Scheduling on Two Unbounded Resources with Communication Costs

Massinissa Ait Aba, Guillaume Aupy, Alix Munier-Kordon

To cite this version:

Massinissa Ait Aba, Guillaume Aupy, Alix Munier-Kordon. Scheduling on Two Unbounded Resources with Communication Costs. [Research Report] RR-9264, Inria. 2019. hal-02076473

HAL Id: hal-02076473

https://hal.inria.fr/hal-02076473

Submitted on 22 Mar 2019

Scheduling on Two Unbounded Resources with Communication Costs

Massinissa Ait Aba, Guillaume Aupy, Alix Munier Kordon

RESEARCH REPORT
N° 9264
March 2019
Project-Team TADaaM

ISSN 0249-6399 ISRN INRIA/RR--9264--FR+ENG

Scheduling on Two Unbounded Resources with Communication Costs

Massinissa Ait Aba [*], Guillaume Aupy[†], Alix Munier Kordon[‡]

Project-Team TADaaM

Research Report n° 9264 — March 2019 — 17 pages

Abstract: Heterogeneous computing systems became a popular and powerful platform, containing several heterogeneous computing elements (e.g. CPU+GPU). In this paper, we consider that we have two platforms, each with an unbounded number of processors. We want to execute an application represented as a Directed acyclic Graph (DAG) using these two platforms. Each task of the application has two possible execution times, depending on the platform it is executed on. Also, there is a cost to transfer data from one platform to another between successive tasks. The goal here is to minimize the finish execution time of the last task of the application (usually called makespan). We show that the problem is NP-complete for graphs of depth at least 3 but polynomial for graphs of depth at most 2. Finally, we focus on particular classes of graphs, by providing polynomial-time algorithms for bi-partite graphs, trees and 2-series-parallel graphs with different assumptions on communication delays.

Key-words: scheduling, DAG, makespan, heterogeneous platform,CPU, GPU

[*] CEA, LIST, Computing and Design Environment Lab.
[†] Inria & Labri, Univ. Bordeaux
[‡] Sorbonne Université, CNRS-UMR 7606 LIP6

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Ordonnancement sur deux ressources illimitées avec coûts de communication

Résumé : Les systèmes de calculs hétérogènes (par exemple CPU+GPU) sont des plateformes populaires. Dans ce travail, nous considérons une machine avec deux plateformes homogènes de calcul, chacune contenant un nombre illimité de ressources de calcul. Nous cherchons à exécuter une application représentée par un graphe de dépendance dirigé et acyclique sur ces plateformes. Chaque tâche de l'application a deux possible modèle d'exécution en fonction de la plateforme sur laquelles elles sont exécutées. En plus nous considérons un coût de communication entre deux tâches successives si elles ne sont pas exécutées sur la même plateforme. Nous travaillons à minimiser le temps d'exécution de l'application.

Nous montrons que le problème est NP-complet pour les graphes de profondeur au moins trois, mais polynomial pour les graphes de profondeur au plus deux. En plus, nous montrons qu'il est possible de calculer des solutions optimales en temps polynomial pour certaines classes de graphes définies récursivement (arbres, graphes série-parallèles).

Mots-clés : ordonnancement, DAG, temps d'exécution, plateforme hétérogène, CPU, GPU

1 Introduction

The past few years have seen an increase demand for developing efficient large computational resources to process large amount of computations related to high performance parallel applications. Thus, High Performance Computers (HPC) became a popular and powerful commercial platform, containing several heterogeneous processing elements. Consequently, more and more attention has been focused on scheduling techniques for solving the problem of optimizing the execution of parallel applications on heterogeneous computing systems.

In this work we revisit the work by Barthou and Jeannot [1]. We consider that we have two platforms, each with an unbounded number of processors. We want to execute an application represented as a Directed Acyclic Graph (DAG) using these two platforms. Each task of the application has two possible execution times, depending on the platform it is executed on. Finally, there is a cost to transfer data from one platform to another one between successive tasks.

In their work, Barthou and Jeannot [1] considered that each task could be executed on both platforms and were able to compute in polynomial time an optimal schedule. Here we study the problem where tasks cannot be re-executed.

While this problem arises more from a theoretical understanding of the process, we can envision several directions linked to the usage of supercomputers (a.k.a High-Performance Computers) where it could be useful.

High-Performance Computers consist of millions of nodes often either homogeneous, or of two types of nodes (e.g. CPU+GPU)[1].

With the increase of computing power, applications are able to generate increasingly more data. However, the bandwidth for the Parallel File System to be able to deal with this data does not increase as fast as the computing power. To deal with this, novel techniques are being implemented (In-Situ/In-Transit analysis [2]) that allow to analyze this data in real-time on separate node. This analysis then informs the main computation (a.k.a *simulation*). Typically, the analysis can be either done on the same nodes as where the simulation is performed (in-situ), or on different nodes that can be more efficient (in-transit), but this implies a cost to transfer data. Note that typically the number of nodes allocated to the simulation is many orders of magnitude higher than the number of analysis task, there are no real restriction on the number of nodes that the analysis pipeline can use.

Another direction where we believe our results can be used is the convergence Big Data-HPC. Several supercomputing centers have started to implement a convergence between Big-Data/Cloud and HPC, where numerous *small* applications are run on a supercomputer [3, 4, 5]. In this context, cloud applications are treated as second class citizen, where they can use the computing power not being used by actual HPC applications. From a typical Cloud-Computing application, the number of nodes needed for each of its task is again many orders of magnitude below that of a supercomputer. Hence one could expect that the main challenge for those jobs will be to determine the type of node that they need (and its property), rather than focusing on the number of available nodes.

Results: Our main contributions are the following. In the model formalized in Section 2, we show that the problem is NP-complete for graphs of depth at least 3 but polynomial for graphs of depth at most 2. We show that the problem cannot be approximated to a factor smaller than 3/2 unless P=NP. Finally, we provide polynomial-time algorithms for several classes of graphs. Those results are presented in Section 3. Finally, after providing some related work in communication-aware scheduling in Section 4, we provide concluding remarks and future directions.

[1] See for example the supercomputers at Argonne National Laboratory https://www.alcf.anl.gov/computing-resources (accessed 09/2018)

2 Model

2.1 Application Model

An application is represented by a Directed Acyclic Graph (DAG) $\mathcal{G} = (V, E)$, such that for all $(v_1, v_2) \in E$, v_2 cannot start its execution before the end of the execution of v_1.

2.2 Platform Model

We consider a parallel platform of two types of machines: machines of type $\mathcal{A}$ and machines of type $\mathcal{B}$. For each type of machine we consider that there are an unbounded number of them.

2.3 Cost and objective functions

We define two cost functions: $t_\mathcal{A} : V \to \mathbb{R}+$ (resp. $t_\mathcal{B} : V \to \mathbb{R}+$) that define the time to execute a task $v \in V$ on a machine of type $\mathcal{A}$ (resp. $\mathcal{B}$).

We also define two communication cost functions: $c_{\mathcal{A}\mathcal{B}} : E \to \mathbb{R}+$ (resp. $c_{\mathcal{B}\mathcal{A}} : E \to \mathbb{R}+$), such that for all $(v_1, v_2) \in E$, if v_1 is scheduled on a machine of type $\mathcal{A}$ (resp. $\mathcal{B}$) and v_2 is scheduled on a machine of type $\mathcal{B}$ (resp. $\mathcal{A}$), then v_2 needs to wait $c_{\mathcal{A}\mathcal{B}}(v_1, v_2)$ (resp. $c_{\mathcal{B}\mathcal{A}}(v_1, v_2)$) units of time after the end of the execution of v_1 to start its execution.

