rounding). Based on the total processing time computed in the previous step, we consider the tasks by non decreasing deadlines and assign them incrementally to the machine with the least

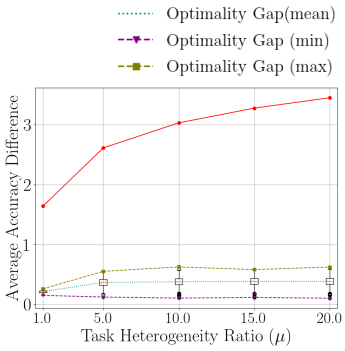
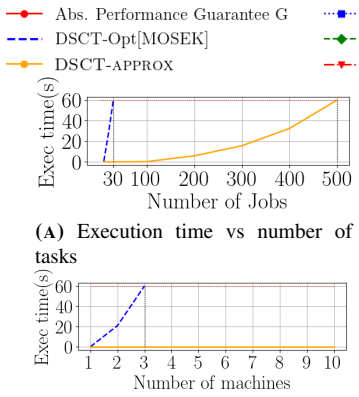


FIGURE 2: Optimality gap (average accuracy difference between DSCT-UB and DSCT-APPROX) over 1000 experiments when varying the task heterogeneity ratio μ. Absolute guarantee G is given as a baseline.



(A) Execution time vs number of tasks

(B) Execution time over number of machines

FIGURE 3: Execution times of DSCT-APPROX vs DSCT-Opt for instances with increasing (a) numbers of tasks and (b) number of machines.

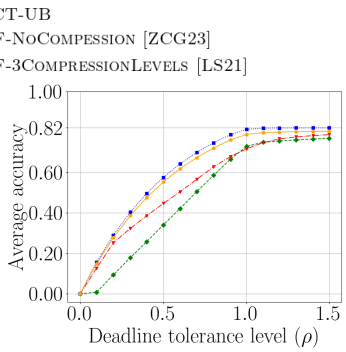


FIGURE 4: Average task accuracy for DSCT-APPROX and the baselines as a function of the deadline tolerance level ρ, for $m = 10$.

amount of work. At the end of this step, each deadline may only be violated by a single task on every machine.

— Step 3 (Incremental shift). For each machine $r \in M$, we consider the tasks in the order of their execution on r. If a task violates the deadline, we cut the amount of time passing the deadline and we shift all the following jobs in machine r.

About the algorithm above, we state the Theorem 1. We denote $\theta_{\max}$ and $\theta_{\min}$ as the maximal and minimal task efficiency, respectively; and $a^{\max}$ and $a^{\min}$ as the maximal and minimal accuracy levels, respectively.

Theorem 1. DSCT-APPROX *is an approximation algorithm with an absolute performance guarantee G:* $OPT - G \leq SOL \leq OPT$, *where OPT is the global accuracy of an optimal solution and SOL is the solution returned by* DSCT-APPROX *and with* $G = mA\left(1 + \frac{1}{e}\ln\left(\frac{\theta_{\max}}{\theta_{\min}}\right)\right)$, *where* $A \stackrel{def}{=} \max_{j\in J}(a_j^{\max} - a_j^{\min})$.

4 Experimental Evaluation

In this section, we evaluate the proposed approximation algorithm.

Experimental scenarios. We build scenarios with different (i) task heterogeneity and (ii) deadline tolerance levels. For (i), we vary the *task heterogeneity ratio* of a set of tasks, denoted as $\mu \stackrel{\text{def}}{=} \frac{\theta_{\max}}{\theta_{\min}}$ which measures the similarity between task efficiencies. For (ii), we define the *deadline tolerance level*, denoted by ρ, as following: $\rho = \frac{m \cdot d_{\max}}{\sum_{j\in J} t_j^{\max}}$., where $d_{\max}$ is the maximal deadline and $t_j^{\max}$ is the maximal processing time for a task $j \in J$. The deadline tolerance level indicates how strict the deadlines are. The greater ρ is the more time is available for scheduling the tasks.

