Polynomial Scheduling Algorithm for Parallel Applications on Hybrid Platforms

Massinissa Ait Aba[1,2(✉)], Lilia Zaourar[1], and Alix Munier[2]

[1] CEA, LIST, Computing and Design Environment Laboratory, 91191 Gif Sur Yvette Cedex, France
massinissa.aitaba@gmail.com
[2] LIP6-UPMC, 4 place Jussieu, 75005 Paris, France

Abstract. This work addresses the problem of scheduling parallel applications into hybrid platforms composed of two different types of resources. We focus on finding a generic approach to schedule applications represented by directed acyclic graphs that minimises makespan with performance guarantee. A three-phase algorithm is proposed; the first two phases consist in solving linear formulations to find the type of processor assigned to execute each task. In the third phase, we compute the start execution time of each task to generate a feasible schedule. Finally, we test our algorithm on a large number of instances. These tests demonstrate that the proposed algorithm achieves a close-to-optimal performance.

Keywords: Scheduling · DAG applications · Makespan · Hybrid platform · CPU · GPU · Approximation algorithm

1 Introduction

Nowadays, High Performance Computers (HPC) are popular and powerful commercial platform due to the increasing demand for developing efficient computing resources to execute large parallel applications. In order to increase the computing power of these platforms while keeping a reasonable level of energy consumption, the heterogeneous platforms have appeared. It is possible to integrate several types of material resources such that each one is specialised for certain types of calculations. Thus we have to take into account that the execution time for any task of the application depends on the type of resource used to execute it. However, using these platforms efficiently became very challenging. 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 [1,2].

This work addresses the problem of scheduling parallel applications onto a particular case of HPC composed of two different types of resources: CPU (Central Processing Unit) and GPU (Graphics Processing Unit). These platforms are

M. Baïou et al. (Eds.): ISCO 2020, LNCS 12176, pp. 143–155, 2020.
https://doi.org/10.1007/978-3-030-53262-8_12

often called hybrid platform. The number of platforms of the *TOP500*[1] equipped with accelerators has significantly increased during the last years. TGCC Curie supercomputer[2] is an example of these platforms.

We focus here in finding a generic approach to schedule applications presented by DAG (Directed Acyclic Graph) into a hybrid platform that minimises the completion time of the application by considering communication delays. An algorithm with three phases has been proposed; the first phase consists in solving a mathematical formulation (P') and then define a new formulation using the solution obtained. The second phase solves an assignment problem to find the type of processor affected to execute the tasks (processing element type 1 or 2) using a linear formulation. In the last phase, we compute the starting execution time of each task to generate a feasible schedule. Our algorithm has been experimented on a large number of instances and evaluated compared to the exact solution.

The rest of the paper is organised as follows: Sect. 2 gives a quick overview of previous research in scheduling strategies on hybrid platforms. Section 3 presents the detailed problem with mathematical formulation. In Sect. 4, we describe the proposed algorithm for our problem and the approximation ratio we obtain for the scheduling problem. Section 5 shows some preliminary numerical results. Finally, we conclude and provide insights for future work in Sect. 6.

2 Related Work

The problem of scheduling tasks on hybrid parallel platforms has attracted a lot of attention. In the case where all processors have the same processing power and there is a cost for any communication ($P|prec, com|C_{max}$), the problem has been shown to be NP-hard [3].

Several works have studied the problem of scheduling independent tasks on ℓ (resp. k) processors of type $\mathcal{A}$ (resp. $\mathcal{B}$) which is represented by $(P\ell, Pk)||C_{max}$. Imreh [4] proves that the greedy algorithm provides a solution with a performance guarantee of $(2 + \frac{\ell-1}{k})$, where $k \leqslant \ell$. Recently, a 2-approximation algorithm has been proposed in [5]. For the same problem, Kedad-Sidhoum et al. [6] proposed two families of approximation algorithms that can achieve an approximation ratio smaller than $(\frac{3}{2}+\epsilon)$. By considering precedence constraints without communication delays $(P\ell, Pk)|prec|C_{max}$, Kedad-Sidhoum et al. [7] developed a tight 6-approximation algorithm for general structure graphs on hybrid parallel multi-core machines. This work was later revisited in [8] 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.