The goal is to find a schedule of each task that minimizes the execution time (or makespan). Because there is an unbounded number of processor of each type, it corresponds to finding an allocation $\sigma : V \to \{\mathcal{A}, \mathcal{B}\}$ of all tasks on each type of processors. The makespan is then obtained by computing the longest path of the graph $\mathcal{G}$ including the corresponding duration of the tasks and the computations costs.

Definition 1 (Length of path)**.** For a path $p = v_1 \to v_2 \to \cdots \to v_p$ of $\mathcal{G}$ and a schedule σ we define $\texttt{len}(p, \sigma)$:

$$\begin{aligned}\texttt{len}(p, \sigma) = {} & t_{\sigma(v_1)}(v_1) + c_{\sigma(v_1)\sigma(v_2)}(v_1, v_2) + t_{\sigma(v_2)}(v_2) + \cdots \\ & + t_{\sigma(v_p)}(v_p)\end{aligned}$$

The following lemma gives the makespan $MS(\mathcal{G}, \sigma)$ of a schedule associated with the allocation σ. It is based on the fact that there is an unbounded number of processors, and that there is no delay due to tasks being conflicted on a shared machine.

Lemma 1.

$$MS(\mathcal{G}, \sigma) = \max_{p \in \{paths\ of\ \mathcal{G}\}} \mathit{len}(p, \sigma)$$

3 Results

In this section, we start by showing that the problem is strongly NP-complete for graph of depth 3, before providing some algorithms for specific graphs.

3.1 Complexity

Theorem 1. *The problem of deciding whether an instance of our main problem has a schedule of length* 2 *is strongly NP-complete even for graphs of depth* 3*.*

We perform the reduction from the 3-Satisfiability (3-SAT) problem which is known to be NP-complete [6]: given $C_1, \cdots, C_m$ be a set of disjunctive clauses where each clause contains exactly three literals over $X = \{x_1, \cdots, x_n\}$ a set of boolean variables. Is there a truth assignment to X such that each clause is satisfied?

In the following, we write each clause $C_i = \tilde{x}_{i_1} \vee \tilde{x}_{i_2} \vee \tilde{x}_{i_3}$ where $(x_{i_1}, x_{i_2}, x_{i_3}) \in X^3$, and $\tilde{x}_k = x_k$ or $\bar{x}_k$. We are looking for a truth assignment such that $\bigwedge_{i=1}^m C_i$ is true.

Proof. From an instance $\mathcal{I}_1$ of 3-SAT: $C_1, \cdots, C_m$ over $\{x_1, \cdots, x_n\}$, we construct the following instance $\mathcal{I}_2$ for our problem.

For all $i \in \{1, \cdots, n\}$, we define 2 tasks v_{ib}^0 and v_i^∞, and an edge (v_i^0, v_i^∞). Then for each clause $C_i = \tilde{x}_{i_1} \vee \tilde{x}_{i_2} \vee \tilde{x}_{i_3}$, 3 tasks $v_{i_1}^i, v_{i_2}^i, v_{i_3}^i$ are created and the following set of edges: $\{(v_{i_1}^i, v_{i_2}^i), (v_{i_2}^i, v_{i_3}^i), (v_{i_1}^i, v_{i_1}^\infty), (v_{i_2}^0, v_{i_2}^i), (v_{i_3}^0, v_{i_3}^i)\}$. For any $j \in \{1, \cdots, n\}$, $v_j^\star$ denotes the set of all the instanciations of x_j in $\mathcal{G}$.

Overall, the graph $\mathcal{G} = (V, E)$ of depth 3 has $2n + 3m$ vertices and $n + 5m$ edges.

We then define the execution and communication costs that can be written in unit size: $\forall j \in \{1, \cdots, n\}$, $t_\mathcal{A}(v_j^\infty) = t_\mathcal{B}(v_j^\infty) = t_\mathcal{A}(v_j^0) = t_\mathcal{B}(v_j^0) = 0$ and $c_{\mathcal{AB}}(v_j^0, v_j^\infty) = c_{\mathcal{BA}}(v_j^0, v_j^\infty) = 3$. For all edges $(v_j^i, v_j^\infty), (v_{j'}^0, v_{j'}^{i'}) \in E$, we add the communication costs $c_{\mathcal{AB}}(v_j^i, v_j^\infty) = c_{\mathcal{BA}}(v_j^i, v_j^\infty) = c_{\mathcal{AB}}(v_{j'}^0, v_{j'}^{i'}) = c_{\mathcal{BA}}(v_{j'}^0, v_{j'}^{i'}) = 3$. Then for $C_i = \tilde{x}_{i_1} \vee \tilde{x}_{i_2} \vee \tilde{x}_{i_3}$ we define the time costs:

$$t_\mathcal{A}(v_{i_j}^i) = 1 - t_\mathcal{B}(v_{i_j}^i) = \begin{cases} 1 & \text{if } \tilde{x}_{i_j} = \bar{x}_{i_j} \\ 0 & \text{if } \tilde{x}_{i_j} = x_{i_j} \end{cases} \tag{1}$$

and we set $c_{\mathcal{AB}}(v_{i_1}^i, v_{i_2}^i) = c_{\mathcal{BA}}(v_{i_1}^i, v_{i_2}^i) = c_{\mathcal{AB}}(v_{i_2}^i, v_{i_3}^i) = c_{\mathcal{BA}}(v_{i_2}^i, v_{i_3}^i) = 0$.

Finally, in the instance $\mathcal{I}_2$, we want to study whether there exists a schedule σ whose makespan is not greater than 2.

We show an example in Figure 1 of the construction of the graph. Here, the clause $C_1 = x_1 \vee \bar{x}_4 \vee x_2$ is associated with the vertices v_1^1, v_4^1 and v_2^1 and the arcs set $\{(v_1^1, v_4^1), (v_4^1, v_2^1), (v_1^1, v_1^\infty), (v_4^0, v_4^1), (v_2^0, v_2^1)\}$. Moreover, $t_\mathcal{A}(v_1^1) = t_\mathcal{A}(v_2^1) = 0$, $t_\mathcal{A}(v_4^1) = 1$, $t_\mathcal{B}(v_1^1) = t_\mathcal{B}(v_2^1) = 1$ and $t_\mathcal{B}(v_4^1) = 0$. Note that $v_1^\star = \{v_1^0, v_1^\infty, v_1^1, v_1^2, v_1^3\}$, $v_2^\star = \{v_2^0, v_2^\infty, v_2^1, v_2^3\}$, $v_3^\star = \{v_3^0, v_3^\infty, v_3^2, v_3^3\}$ and $v_4^\star = \{v_4^0, v_4^\infty, v_4^1, v_4^2\}$.

Next lemmas provide dominance properties on feasible schedules of $\mathcal{I}_2$:

Lemma 2. *Let $\mathcal{S}$ be the set of schedules such that, $\forall \sigma \in \mathcal{S}$, all tasks from $v_j^\star$ are scheduled by the same type of machines,* i.e, *for any couple $(v_j^\alpha, v_j^\beta) \in v_j \times v_j$, $\sigma(v_j^\alpha) = \sigma(v_j^\beta)$. Any feasible solution σ of $\mathcal{I}_2$ belongs to $\mathcal{S}$.*

Proof. Let us suppose by contradiction that a feasible solution $\sigma \notin \mathcal{S}$. Two cases must then be considered:

- If there exists $j \in \{1, \cdots, n\}$ with $\sigma(v_j^0) \neq \sigma(v_j^\infty)$, then there is a communication delay of 3 between them and $\texttt{len}(v_j^0 \to v_j^1, \sigma) = 3$.
- Otherwise, $\forall j \in \{1, \cdots, n\}, \sigma(v_j^0) = \sigma(v_j^\infty)$. Thus, there exists a task v_j^i with $\sigma(v_j^i) \neq \sigma(v_j^0)$. If v_j^i is associated to the first term of the clause C_i, then $(v_j^0, v_j^i) \in E$ and $\texttt{len}(v_j^0 \to v_j^i, \sigma) = 3$. Otherwise, $(v_j^i, v_j^\infty) \in E$ and $\texttt{len}(v_j^i \to v_j^\infty, \sigma) = 3$.