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

- DSCT-UB, an upper bound solution provided by the fractional relaxation of the DSCT problem.
- EDF-NoCompression: Here, compression is not allowed. Then, each task must be executed with its maximal processing time ($t_j^{\max}$), which corresponds to the neural network in its maximal configuration. The scheduling strategy used here is the EDF (Earliest Deadline First) [ZCG23].
- EDF-3CompressionLevels: Here, a discrete set of compression is considered. In this approach, we use 3 levels of compression, corresponding to 3 accuracy levels : 27%, 55% and 82%. For this algorithm, we implemented a strategy based on [LS21]

Study of the approximation guarantee. We first compare the performance of our algorithm to the absolute performance guarantee derived in Theorem 1, i.e. $G = mA(1 + \frac{1}{e}\ln(\mu))$. We thus make μ, the task heterogeneity ratio, vary (from 1 to 20). The guarantee is derived from a worst case analysis.

Fig. 2 presents the distribution over the experiments of two optimality gaps: (i) the *absolute performance guarantee G*, which is actually a *worst-case optimality gap*; and (ii) the *optimality gap* provided by the difference between the fractional relaxation DSCT-UB and the approximation algorithm DSCT-APPROX solutions. We observe that the optimality gaps increase with the task heterogeneity ratio as expected, but the gap with respect to the fractional relaxation solution is well below G. In our experiments, in average, the ratio between the *mean of the optimality gap* and G was 12.36%.

Algorithm processing times. We also tested the execution time of DSCT-APPROX against DSCT-Opt, which uses the cvx-MOSEK software, a widely used commercial solver which allows solving Mixed-Integer Programs (MIP). The results are described in Fig. 3. We considered two scenarios: keeping the number of tasks fixed ($n = 50$), and varying the number of machines and the reverse, fixing $m = 5$. We set the solver's time limit to 60 s. We observe that DSCT-APPROX can handle hundred of jobs and several machines before reaching the time limit. The MOSEK solver, however, can reach a maximal value of 30 jobs and 3 machines.

Comparison with benchmarks for different deadline tolerance levels. We now compare the global accuracy obtained by DSCT-APPROX with DSCT-UB and with the ones reached by EDF-NOCOMPRESSION [ZCG23] and EDF-3COMPRESSIONLEVELS [LS21]. To explore the different possible scenarios of usage, we vary the deadline tolerance level in a range between 0 and 1.5, with a step of 0.1. We observe that DSCT-APPROX presents a performance close to DSCT-UB, providing near-optimal results. Moreover, DSCT-APPROX outperforms the baselines for every deadline tolerance value tested. The difference between DSCT-APPROX and EDF-NOCOMPRESSION (and EDF-3COMPRESSIONLEVELS) performances is significant and reaches more than .18 point (.1) of accuracy, e.g. representing a gain of 57.9% (22.2%) for $\rho = 0.4$.

Fairness. A byproduct of allowing for task compression is that the allocation of processing times over tasks is *fairer* than without compression, in the sense that it allows them to reach similar accuracy values. When not using compression, a task is either executed, and reaches its maximum possible accuracy, or not, leading to a minimum one.

5 Conclusion and Discussion

We studied the problem of scheduling with compressible jobs. We analyzed and modeled the tradeoff between accuracy and processing time of a family of compressible neural networks.

We proposed an approximation algorithm for multiple machines scenario, which is a NP-hard problem. The algorithm proposed obtained a near-optimal value, outperforming the baseline average accuracy by up to 57%. Also, we observed the algorithm presents smaller execution time compared to commercial MIP solvers. Extending our results to a wide range of optimization model, such as energy efficiency and communication latency, is an interesting avenue for future work.

References

[CGW+20] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *ICLR*, 2020.

[dSBGAP+24] T. da Silva Barros, F. Giroire, R. Aparicio-Pardo, S. Pérennes, and E. Natale. Scheduling with fully compressible tasks: Application to deep learning inference with neural network compression. In *ACM/IEEE CCGRID*, 2024.

[LS21] Dayoung Lee and Minseok Song. Quality-oriented task allocation and scheduling in transcoding servers with heterogeneous processors. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1667–1680, 2021.

[NBBW22] Vinod Nigade, Pablo Bauszat, Henri Bal, and Lin Wang. Jellyfish: Timely inference serving for dynamic edge networks. In *IEEE Real-Time Systems Symposium (RTSS)*, 2022.

[ZCG23] Yi-Wen Zhang, Rong-Kun Chen, and Zonghua Gu. Energy-aware partitioned scheduling of imprecise mixed-criticality systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.