In term of heuristic strategies, the most famous one is Heterogeneous Earliest Finish Time algorithm (HEFT) [9], which is developed for the problem of DAG scheduling on heterogeneous platforms considering communication delays

[1] Top500.org ranking. URL https://www.top500.org/lists/2017/11/.
[2] Tgcc curie supercomputer, http://www-hpc.cea.fr/en/complexe/tgcc-curie.htm.

$(Rm|prec, com|C_{max})$. It could also be applied for hybrid platforms. It has no performance guarantee, but performs particularly well. Other heuristics for this problem can be roughly partitioned into two classes: clustering and list scheduling algorithms.

Clustering algorithms [10,11] 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.

List scheduling algorithms [12] are often used to handle a limited number of processors. Most of them [13,14] 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 a priority list. Experimentally, a comparison of different list scheduling algorithms can be found in the work of Kushwaha and Kumar [14].

Our problem was first treated in [15], a non polynomial-time two-phase approach was proposed with a performance guarantee of 6. Numerical evaluations demonstrate that the proposed algorithm achieves a close-to-optimal performance. However, the running time of this method can be important for large instances. We focus here in finding a polynomial-time approach which is able to maintain an interesting performance with reasonable complexity.

3 Problem Definition

We consider in this work a hybrid platform composed of 2 unrelated Processing elements Pe_1 and Pe_2 (1 CPU and 1 GPU, or 2 different GPUs or CPUs, ...).

An application A of n tasks is represented by a Directed Acyclic Graph (DAG) oriented $G(V, E)$, each vertex represents a task t_i. Each arc $e = (t_i, t_j)$ represents a precedence constraint between two tasks t_i and t_j. We associate it with the value $ct_{i,j}$ which represents the communication delay between t_i and t_j if they are executed on two different resource types. The exact formula to evaluate $ct_{i,j}$ which takes into consideration latencies and available bandwidth between processors is provided in [16]. We denote by $\Gamma^-(i)$ (resp. $\Gamma^+(i)$) the sets of the predecessors (resp. successors) of task t_i. Any task t_i can be executed by both processing elements. Executing the task t_i on Pe_1 (resp. Pe_2) generates an execution time equal to $w_{i,0}$ (resp. $w_{i,1}$). A task t_i can be executed only after the complete execution of its predecessors $\Gamma^-(i)$. We do not allow duplication of tasks and preemption. We denote by C_{max} the completion time of the application A (makespan). The aim is to minimise C_{max}.

Our problem can be modelled by a Mixed Integer formulation (Opt). Let x_i be the decision variable which is equal to 1 if the task t_i is assigned to a Pe_1 and 0 otherwise. Let C_i be the finish time of the task t_i. To manage overlapping tasks on the same processing element, we add an intermediary variable $o_{i,j}$ for each two different tasks t_i and t_j. If t_i and t_j are executed in the same processing element and t_j is executed after the finish execution time of t_i, then $o_{i,j} = 1$,

otherwise $o_{i,j} = 0$. Finally, for each two successive tasks $(t_i, t_k) \in E$, we add an intermediary variable $\zeta_{i,k}$ to manage communication delays.

$$(Opt)\begin{cases} C_i + x_j w_{j,0} + (1 - x_j) w_{j,1} + \zeta_{i,j} ct_{i,j} \leqslant C_j \ \forall (t_i, t_j) \in E & (1) \\ x_i - x_j \leqslant \zeta_{i,j}, \ \forall (t_i, t_j) \in E & (2) \\ x_j - x_i \leqslant \zeta_{i,j}, \ \forall (t_i, t_j) \in E & (3) \\ x_i w_{i,0} + (1 - x_i) w_{i,1} \leqslant C_i, \forall i \in \{1, \ldots, n\}, \Gamma^-(i) = \emptyset & (4) \\ 0 \leqslant C_i \leqslant C_{max}, \forall i \in \{1, \ldots, n\}, \Gamma^+(i) = \emptyset & (5) \\ C_i + x_j w_{j,0} \leqslant C_j + B \times (3 - x_i - x_j - o_{i,j}) \ \forall t_i \neq t_j & (6) \\ C_j + x_i w_{i,0} \leqslant C_i + B \times (2 - x_i - x_j + o_{i,j}) \ \forall t_i \neq t_j & (7) \\ C_i + (1 - x_j) w_{j,1} \leqslant C_j + B \times (1 + x_i + x_j - o_{i,j}) \ \forall t_i \neq t_j & (8) \\ C_j + (1 - x_i) w_{i,1} \leqslant C_i + B \times (x_i + x_j + o_{i,j}) \ \forall t_i \neq t_j & (9) \\ x_i, \ \zeta_{i,j}, \ o_{i,j} \in \{0, 1\}, \ \forall i \in \{1, \ldots, n\}, \ B = Cte & \\ Z(min) = C_{max} & \end{cases}$$