The makespan of σ is at least 3 in both cases, the contradiction. □

Lemma 3. *For any schedule $\sigma \in \mathcal{S}$, $MS(\mathcal{G}, \sigma) = \max_{i \in \{1, \cdots, m\}} \texttt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma)$.*

Proof. To do this, we study the length of paths of $\mathcal{G}$.

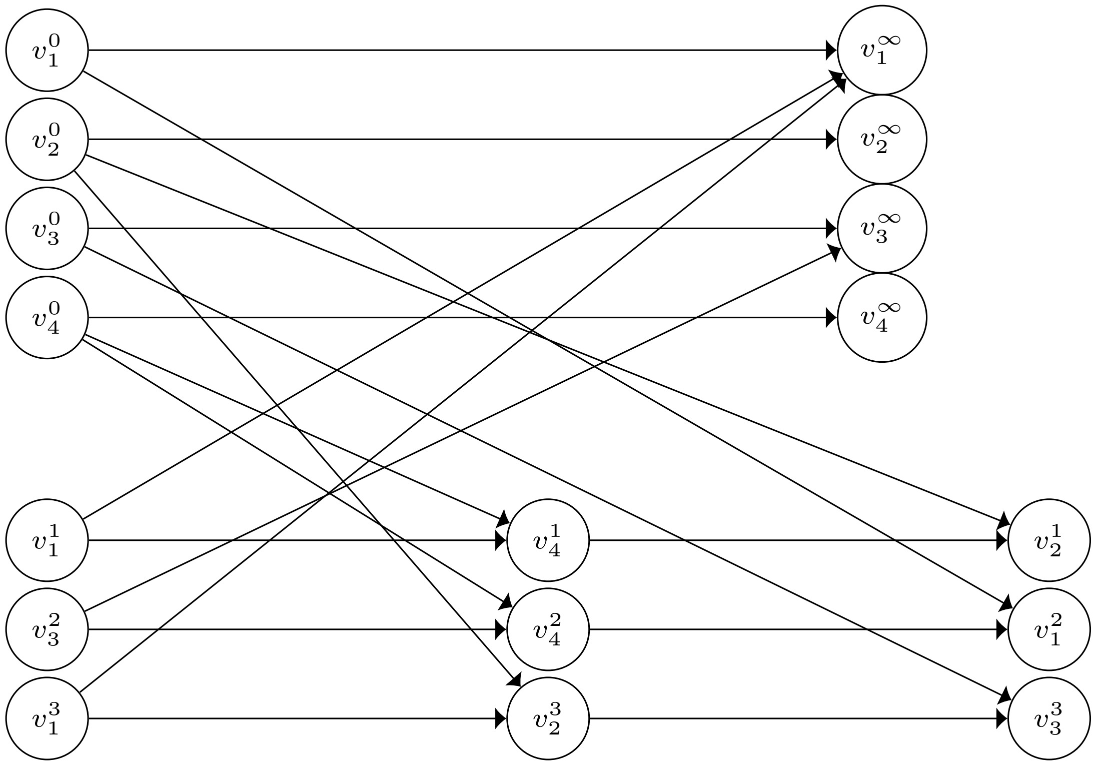


Figure 1: Transformation of $(x_1 \vee \bar{x}_4 \vee x_2) \bigwedge (\bar{x}_3 \vee \bar{x}_4 \vee x_1) \bigwedge (x_1 \vee x_2 \vee x_3)$ ($m = 3$ clauses, $n = 4$ variables) into the associated graph $\mathcal{G} = (V, E)$.

- Let $j \in \{1, \cdots, n\}$, $\texttt{len}(v_j^0 \to v_j^\infty, \sigma) = 0$ since $\sigma(v_j^0) = \sigma(v_j^\infty)$.
- Let $i \in \{1, \cdots, m\}$ associated with the clause $C_i = \tilde{x}_{i_1} \vee \tilde{x}_{i_2} \vee \tilde{x}_{i_3}$:

1. Let consider first the path $v_{i_1}^i \to v_{i_1}^\infty$. By Lemma 2, $\sigma(v_{i_1}^i) = \sigma(v_{i_1}^\infty)$ and thus $c_{\sigma(v_{i_1}^i)\sigma(v_{i_1}^\infty)}(v_{i_1}^i, v_{i_1}^\infty) = 0$. Since $\texttt{len}(v_{i_1}^\infty, \sigma) = 0$,

$$\texttt{len}(v_{i_1}^i \to v_{i_1}^\infty, \sigma) = \texttt{len}(v_{i_1}^i, \sigma) \leq \texttt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma).$$

2. Let consider now the path $v_{i_2}^0 \to v_{i_2}^i \to v_{i_3}^i$. Similarly, $\sigma(v_{i_2}^0) = \sigma(v_{i_2}^i)$ hence

$$\texttt{len}(v_{i_2}^0 \to v_{i_2}^i \to v_{i_3}^i, \sigma) = \texttt{len}(v_{i_2}^i \to v_{i_3}^i, \sigma) \leq \texttt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma).$$

3. Lastly, for the path $(v_{i_3}^0 \to v_{i_3}^i)$, since $\sigma(v_{i_3}^0) = \sigma(v_{i_3}^i)$,

$$\texttt{len}(v_{i_3}^0 \to v_{i_3}^i, \sigma) = \texttt{len}(v_{i_3}^i, \sigma) \leq \texttt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma),$$

which concludes the lemma.

□

Assume that λ is a solution of $\mathcal{I}_1$, let us show that the schedule defined as follow, $\forall j \in \{1, \cdots, n\}$, $\forall v_j^\alpha \in v_j^\star$,

$$\sigma_\lambda : v_j^\alpha \mapsto \begin{cases} \mathcal{A} & \text{if } \lambda(x_j) = 1 \\ \mathcal{B} & \text{if } \lambda(x_j) = 0 \end{cases}$$

has a makespan not greater than 2 and thus is a solution. Following Lemma 3, we must prove that $\forall i \in \{1, \cdots, n\}$, $\mathtt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma_\lambda) \leq 2$.

For any clause $C_i = \tilde{x}_{i_1} \vee \tilde{x}_{i_2} \vee \tilde{x}_{i_3}$, since $\lambda(C_i) = 1$, there exists $j \in \{1, 2, 3\}$ such that $\lambda(\tilde{x}_{i_j}) = 1$. Two cases must be considered:

1. If $\tilde{x}_{i_j} = x_{i_j}$, then by definition $t_\mathcal{A}(v_{i_j}^i) = 0$. Since $\lambda(x_{i_j}) = 1$, $\sigma_\lambda(v_{i_j}^i) = \mathcal{A}$ and thus $\mathtt{len}(v_{i_j}^i, \sigma_\lambda) = t_\mathcal{A}(v_{i_j}^i) = 0$.

2. Otherwise, $\tilde{x}_{i_j} = \bar{x}_{i_j}$ and $t_\mathcal{B}(v_{i_j}^i) = 0$. Now, as $\lambda(x_{i_j}) = 0$, $\sigma_\lambda(v_{i_j}^i) = \mathcal{B}$ and thus $\mathtt{len}(v_{i_j}^i, \sigma_\lambda) = t_\mathcal{B}(v_{i_j}^i) = 0$.

$\mathtt{len}(v_{i_j}^i, \sigma_\lambda) = 0$ in both cases, so $\mathtt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma_\lambda) \leq 2$.

Assume now that we have a solution σ of $\mathcal{I}_2$, let us show that $\lambda_\sigma(x_j) = [\sigma(v_j^\infty) = \mathcal{A}]$ is a solution to $\mathcal{I}_1$.

Following Lemma 2, $\sigma \in \mathcal{S}$. Moreover, for any clause $C_i = \tilde{x}_{i_1} \vee \tilde{x}_{i_2} \vee \tilde{x}_{i_3}$, the corresponding path of $\mathcal{G}$ verifies $\mathtt{len}(v_{i_1}^i \to v_{i_2}^i \to v_{i_3}^i, \sigma) \leq 2$. Thus, there is $j \in \{1, 2, 3\}$ with $\mathtt{len}(v_{i_j}^i, \sigma) = 0$. Two cases must be considered:

1. If $\tilde{x}_{i_j} = x_{i_j}$ then by definition $t_\mathcal{A}(v_{i_j}^i) = 0$ and $t_\mathcal{B}(v_{i_j}^i) = 1$. So, $\sigma(v_{i_j}^i) = \mathcal{A}$ and thus $\lambda_\sigma(x_{i_j}) = 1$.

2. Else, $\tilde{x}_{i_j} = \bar{x}_{i_j}$ and thus $t_\mathcal{A}(v_{i_j}^i) = 1$ and $t_\mathcal{B}(v_{i_j}^i) = 0$. So, $\sigma(v_{i_j}^i) = \mathcal{B}$ and thus $\lambda_\sigma(\bar{x}_{i_j}) = 1$.

So, at least one term of C_i is true following λ_σ, λ_σ is then a solution to $\mathcal{I}_1$.

This concludes the proof that the problem is strongly NP-complete. □

Corollary 1. *There is no polynomial-time algorithm for the problem with a performance bound smaller than $\frac{3}{2}$ unless $\mathcal{P} = \mathcal{NP}$.*

Proof. By contradiction, let us suppose that there exists a polynomial-time algorithm with a performance ratio $\rho < \frac{3}{2}$. This algorithm can be used to decide the existence of a schedule a length at most 2 for any instance $\mathcal{I}$. We deduce that there exists a polynomial time algorithm to decide the existence of a schedule of length strictly less than 3, which contradicts Theorem 1. □

3.2 Polynomial algorithms

Bi-partite graphs We have shown that the problem is NP-hard if the graph has depth 3. The natural question that arises is whether it is already NP-hard for graphs of lower depth. We show that it can be solved in polynomial time for graphs of depth 2 (bipartite graphs).

Theorem 2. BIPARTALGO($\mathcal{G}$) *provides an optimal solution in polynomial time when $\mathcal{G}$ has depth 2.*

The intuition of the proof is that in the case of a biparte graph $\mathcal{G} = (V, E)$, the paths are exactly the edges of $\mathcal{G}$. Hence we measure the makespan of all possible allocations for all edges (four allocations per edge i, j: $(\sigma(i), \sigma(j)) \in \{\mathcal{A}, \mathcal{B}\}^2 = \{(\mathcal{A}, \mathcal{A}), (\mathcal{A}, \mathcal{B}), (\mathcal{B}, \mathcal{A}), (\mathcal{B}, \mathcal{B})\}$). We call this set: `WgPaths`.

$$\begin{aligned}\texttt{WgPaths} = \Big\{ &(\texttt{len}(i \to j, \sigma), i, j, \sigma_i, \sigma_j) \big| \\ &(i, j) \in V, (\sigma(i), \sigma(j)) \in \{\mathcal{A}, \mathcal{B}\}^2, \quad \sigma(i) = \sigma_i, \sigma(j) = \sigma_j \Big\}.\end{aligned} \tag{2}$$

Note that this set can be constructed in polynomial time (see Algorithm 1).

Algorithm 1 Constructing the set `WgPaths`

```
procedure MkWgPaths($V, E$)
    WgPaths $\leftarrow \{\}$
    for $(i, j) \in E$ do
        for $(\sigma_i, \sigma_j) \in \{\mathcal{A}, \mathcal{B}\}^2$ do
            $t_{\sigma_i \sigma_j} \leftarrow t_{\sigma_i}(i) + c_{\sigma_i \sigma_j}(i, j) + t_{\sigma_j}(j)$
                        /*$t_{\sigma_i \sigma_j} = \texttt{len}(i \to j, \sigma)$ when $\sigma(i) = \sigma_i$ and $\sigma(j) = \sigma_j$*/
            WgPaths $\leftarrow$ WgPaths $\cup \left(t_{\sigma_i \sigma_j}, i, j, \sigma_i, \sigma_j\right)$
        end for
    end for
    return WgPaths
end procedure
```

Finally to minimize the makespan, we iteratively remove the allocation that would maximize the makespan. We check that there still exists a possible schedule.

Algorithm 2 Polynomial algorithm for $\mathcal{G} = (V, E)$ a bipartite graph

```
procedure BiPartAlgo($\mathcal{G}$)
    WgPaths $\leftarrow$ MkWgPaths($\mathcal{G}$)
    $P_{\text{alg}} \leftarrow True$; $P_{tmp} \leftarrow True$
                        /* $P_{\texttt{alg}}$ and $P_{tmp}$ are clauses with $n$ variables*/
    for $\left(t_{\sigma_i \sigma_j}, i, j, \sigma_i, \sigma_j\right) \in$ WgPaths, by decreasing value of $t_{\sigma_i \sigma_j}$ do
        $P_{tmp} \leftarrow P \wedge \left([X_i = \mathbb{1}_{\mathcal{B}}(\sigma_i)] \vee [X_j = \mathbb{1}_{\mathcal{B}}(\sigma_j)]\right)$
                        /* where $\mathbb{1}_{\mathcal{B}}(x) = 0$ if $x = \mathcal{A}$, 1 otherwise*/
        if $P_{tmp}$ is satisfiable then
            $P_{\text{alg}} \leftarrow P_{tmp}$
        else
            Break
        end if
    end for
    $X_1, \cdots, X_n \leftarrow$ Solve($P_{\text{alg}}$)
    return $\sigma_\lambda : i \mapsto \lambda(X_i) \cdot \mathcal{A} + (1 - \lambda(X_i)) \cdot \mathcal{B}$
end procedure
```

In what follows, we use the following notation. For a schedule σ and a time D:

$$\begin{aligned}\mathrm{WP}(D) = &\big\{(i,j,\sigma_i,\sigma_j) \text{ s.t. } \big(t_{\sigma_i\sigma_j}, i, j, \sigma_i, \sigma_j\big) \in \texttt{WgPaths} \\ &\text{and } t_{\sigma_i\sigma_j} > D\big\} \\ P_D(\sigma) = &\bigwedge_{(i,j,\sigma_i,\sigma_j)\in\mathrm{WP}(D)} [(\sigma(i) \neq \sigma_i) \vee (\sigma(j) \neq \sigma_j)]\end{aligned}$$

Intuitively, $\mathrm{WP}(D)$ is the set of paths and allocations of length greater than D.

Lemma 4. *Let σ be a schedule of makespan D, then $P_D(\sigma)$ is satisfied.*

This result is a direct consequence of the fact that there should be no path of length greater than D. Hence for $(i,j,\sigma_i,\sigma_j) \in \mathrm{WP}(D)$, we know that we do not have simultaneously in the schedule: $(\sigma(i) = \sigma_i) \wedge (\sigma(j) = \sigma_j)$.