Constraints (1 to 3) describe the critical path, such as if task t_i precedes t_j, and these two tasks are assigned to two different processors, we obtain two cases: either $x_i = 1$ and $x_j = 0$ or $x_i = 0$ and $x_j = 1$. In the two cases, we obtain $\zeta_{i,j} \geqslant 1$. If tasks t_i and t_j are assigned to the same processor, $x_i = 0$ and $x_j = 0$ or $x_i = 1$ and $x_j = 1$. In the two cases, $\zeta_{i,j} \geqslant 0$. Since it is a minimisation problem and without loss of generality, $\zeta_{i,j}$ should take the smallest possible value. Tasks without predecessors (respectively successors) are considered in the constraint (4) (resp. (5)). Overlapping tasks on Pe_1 (resp. Pe_2) is avoided by constraints (6) and (7) (resp. (8) and (9)) by using a large constant B (upper bound for example), such that if two tasks t_i and t_j are executed on the same processor, then either t_i starts after the completion time of the task t_j or t_j starts after the completion time of the task t_i. We have two cases:

1. t_i and t_j are executed on Pe_1, then $x_i = 1$ and $x_j = 1$:

$$\begin{cases} C_i + x_j w_{j,0} \leqslant C_j + B(1 - o_{i,j}) \ (6) \\ C_j + x_i w_{i,0} \leqslant C_i + B(o_{i,j}) \quad (7) \end{cases} \quad \begin{cases} C_i + x_j w_{j,0} \leqslant C_j + B(3 - o_{i,j}) \ (8) \\ C_j + x_i w_{i,0} \leqslant C_i + B(2 + o_{i,j}) \ (9) \end{cases}$$

   If $o_{i,j} = 1$ (resp. $o_{i,j} = 0$), only constraint (6) (resp. (7)) becomes relevant, with $C_i + x_j w_{j,0} \leqslant C_j$ (resp. $C_j + x_i w_{i,0} \leqslant C_i$), then t_j (resp. t_i) starts after the finish execution time of task t_i (resp. t_j). Other constraints will remain valid no matter the execution order of t_i and t_j.
2. t_i and t_j are executed on Pe_2, then $x_i = 0$ and $x_j = 0$:

$$\begin{cases} C_i + x_j w_{j,0} \leqslant C_j + B(3 - o_{i,j}) \ (6) \\ C_j + x_i w_{i,0} \leqslant C_i + B(2 - o_{i,j}) \ (7) \end{cases} \quad \begin{cases} C_i + x_j w_{j,0} \leqslant C_j + B(1 - o_{i,j}) \ (8) \\ C_j + x_i w_{i,0} \leqslant C_i + B(o_{i,j}) \quad (9) \end{cases}$$

   If $o_{i,j} = 1$ (resp. $o_{i,j} = 0$), only constraint (8) (resp. (9)) becomes relevant, with $C_i + (1 - x_j) w_{j,1} \leqslant C_j$ (resp. $C_j + (1 - x_i) w_{i,1} \leqslant C_i$), then t_j (resp. t_i) starts after the finish execution time of task t_i (resp. t_j). Other constraints will remain valid no matter the execution order of t_i and t_j.