Hence,

$$\begin{aligned}&\neg \bigvee_{(i,j,\sigma_i,\sigma_j)\in\mathrm{WP}(D)} [(\sigma(i) = \sigma_i) \wedge (\sigma(j) = \sigma_j)] \\ &= \bigwedge_{(i,j,\sigma_i,\sigma_j)\in\mathrm{WP}(D)} [(\sigma(i) \neq \sigma_i) \vee (\sigma(j) \neq \sigma_j)] \\ &= P_D(\sigma)\end{aligned}$$

Proof of Theorem 2. Consider an instance $\mathcal{G}$ of the problem. Let D_{alg} be the deadline of the schedule returned by BiPartAlgo($\mathcal{G}$), and P_{alg} be the set of clauses computed by it (line 12).

Let $W_{\mathrm{alg}} = \{(i,j,\sigma_i,\sigma_j) | (t_{\sigma_i\sigma_j}, i, j, \sigma_i, \sigma_j) \in \texttt{WgPaths}\}$ s.t.

$$P_{\mathrm{alg}} = \bigvee_{(i,j,\sigma_i,\sigma_j)\in W_{\mathrm{alg}}} [(\sigma(i) = \sigma_i) \wedge (\sigma(j) = \sigma_j)]\,. \tag{3}$$

Then by construction of P_{alg}, we have the following properties:

1. Forall $\varepsilon > 0$,
$$\mathrm{WP}(D_{\mathrm{alg}}) \subset W_{\mathrm{alg}} \subset \mathrm{WP}(D_{\mathrm{alg}-\varepsilon})$$
This is because we add paths by decreasing the value of makespan (line 4).

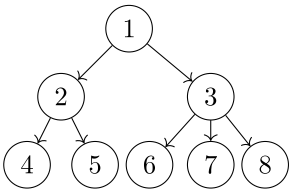
2. There exists $(D_{\mathrm{alg}}, i_0, j_0, \sigma_{i_0}, \sigma_{j_0}) \in \texttt{WgPaths}$ s.t.
$$\begin{aligned}&P_{\mathrm{alg}} && \text{is satisfiable,} \\ &P_{\mathrm{alg}} \bigvee [(\sigma(i_0) = \sigma_{i_0}) \wedge (\sigma(j_0) = \sigma_{j_0})] && \text{is not satisfiable.}\end{aligned}$$
This is the stopping condition on line 9.

We show the optimality of Algorithm 2 by contradiction. If it is not optimal, then $D_{\mathrm{opt}} < D_{\mathrm{alg}}$, and

$$W_{\mathrm{alg}} \cup (i_0, j_0, \sigma_{i_0}, \sigma_{j_0}) \subset \mathrm{WP}(D_{\mathrm{opt}}).$$

Furthermore, according to Lemma 4, $P_{D_{\mathrm{opt}}}(\sigma_{\mathrm{opt}})$ is satisfied, hence σ_{opt} is also a solution to $P_{\mathrm{alg}} \bigvee [(\sigma(i_0) = \sigma_{i_0}) \wedge (\sigma(j_0) = \sigma_{j_0})]$. This contradicts the fact that it does not admit a solution hence contradicting the non optimality.

Finally, the complexity of MkWgPaths(V,E) is $\Theta(|E|)$. In Algorithm 2, we unwind the loop for (line 4) $4|E|$ times, and we verify if P_{tmp} is satisfiable in line 6 with a complexity of $\Theta(n+k)$ where k is the number of clauses is P_{tmp}. Since the number of iterations is bounded by $3|E|$, the complexity of Algorithm 2 is $\mathcal{O}(|E|^2)$. □



$v \in V$	$t_{\mathcal{A}}(v)$	$t_{\mathcal{B}}(v)$
1	2	3
2	3	2
3	4	2
4	5	2
5	3	3
6	5	1
7	2	5
8	3	1

$a \in E$	$c_{\mathcal{AB}}(a)$	$c_{\mathcal{BA}}(a)$
$(3,8)$	2	3
$(3,7)$	3	3
$(3,6)$	3	4
$(2,4)$	2	2
$(2,5)$	3	2
$(1,2)$	1	4
$(1,3)$	4	3

Figure 2: An out-tree $\mathcal{G}$, duration of tasks and communication costs.

Tree-shaped graphs We assume now that the DAG $\mathcal{G} = (V, E)$ is an out-tree rooted by $r \in V$. For any task $u \in V$, the sub-tree rooted by u is the sub-graph $\mathcal{G}_u$ of $\mathcal{G}$ which vertices are the descendants of u.

For any task $u \in V$, let us denote by $D^{\mathcal{A}}(u)$ (resp. $D^{\mathcal{B}}(u)$) the lower bound of the minimal makespan of $\mathcal{G}_u$ assuming that $\sigma(u) = \mathcal{A}$ (resp. $\sigma(u) = \mathcal{B}$). Let us suppose that the arc $(u, v) \in E$. Observe that, if $D^{\mathcal{A}}(v) \le c_{\mathcal{AB}}(u, v) + D^{\mathcal{B}}(v)$, then $D^{\mathcal{A}}(u) \ge t_{\mathcal{A}}(u) + D^{\mathcal{A}}(v)$. In the opposite, $D^{\mathcal{A}}(u) \ge t_{\mathcal{A}}(u) + c_{\mathcal{AB}}(u, v) + D^{\mathcal{B}}(v)$ and thus $D^{\mathcal{A}}(u) \ge t_{\mathcal{A}}(u) + \min(D^{\mathcal{A}}(v), c_{\mathcal{AB}}(u, v) + D^{\mathcal{B}}(v))$. Similarly, $D^{\mathcal{B}}(u) \ge t_{\mathcal{B}}(u) + \min(D^{\mathcal{B}}(v), c_{\mathcal{BA}}(u, v) + D^{\mathcal{A}}(v))$.

For any task $u \in V$, we set $\Gamma^+(u) = \{v \in V, (u, v) \in E\}$. For any allocation function σ, let $\bar\sigma(u) = \mathcal{A}$ if $\sigma(u) = \mathcal{B}$, $\bar\sigma(u) = \mathcal{B}$ otherwise. Then, for any task $u \in V$, we get $D^{\sigma(u)}(u) = t_{\sigma(u)}(u) + \max_{v \in \Gamma^+(u)} \min(D^{\sigma(u)}(v), c_{\sigma(u)\bar\sigma(u)} + D^{\bar\sigma(u)}(v))$.

Theorem 3. *An allocation σ may be built such that the corresponding schedule of length $D(r)$ verifies $D(r) = \min(D^{\mathcal{A}}(r), D^{\mathcal{B}}(r))$ and thus is optimal.*

Proof. Let us suppose that lower bounds $D^{\mathcal{A}}(u)$ and $D^{\mathcal{B}}(u)$ for $u \in V$ are given. Let us define the allocation σ as $\sigma(r) = \mathcal{A}$ if $D^{\mathcal{A}}(r) \le D^{\mathcal{B}}(r)$ and $\sigma(r) = \mathcal{B}$ in the opposite. For any task $v \neq r$ with $(u, v) \in E$, we set $\sigma(v) = \sigma(u)$ if $D^{\sigma(u)}(v) < D^{\bar\sigma(u)}(v) + c_{\sigma(u)\bar\sigma(u)}(u, v)$, and $\sigma(v) = \bar\sigma(u)$ otherwise.

For any task u, we prove that length $D(u)$ of the schedule of $\mathcal{G}_u$ for the allocation σ verifies $D(u) = D^{\sigma(u)}(u)$. If u is a leaf, $D(u) = t_{\sigma(u)}(u) = D^{\sigma(u)}(u)$.

Now, let suppose that $\Gamma^+(u) \neq \emptyset$ and that, for any arc $(u, v) \in E$, if $\sigma(u) = \sigma(v)$, $c_{\sigma(u)\sigma(v)}(u, v) = 0$. Then, if we set $\Delta^\sigma(u, v) = D(v) + c_{\sigma(u)\sigma(v)}(u, v)$, we get by induction $\Delta^\sigma(u, v) = D^{\sigma(v)}(v) + c_{\sigma(u)\sigma(v)}(u, v)$ and by definition of σ, $\Delta^\sigma(u, v) = \min(D^{\sigma(u)}(v), D^{\bar\sigma(u)}(v) + c_{\sigma(u)\bar\sigma(u)}(u, v))$. Now, $D(u) = t_{\sigma(u)}(u) + \max_{v \in \Gamma^+(u)} \Delta^\sigma(u, v)$ and thus by definition of $D^{\sigma(u)}$, $D(u) = D^{\sigma(u)}$, which concludes the proof.

□

A polynomial time algorithm of time complexity $\Theta(n)$ can be deduced by computing first $D^{\mathcal{A}}$, $D^{\mathcal{B}}$ and then σ.

Example. Let us consider as example the out-tree pictured by Figure 2. Figure 3 shows the lower bound $D^{\mathcal{A}}$ and $D^{\mathcal{B}}$ and a corresponding optimal schedule.

Series-Parallel graphs Let us consider a two terminal Series Parallel digraph ($2SP$ in short) as defined in [7, 8]. Each element of this class has a unique source s and a unique sink t with $s \neq t$. It is formally defined as follows where $\mathcal{G}$ and $\mathcal{H}$ are two $2SP$ graphs.

$v \in V$	$D^{\mathcal{A}}(u)$	$D^{\mathcal{B}}(u)$
1	10	10
2	7	5
3	8	7
4	5	2
5	3	3
6	5	1
7	2	5
8	3	1

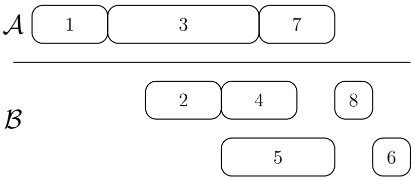


Figure 3: Lower bounds $D^{\mathcal{A}}$ and $D^{\mathcal{B}}$. An optimal schedule is presented for the allocation $\sigma(1) = \mathcal{A}$, $\sigma(2) = \mathcal{B}$, $\sigma(3) = \mathcal{A}$, $\sigma(4) = \mathcal{B}$, $\sigma(5) = \mathcal{B}$, $\sigma(6) = \mathcal{B}$, $\sigma(7) = \mathcal{A}$ and $\sigma(8) = \mathcal{B}$.

- The arc $(s, t) \in 2SP$;
- The series composition of $\mathcal{G}$ and $\mathcal{H}$ is denoted by $\mathcal{G}.\mathcal{H}$ and is built by identifying the sink of $\mathcal{G}$ with the source of $\mathcal{H}$;
- The parallel composition is denoted by $\mathcal{G} + \mathcal{H}$ and identifies respectively the sinks and the sources of the two digraphs.

Figure 4 pictures a $2SP$ graph and its associated decomposition tree.

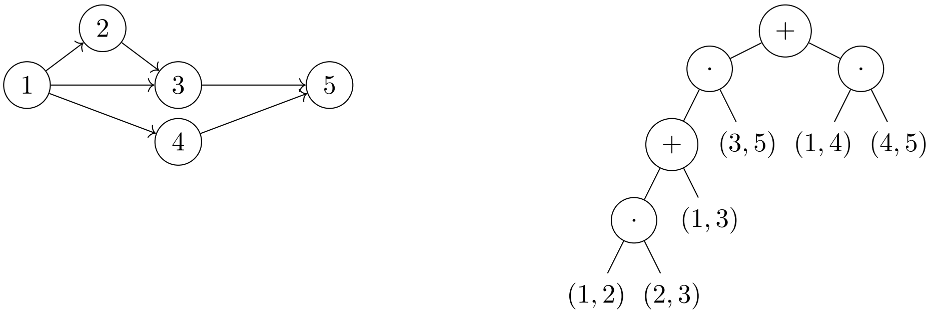


Figure 4: A $2SP$ graph and its associated decomposition tree. Leaves correspond to arcs, while internal nodes are series or parallel compositions.

For any element $\mathcal{G} \in 2SP$ with a source s and a sink t and for any couple $(\alpha, \beta) \in \{\mathcal{A}, \mathcal{B}\}^2$, let us denote by $D^{\alpha\beta}(\mathcal{G})$ a lower bound defined as follows of the minimum length of a schedule of $\mathcal{G}$ with $\sigma(s) = \alpha$ and $\sigma(t) = \beta$. For any graph $\mathcal{G}$ with a unique arc $e = (s, t)$, for any couple $(\alpha, \beta) \in \{\mathcal{A}, \mathcal{B}\}^2$,

$$D^{\alpha\beta}(\mathcal{G}) = \begin{cases} t_\alpha(s) + t_\beta(t) + c_{\alpha\beta}(s, t) & \text{if } \alpha \neq \beta \\ t_\alpha(s) + t_\beta(t) & \text{otherwise.} \end{cases}$$

Now, if $\mathcal{G}$ and $\mathcal{H}$ are two $2SP$, then for the series composition, we set $D^{\alpha\beta}(\mathcal{G}.\mathcal{H}) = \min_{\gamma \in \{\mathcal{A},\mathcal{B}\}}(D^{\alpha\gamma}(\mathcal{G}) + D^{\gamma\beta}(\mathcal{H}) - t_\gamma(t))$ where t is the sink of $\mathcal{G}$. Similarly, for the parallel composition, we set $D^{\alpha\beta}(\mathcal{G} + \mathcal{H}) = \max(D^{\alpha\beta}(\mathcal{G}), D^{\alpha\beta}(\mathcal{H}))$.

We define the allocation function σ associated with a $2SP$ graph $\mathcal{G}$ and the corresponding length $D(\mathcal{G})$ as follows. We set $D(\mathcal{G}) = \min_{(\alpha,\beta) \in \{\mathcal{A},\mathcal{B}\}^2}(D^{\alpha\beta}(\mathcal{G}))$. We also set $\sigma(s)$ and $\sigma(t)$

the allocation function of the source and the sink of $\mathcal{G}$ as $D(\mathcal{G}) = D^{\sigma(s)\sigma(t)}(\mathcal{G})$. Now, for any series composition, let us suppose that s and t (resp. s' and t') are the source and the sink of $\mathcal{G}$ (resp. $\mathcal{H}$). We also suppose that $\sigma(s)$ and $\sigma(t')$ are fixed. Then, for $\mathcal{G}.\mathcal{H}$, $t = s'$ and we get $\sigma(t) = \gamma \in \{\mathcal{A}, \mathcal{B}\}$ such that $D(\mathcal{G}.\mathcal{H}) = D^{\sigma(s)\sigma(t)}(\mathcal{G}) + D^{\sigma(s')\sigma(t')}(\mathcal{H}) - t_{\sigma(t)}(t)$.

If $\mathcal{G}$ is a $2SP$ graph of source s and sink t, any vertex $v \in V - \{s, t\}$ is involved in a series composition, and thus σ is completely defined.