The formulation (*Opt*) can be used to obtain an optimal solution for only small instances with limited number of tasks using solvers like *CPLEX* [17]. To solve larger instances, a polynomial method is proposed in the following.

4 Solution Method

In this section, we develop a three-phase algorithm. In Phase 1, we start by proposing a new formulation (P) then we solve its relaxation ($P^{'}$). After that, we use in Phase 2 the solution obtained by this formulation to define another formulation ($P1$). Finally, after rounding the fractional solution of the formulation ($P1$) to obtain a feasible assignment of the tasks, in Phase 3 we use a list scheduling algorithm to find a feasible schedule. Details of each phase are described in the following.

4.1 Phase 1: Formulation (P) and Its Relaxation ($P^{'}$)

We solve here a linear formulation with continuous variables. From the formulation (*Opt*), we define a more simplified formulation (P) which is more useful for the next phase. The first 5 constraints of (*Opt*) are thus taken up again, but the non-overlapping constraints (6) and (7) are replaced by two workload constraints. Thus, (P) is defined as follow, where Constraint (6) (resp. (7)) simply expresses that the makespan should be be larger than the average Pe_1 (resp. Pe_2) workload. The aim is to minimise C_{maxp}.

$$(P)\begin{cases} C_i + x_j w_{j,0} + (1-x_j)w_{j,1} + \zeta_{i,j} ct_{i,j} \leqslant C_j,\ \forall (t_i,t_j) \in E & (1)\\ x_i - x_j \leqslant \zeta_{i,j},\ \forall (t_i,t_j) \in E & (2)\\ x_j - x_i \leqslant \zeta_{i,j},\ \forall (t_i,t_j) \in E & (3)\\ x_i w_{i,0} + (1-x_i)w_{i,1} \leqslant C_i, \forall i \in \{1,\ldots,n\}, \Gamma^-(i) = \emptyset & (4)\\ 0 \leqslant C_i \leqslant C_{maxp}, \forall i \in \{1,\ldots,n\}, \Gamma^+(i) = \emptyset & (5)\\ \sum_{i=1}^{n} x_i w_{i,0} \leqslant C_{maxp} & (6)\\ \sum_{i=1}^{n} (1-x_i) w_{i,1} \leqslant C_{maxp} & (7)\\ x_i, \zeta_{i,j} \in \{0,1\},\ \forall i \in \{1,\ldots,n\}\\ Z(min) = C_{maxp} \end{cases}$$

Remark 1. The optimal solution $C^{\star}_{maxp}$ of this formulation does not take into account non-overlapping constraints, so it represents a lower bound for our problem, $C^{\star}_{maxp} \leqslant C^{\star}_{max}$.

Lemma 1. *For each two successive tasks* $(t_i, t_j) \in E$, *constraints* (2) *and* (3) *can be written as* $max(x_i, x_j) - min(x_i, x_j) \leqslant \zeta_{i,j}$. *Furthermore,* $max(x_i, x_j) - min(x_i, x_j) = (1 - min(x_i, x_j)) + (max(x_i, x_j) - 1) = max(1-x_i, 1-x_j) - min(1-x_i, 1-x_j)$. *Thus, constraints* (2) *and* (3) *can also be written as* $max(1-x_i, 1-x_j) - min(1-x_i, 1-x_j) \leqslant \zeta_{i,j}$.

Remark 2. In each feasible solution of (P), for each couple of tasks $(t_i, t_j) \in E$, we have always $max(x_i, x_j) = 1$ or $max(1-x_i, 1-x_j) = 1$ (or both), $\forall (x_i, x_j) \in \{0,1\} \times \{0,1\}$.

Lemma 2. *In the optimal solution of* (P)*, for each couple of tasks* $(t_i, t_j) \in E$*, from Lemma 1 we have at least* $max(x_i, x_j) = 1$ *or* $max(1-x_i, 1-x_j) = 1$*, such that:*

- *If* $max(x_i, x_j) = 1$*, then constraints* (2) *and* (3) *can be represented by* $\widetilde{Con}^1_{i,j}$: $1 - min(x_i, x_j) \leqslant \zeta_{i,j}$.
- *If* $max(1-x_i, 1-x_j) = 1$*, then constraints* (2) *and* (3) *can be represented by* $\widetilde{Con}^2_{i,j} : 1 - min(1-x_i, 1-x_j) \leqslant \zeta_{i,j}$.