Theorem 4. *For any $2SP$ graph $\mathcal{G}$ of source s and sink t, $D(\mathcal{G}) = D^{\sigma(s)\sigma(t)}(\mathcal{G})$.*

Proof. The equality is clearly true if $\mathcal{G}$ is an arc (s, t). Indeed, we get in this case $D(\mathcal{G}) = \min_{(\alpha,\beta)\in\{\mathcal{A},\mathcal{B}\}^2}(D^{\alpha\beta}(\mathcal{G})) = D^{\sigma(s)\sigma(t)}(\mathcal{G})$.

Now, let us suppose that s and t (resp. s' and t') are the source and the sink of $\mathcal{G}$ (resp. $\mathcal{H}$) and that $D(\mathcal{G}) = D^{\sigma(s)\sigma(t)}(\mathcal{G})$ and $D(\mathcal{H}) = D^{\sigma(s')\sigma(t')}(\mathcal{H})$. For a parallel composition, $D(\mathcal{G} + \mathcal{H}) = \max(D^{\sigma(s)\sigma(t)}(\mathcal{G}), D^{\sigma(s')\sigma(t')}(\mathcal{H})) = D^{\sigma(s)\sigma(t)}(\mathcal{G} + \mathcal{H})$ as $s = s'$ and $t = t'$.

For the series composition, $D(\mathcal{G}.\mathcal{H}) = D(\mathcal{G}) + D(\mathcal{H}) - t_{\sigma(t)}(t) = D^{\sigma(s)\sigma(t)}(\mathcal{G}.\mathcal{H})$, since $t = s'$, which concludes the proof. □

Corollary 2. *A polynomial-time algorithm of time complexity $\Theta(|E|)$ can be deduced by computing lower bounds $D^{\alpha\beta}$, $(\alpha, \beta) \in \{A, B\}^2$ for each graph issued from the decomposition of G and a corresponding allocation σ.*

4 Related Work

We discuss in this section the different works related to scheduling either with two types of machines, with an unbounded number of processors, or/and with communication costs. Note that these topics have been widely studied in the past, so we focus on the work closest to our problem.

Hybrid parallel platforms Different works have considered the problem of hybrid parallel platforms, that is where there are k-types of homogeneous machines, each with a limited number of processors in order to minimize the makespan. This problem is a recurring problem when considering CPU/GPU platforms ($k = 2$). Even with no communication delays, the problem is also NP-hard if the number of processors is limited. For independent tasks, a 2-approximation algorithms have been proposed by Marchal et al. [9] and Kedad et al. [10]. A similar approximation was given by Lenstra et al. [11] for fully heterogeneous platforms with no communication delays. Kedad et al. [12] developed a tight 6-approximation algorithm for general structure graphs on hybrid parallel multi-core machines, composed of CPUs with additional accelerators (GPUs). This work was later revisited by Amaris et al.[13] who showed that by separating the allocation phase and the scheduling phase, they could obtain algorithms with a similar approximation ratio but that performs significantly better in practice. Using a platform with processors of K different speeds, a $2(K + 1)$-approximation algorithm has been developed by Chudak et al. [14].

Finally, Ait Aba et al. [15] have extended the work of Amaris et al.[13] to take into account communications in the case of an hybrid platform. They provide a non polynomial-time two-phases approach with a performance guarantee of 6. The first phase consists in solving the assignment problem to find the type of processor assigned to execute each task (CPU or GPU) using a mixed integer linear program. In the second phase, a list scheduling algorithm has been proposed to generate a feasible schedule using two rounding policies. Note that the same allocation phase can be used for our model, which would reduce the approximation ratio to 2. However because it is not polynomial, for the same complexity one could obtain directly the optimal solution.

Duplications to reduce communications Duplication has been often used in the context of scheduling with communication costs. Indeed, executing a task on more than one processors allows for its successor to be ready sooner by not waiting for the data transfer. Bajai and Agrawal [16] proposed a task duplication based scheduling algorithm for network of heterogeneous systems. This algorithm combines cluster based scheduling and duplication based scheduling to find the optimal solution, provided a set of conditions on task computation and communication time could be satisfied. Kwok and Ahmad [17] wrote a large survey of many algorithms for scheduling in the presence of communication delays with duplication and a limited number of processors.

In the context of an unlimited number of processors, one can only consider a finite number of performance profile (computation cost) and of communication cost for the problem to remain in NP. In the case where all processors have the same processing power and there is a cost for any communication the problem remains NP-complete. However, Darbha and Agrawal [18] provide an optimal solution TDS (Task Duplication based Scheduling) when the communications satisfy some constraints (namely they are not too large compare to the computation costs). This work was later extended by Park and Choe [19] for the opposite case, meaning that the communications are significantly larger than computations.

This work was then studied for hybrid platforms (with two types of machines) by Barthou and Jeannot [1]. They provide a $O(4|E|+2|V|)$ polynomial-time algorithm in this context. They further discuss a possible extension of their work to the case where the number of processors of each type is limited by differentiating the allocation part (using their algorithm) and the scheduling part. [20] has developed a genetic algorithm (GA) approach to the problem of task scheduling for multiprocessor systems.

Communication without duplication While it makes sense to allow duplication when the number of processors is unbounded to reduce the makespan, it may lead to other problem, such as additional energy consumption and significant memory footprint. In these case it may be interesting to study the same problem where duplication is not allowed.

There are many works that study the NP-complete problem of scheduling graphs on parallel platforms with communication work in order to minimize the makespan. The most famous heuristic developed for this problem is HEFT [21] which has no performance guarantee but that performs particularly well. Other heuristics for this problem can be roughly partitioned in two classes: clustering and list scheduling algorithms.

Clustering algorithms [22, 23] usually provide good solutions for communication-intensive graphs by scheduling heavily communicating tasks onto the same processor. After grouping tasks into a set of clusters using different clustering policies, clusters are mapped onto processors using communication sensitive or insensitive heuristics. Note that these solutions cannot be necessarily adapted to our problem because with an infinite number of processors of each type, the risk is to see in the end a single cluster. However in the future it may be an interesting direction to develop efficient algorithms.

List scheduling algorithms [24] are often used to handle a limited number of processors. For our general problem, most of them [25, 26, 27, 28] can be decomposed in two main phases. The first one assigns priorities based on certain task properties, typically run time and/or communication delays. The second phase assigns tasks to processors following priorities.

Experimentally, a comparison of different list scheduling algorithms can be found in the work of Kushwaha and Kumar [28]. Wang and Sinnen [29] provide a wide comparison of clustering and list scheduling algorithms for limited and unlimited number of processors.

We summarize all references in Table 1 depending on the constraint models or platforms.

Table 1: List of works on different applications and platforms to minimize the execution time. We call hybrid a platform with k types of processors where there are no communication costs within a type.

Platform / **Application**	**Unlimited processors**		**Limited processors**	
	Homogeneous processors	**Hybrid platforms**	**Heterogeneous processors**	**Hybrid platforms**
Independent tasks	P (folklore)	P (folklore)	NP-c (folklore) [11]	2-approx [9, 10] $(k = 2)$
Dependant task without communication delays	P (folklore)	P (folklore)	NP-c (folklore) $2(K + 1)-$approx [14]	NP-c (folklore) 6-approx [12, 13] $(k = 2)$
Dependant task with communication delays with duplication	NP-c (general) P (special cases): [18, 19]	P [1] $(k = 2)$	NP-c [20, 16]	NP-c Heuristics [1]
Dependant task with communication delays without duplication	NP-c [23, 30]	[This work]: NP-c P (special cases)	NP-c [22, 28, 25] [26, 27]	NP-c [15]

5 Future directions

With this work we have studied the problem of scheduling a Directed Acyclic Graph on a unbounded hybrid platforms. Specifically our platform consists of two machines, each with an unbounded number of resources. Moving data from one machine to the other one has a communication cost. We have shown the intractability of the problem by reducing this problem to the 3-satisfiability problem. We have shown that there does not exist 3/2-approximation algorithms unless P=NP. We have further provided some polynomial time algorithms for special cases of graphs.

The model presented here can be interesting in the context of high-performance computing where one wants to schedule critical analysis DAGs that need significantly fewer compute resources than the main simulations (*In-Situ/In-Transit analysis*).

There are several extensions that we can see to this work. In the context of two unbounded platforms, it would be interesting to find some polynomial time algorithms with proven bounds to the optimal. We do not expect to be able to find one in the general case, but we hope that with some constraints between the communication costs and computation cost (as is often done in the context of scheduling DAGs with communications), one may able to find such algorithms.

We also plan to study different versions of this problem that make sense in other context. For example in the context of smartphone applications, we can model the frontend/backend context where the phone (Machine 1) has a limited number of available processors, but can rely on sending some of the computation on a backend machine (cloud-based), with an unbounded number of processors. Similarly to here, the problem is a data and communication problem: given the cost to transfer data from one machine to the other one, what is the most efficient strategy.

References

[1] D. Barthou and E. Jeannot, "Spaghetti: Scheduling/placement approach for task-graphs on heterogeneous architecture," in *Euro-Par 2014 Parallel Processing*. Springer International Publishing, 2014, pp. 174–185.

[2] M. Dorier, M. Dreher, T. Peterka, J. M. Wozniak, G. Antoniu, and B. Raffin, "Lessons learned from building in situ coupling frameworks," in *Proceedings of the First Workshop on In Situ Infrastructures for Enabling Extreme-Scale Analysis and Visualization*. ACM, 2015, pp. 19–24.

[3] M. Asch, T. Moore, R. Badia, M. Beck, P. Beckman, T. Bidot, F. Bodin, F. Cappello, A. Choudhary, B. de Supinski *et al.*, "Big data and extreme-scale computing: Pathways to convergence-toward a shaping strategy for a future software and data ecosystem for scientific inquiry," *The International Journal of High Performance Computing Applications*, vol. 32, no. 4, pp. 435–479, 2018.

[4] X. Yang, C. Wu, K. Lu, L. Fang, Y. Zhang, S. Li, G. Guo, and Y. Du, "An interface for biomedical big data processing on the tianhe-2 supercomputer," *Molecules*, vol. 22, no. 12, 2017.

[5] I. Raicu, I. T. Foster, and Y. Zhao, "Many-task computing for grids and supercomputers," in *Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on*. IEEE, 2008, pp. 1–11.

[6] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[7] J. Valdes, R. Tarjan, and E. Lawler, "The recognition of series parallel digraphs," *SIAM Journal on Computing*, vol. 11, no. 2, pp. 298–313, 1982. [Online]. Available: https://doi.org/10.1137/0211023

[8] B. Schoenmakers, *A new algorithm for the recognition of series parallel graphs*, ser. CWI report. CS-R. CWI, 1995.

[9] L. Marchal, L.-C. Canon, and F. Vivien, "Low-cost approximation algorithms for scheduling independent tasks on hybrid platforms," Ph.D. dissertation, Inria-Research Centre Grenoble–Rhône-Alpes, 2017.

[10] S. Kedad-Sidhoum, F. Monna, G. Mounié, and D. Trystram, "A family of scheduling algorithms for hybrid parallel platforms," *International Journal of Foundations of Computer Science*, vol. 29, no. 01, pp. 63–90, 2018.

[11] J. K. Lenstra, D. B. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Mathematical programming*, vol. 46, no. 1-3, pp. 259–271, 1990.

[12] S. Kedad-Sidhoum, F. Monna, and D. Trystram, "Scheduling tasks with precedence constraints on hybrid multi-core machines," in *IPDPSW*. IEEE, 2015, pp. 27–33.

[13] M. Amaris, G. Lucarelli, C. Mommessin, and D. Trystram, "Generic algorithms for scheduling applications on hybrid multi-core machines," in *European Conference on Parallel Processing*. Springer, 2017, pp. 220–231.

[14] F. A. Chudak and D. B. Shmoys, “Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds,” *Journal of Algorithms*, vol. 30, no. 2, pp. 323–343, 1999.

[15] M. Ait Aba, L. Zaourar, and A. Munier, “Approximation algorithm for scheduling applications on hybrid multi-core machines with communications delays,” in *IPDPSW*. IEEE, 2018.

[16] R. Bajaj and D. P. Agrawal, “Improving scheduling of tasks in a heterogeneous environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 2, pp. 107–118, 2004.

[17] Y.-K. Kwok and I. Ahmad, “Static scheduling algorithms for allocating directed task graphs to multiprocessors,” *ACM Computing Surveys (CSUR)*, vol. 31, no. 4, pp. 406–471, 1999.

[18] S. Darbha and D. P. Agrawal, “Optimal scheduling algorithm for distributed-memory machines,” *IEEE transactions on parallel and distributed systems*, vol. 9, no. 1, pp. 87–95, 1998.

[19] C.-I. Park and T.-Y. Choe, “An optimal scheduling algorithm based on task duplication,” in *Parallel and Distributed Systems, 2001. ICPADS 2001. Proceedings. Eighth International Conference on*. IEEE, 2001, pp. 9–14.

[20] A. S. Wu, H. Yu, S. Jin, K.-C. Lin, and G. Schiavone, “An incremental genetic algorithm approach to multiprocessor scheduling,” *IEEE Transactions on parallel and distributed systems*, vol. 15, no. 9, pp. 824–834, 2004.

[21] H. Topcuoglu, S. Hariri, and M.-y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[22] C. Boeres, V. E. Rebello *et al.*, “A cluster-based strategy for scheduling task on heterogeneous processors,” in *Computer Architecture and High Performance Computing, 2004. SBAC-PAD 2004. 16th Symposium on*. IEEE, 2004, pp. 214–221.

[23] T. Yang and A. Gerasoulis, “Dsc: Scheduling parallel tasks on an unbounded number of processors,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 5, no. 9, pp. 951–967, 1994.

[24] M. R. Garey and D. S. Johnson, “Complexity results for multiprocessor scheduling under resource constraints,” *SIAM Journal on Computing*, vol. 4, no. 4, pp. 397–411, 1975.

[25] L. F. Bittencourt, R. Sakellariou, and E. R. Madeira, “Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm,” in *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on*. IEEE, 2010, pp. 27–34.

[26] M. A. Khan, “Scheduling for heterogeneous systems using constrained critical paths,” *Parallel Computing*, vol. 38, no. 4-5, pp. 175–193, 2012.

[27] H. Arabnejad and J. G. Barbosa, “List scheduling algorithm for heterogeneous systems by an optimistic cost table,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 3, pp. 682–694, 2014.

[28] S. Kushwaha and S. Kumar, “An investigation of list heuristic scheduling algorithms for multiprocessor system.” *IUP Journal of Computer Sciences*, vol. 11, no. 2, 2017.

[29] H. Wang and O. Sinnen, "List-scheduling vs. cluster-scheduling," *IEEE Transactions on Parallel and Distributed Systems*, 2018.

[30] T. Yang and A. Gerasoulis, "A fast static scheduling algorithm for dags on an unbounded number of processors," in *Proceedings of the 1991 ACM/IEEE conference on Supercomputing.* ACM, 1991, pp. 633–642.

RESEARCH CENTRE
BORDEAUX – SUD-OUEST

200 avenue de la Vieille Tour
33405 Talence Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399