The optimal solution obtained by the formulation (P) without constraints (6) and (7) represents the optimal solution of the scheduling problem on platforms with unlimited resources. This problem has been proven to be NP-hard [18]. Thus, the problem of finding the optimal mapping using (P) is also NP-complete. In order to simplify the problem, we relax the integer variables x_i for $i \in \{1, \ldots, n\}$ and we obtain the relaxed formulation $(P^{'})$. We denote by $\widetilde{x}^{'}_i \in [0,1]$, the value of x_i in the optimal solution of the formulation $(P^{'})$.

4.2 Phase 2: Formulation $(P1)$

Based on Lemma 1 and using the solution of $(P^{'})$, we define another formulation $(P1)$. The decision variables are $x^{'}_i$, and an intermediary variable $y^{'}_{i,j} \in [0,1]$, with $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, n\}$. For all $(t_i, t_j) \in E$, we define the constraint $Con_{i,j}$ as follows:

- If $\min\{\widetilde{x}^{'}_i, \widetilde{x}^{'}_j\} > \min\{1-\widetilde{x}^{'}_i, 1-\widetilde{x}^{'}_j\}$, then $Con_{i,j} = \begin{cases} y^{'}_{i,j} \leqslant x^{'}_i & (1) \\ y^{'}_{i,j} \leqslant x^{'}_j & (2) \\ \zeta^{'}_{i,j} = (1 - y^{'}_{i,j}) & (3) \end{cases}$

  From $Con_{i,j}$, we have $y^{'}_{i,j} \leqslant min\{x^{'}_i, x^{'}_j\}$. Then, $\zeta^{'}_{i,j} = 1 - y^{'}_{i,j} \geqslant (1 - min\{x^{'}_i, x^{'}_j\})$, which is equivalent to constraint $\widetilde{Con}^1_{i,j}$. Since it is a minimisation problem, we can set $\zeta^{'}_{i,j} = (1 - min\{x^{'}_i, x^{'}_j\})$.
- If $\min\{\widetilde{x}^{'}_i, \widetilde{x}^{'}_j\} \leqslant \min\{1-\widetilde{x}^{'}_i, 1-\widetilde{x}^{'}_j\}$ then $Con_{i,j} = \begin{cases} y^{'}_{i,j} \leqslant 1 - x^{'}_i & (1) \\ y^{'}_{i,j} \leqslant 1 - x^{'}_j & (2) \\ \zeta^{'}_{i,j} = (1 - y^{'}_{i,j}) & (3) \end{cases}$

  From $Con_{i,j}$, we have $y^{'}_{i,j} \leqslant min\{1-x^{'}_i, 1-x^{'}_j\}$. Then, $\zeta^{'}_{i,j} = 1 - y^{'}_{i,j} \geqslant (1 - min\{1-x^{'}_i, 1-x^{'}_j\})$, which is equivalent to constraint $\widetilde{Con}^2_{i,j}$. Since it is a minimisation problem, we can set $\zeta^{'}_{i,j} = (1 - min\{1-x^{'}_i, 1-x^{'}_j\})$.

The formulation $(P1)$ is then given by:

$$(P1)\begin{cases} C_i^{'} + x_j^{'} w_{j,0} + (1 - x_j^{'}) w_{j,1} + \zeta_{i,j}^{'} ct_{i,j} \leqslant C_j^{'}, \forall (t_i, t_j) \in E & (1) \\ Con_{i,j}, \forall (t_i, t_j) \in E & (2) \\ x_i^{'} w_{i,0} + (1 - x_i^{'}) w_{i,0} \leqslant C_i^{'}, \forall i \in \{1, \ldots, n\}, \Gamma^{-}(i) = \emptyset & (3) \\ 0 \leqslant C_i^{'} \leqslant C_{max1'}, \forall i \in \{1, \ldots, n\}, \Gamma^{+}(i) = \emptyset & (4) \\ \sum_{i=1}^{n} x_i^{'} w_{i,0} \leqslant C_{max1'} & (5) \\ \sum_{i=1}^{n} (1 - x_i^{'}) w_{i,0} \leqslant C_{max1'} & (6) \\ x_i^{'}, y_{i,j}^{'}, \zeta_{i,j}^{'} \in [0,1], \ \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, n\} & \\ Z(min) = C_{max1'} & \end{cases}$$

We can notice that constraints (1, 3, 4, 5, 6) of the formulation $(P1)$ are equivalent to constraints (1, 4, 5, 6, 7) of the formulation $(P^{'})$. We denote by $C_{max1'}^{\star}$ the optimal solution of the formulation $(P1)$ and $C_{maxp'}^{\star}$ the optimal solution of $(P^{'})$.

Theorem 1. *If the optimal solution $\widetilde{x}_i^{\star}$ obtained by $(P^{'})$ is an integer for all $i \in \{1, \ldots, n\}$, then $C_{max1'}^{\star} = C_{maxp'}^{\star}$.*

Proof. By setting the value of $x_i^{'} = \widetilde{x}_i^{\star}$, for all $i \in \{1, \ldots, n\}$, then for each two successive tasks $(t_i, t_j) \in E$, we have two cases:

1. $\min\{\widetilde{x}_i^{\star}, \widetilde{x}_j^{\star}\} > \min\{1 - \widetilde{x}_i^{\star}, 1 - \widetilde{x}_j^{\star}\}$, then $Con_{i,j} = \begin{cases} y_{i,j}^{'} \leqslant x_i^{'} & (1) \\ y_{i,j}^{'} \leqslant x_j^{'} & (2) \\ \zeta_{i,j}^{'} = (1 - y_{i,j}^{'}) & (3) \end{cases}$

   Furthermore, $\min\{\widetilde{x}_i^{\star}, \widetilde{x}_j^{\star}\} = 1$, then $\widetilde{x}_i^{\star} = 1$ and $\widetilde{x}_j^{\star} = 1$, follows $\widetilde{x}_i^{\star} - \widetilde{x}_j^{\star} = 0 \leqslant \widetilde{\zeta}_{i,j}^{\star}$ ($\widetilde{\zeta}_{i,j}^{\star} = 0$ since it is a minimisation problem). Furthermore, $\zeta_{i,j}^{'} = (1 - min\{x_i^{'}, x_j^{'}\}) = 1 - 1 = 0$, then $\zeta_{i,j}^{'} = \widetilde{\zeta}_{i,j}^{\star}$.

2. $\min\{\widetilde{x}_i^{\star}, \widetilde{x}_j^{\star}\} \leqslant \min\{1 - \widetilde{x}_i^{\star}, 1 - \widetilde{x}_j^{\star}\}$, then $Con_{i,j} = \begin{cases} y_{i,j}^{'} \leqslant 1 - x_i^{'} & (1) \\ y_{i,j}^{'} \leqslant 1 - x_j^{'} & (2) \\ \zeta_{i,j}^{'} = (1 - y_{i,j}^{'}) & (3) \end{cases}$

   – if $\min\{1 - \widetilde{x}_i^{\star}, 1 - \widetilde{x}_j^{\star}\} = 1$, then $1 - \widetilde{x}_i^{\star} = 1$ and $1 - \widetilde{x}_j^{\star} = 1$, follows $\widetilde{x}_i^{\star} - \widetilde{x}_j^{\star} = 0 \leqslant \widetilde{\zeta}_{i,j}^{\star}$ with $\widetilde{x}_i^{\star} = 0$ and $\widetilde{x}_j^{\star} = 0$. Furthermore, $\zeta_{i,j}^{'} = (1 - min\{1 - x_i^{'}, 1 - x_j^{'}\}) = 1 - 1 = 0$.

   – if $\min\{1 - \widetilde{x}_i^{\star}, 1 - \widetilde{x}_j^{\star}\} = 0$, then we suppose that $\widetilde{x}_i^{\star} = 1$ and $\widetilde{x}_j^{\star} = 0$, follows $\widetilde{x}_i^{\star} - \widetilde{x}_j^{\star} = 1 \leqslant \widetilde{\zeta}_{i,j}^{\star}$. Furthermore, $\zeta_{i,j}^{'} = (1 - min\{1 - x_i^{'}, 1 - x_j^{'}\}) = 1 - 0 = 1$.

   In both cases, we have $\zeta_{i,j}^{'} = \widetilde{\zeta}_{i,j}^{\star}$.

Thus, the finish execution time of each task in the formulation $(P^{'})$ is the same in $(P1)$. Furthermore, since $\sum_{i=1}^{n} x_i^{'} w_{i,0} = \sum_{i=1}^{n} x_i^{\star} w_{i,0}$ and $\sum_{i=1}^{n} (1 - x_i^{'}) w_{i,1} = \sum_{i=1}^{n} (1 - x_i^{\star}) w_{i,1}$, then constraints (5) and (6) of the formulation $(P^{'})$ are the same in $(P1)$. Finally, $C_{max1'}^{\star} = C_{maxp'}^{\star}$.

However, finding the ratio between $C_{max1'}^{\star}$ and $C_{maxp'}^{\star}$ for the general case is difficult. In the following, we suppose that $C_{max1'}^{\star} \leqslant \alpha C_{maxp'}^{\star}$, with $\alpha \in \mathbb{R}^{+}$.

Table 1 shows the standard deviation between $C^{\star}_{max1'}$ and $C^{\star}_{maxp'}$ for 20 randomly generated instances of different sizes (DAG graphs). For each instance I_i, we compute $\alpha_i = \frac{C^{\star}_{max1'}(I_i)}{C^{\star}_{maxp'}(I_i)}$. Then, we calculate **Average GAP**= $\frac{\sum_{i=1}^{20} \alpha_i}{20}$ and **Standard deviation**= $\sqrt{\frac{\sum_{i=1}^{20} \alpha_i^2}{20}}$.

From table 1, we can notice that the value of α tends towards 1 when we increase the size of the instances. What can be said, is that the solution of $C^{\star}_{max1'}$ is very close to the solution of $C^{\star}_{maxp'}$ in the general case.

Table 1. GAP and standard deviation

Instances	Number of tasks	Average GAP	Standard deviation
test_1	10	1.12457	1.12871
test_2	30	1.11632	1.12065
test_3	60	1.00046	1.00046
test_4	100	1.00007	1.00007
test_5	200	1.00124	1.00126
test_6	400	1	1
test_7	500	1	1
test_8	600	1	1
test_9	800	1	1
test_10	1000	1	1
Average	/	1,024266	1,025115

Lemma 3. *The ratio between the optimal solution $C^{\star}_{max1'}$ of the formulation $(P1)$ and the optimal scheduling solution $C^{\star}_{max}$ of our main problem is given by $C^{\star}_{max1'} \leqslant \alpha C^{\star}_{max}$.*

Proof. From Remark 1, we have $C^{\star}_{maxp} \leqslant C^{\star}_{max}$. Then, $C^{\star}_{max1'} \leqslant \alpha C^{\star}_{maxp'} \leqslant \alpha C^{\star}_{maxp} \leqslant \alpha C^{\star}_{max}$.

Rounding strategy: If $x_i^{'}$ is integer for $i \in \{1, \ldots, n\}$, the solution obtained is feasible and optimal for $(P1)$, otherwise the fractional values are rounded. We denote by x_i^r the rounded value of the fractional value of the assignment variable of task t_i in the optimal solution of $(P1)$. We set $x_i^r = 0$ if $x_i^{'} < \frac{1}{2}$, $x_i^r = 1$ otherwise.

Let θ_1 be the mapping obtained by this rounding. Each task t_i is mapped in either Pe_1 or Pe_2. Thus, $\theta_1(t_i) \longrightarrow \{Pe_1, Pe_2\}$.

4.3 Phase 3: Scheduling Algorithm

Using the mapping θ_1, the following algorithm determines for a task order given by a priority list L, the corresponding scheduling by executing the first task ready of the list as long as there are free processing elements.

The priority list L can be defined in different ways. To achieve good scheduling, the most important and influential tasks must be executed first. For this purpose, the following list is particularly interesting for this problem because it takes into account the critical path of the graph. First, we start by defining graph $G'(V, E)$, with $V = \{t_1, t_2, ..., t_n\}$ and E represents the set of graph edges. The vertices are labelled by the execution time of each task according to their assignments. The edges are labelled by the communication costs if t_i precedes t_j and $x_i^r \neq x_j^r$, 0 otherwise. Then, we can calculate the longest path $LP(t_i)$ from each task t_i to its last successor. The list LLP is given by $LLP = \{t_1, t_2, ..., t_n\}$,

such that $LP(t_1) \geqslant LP(t_2) \geqslant ... \geqslant LP(t_n)$. The following algorithm executes task by task, executing first the task t_i with the highest $LP(t_i)$ from ready tasks. It uses an insertion policy that tries to insert a task at the earliest idle time between two already scheduled tasks on the processing element, if the slot is large enough to execute the task.

Algorithm 1: PLS (Polynomial List Scheduling) algorithm

Data: mapping θ_1, list LPP.
Result: Feasible scheduling.
begin
  Create an empty list ready-list;
  ready-list= $\{t_j,\ \Gamma^-(j) = \emptyset,\ j \in \{1, \ldots, n\}\}$;
  while *ready-list* $\neq \emptyset$ **do**
    $t_i \longleftarrow$ task with highest $LP(t_i)$ from ready-list;
    Execute t_i on $\theta_1(t_i)$ using insertion-based scheduling policy;
    Update ready-list;

The three steps of PLS (Polynomial List Scheduling) algorithm can be summarized as follows. Solve the relaxed formulation $(P^{'})$ then us its solution to define another formulation $(P1)$, then solve $(P1)$. Finally, After rounding the solutions obtained by $(P1)$, use Algorithm 1 with the obtained mapping θ_1 and the priority list LLP.

Complexity: Mapping θ_1 is based on solving two linear formulations ($(P^{'})$ and $(P1)$) with continuous variables, which are two polynomial problems. This gives polynomial-time solving methods for the first two phases of PLS algorithm. In the last phase, ready-list is calculated with $\mathcal{O}(n^2)$ time complexity. The insertion policy is verified on a processing element by checking the non-overlapping with at most $(n-1)$ tasks. This makes a complexity of $\mathcal{O}(n^2)$ for the last phase. Thus, the complexity time of PLS algorithm is polynomial.

Algorithm Analysis: In the following, we study the performance of PLS algorithm in the worst case compared to the optimal solution. We look for the ratio between the solution $\widehat{C}_{max}$ obtained by PLS algorithm and the optimal scheduling solution $C^{\star}_{max}$ of our main problem.

Lemma 4. *The rounding θ_1 previously defined satisfies the following inequalities: $x_i^r \leqslant 2x_i^{'}$ and $(1 - x_i^r) \leqslant 2(1 - x_i^{'})$.*

Proof. If $0 \leqslant x_i^{'} < \frac{1}{2}$, then $x_i^r = 0 \leqslant 2x_i^{'}$. Furthermore, $2x_i^{'} \leqslant 1$, then $0 \leqslant 1-2x_i^{'}$, follows $-x_i^r = 0 \leqslant 1-2x_i^{'}$, then $1-x_i^r \leqslant 2(1-x_i^{'})$. If $\frac{1}{2} \leqslant x_i^{'}$ then $1 \leqslant 2x_i^{'}$, follows $x_i^r = 1 \leqslant 2x_i^{'}$. Furthermore, $x_i^{'} \leqslant 1$ then $-2x_i^{'} \geqslant -2$, follows $1 - 2x_i^{'} \geqslant -1$, then $-x_i^r = -1 \leqslant 1 - 2x_i^{'}$, then $1 - x_i^r \leqslant 2(1 - x_i^{'})$.

Let w_i^r be the execution time of the task t_i by considering the rounding θ_1, where $w_i^r = w_{i,0}$ if $x_i^r = 1$, $w_i^r = w_{i,1}$ otherwise. Let $w_i^{'}$ be the execution time of the task t_i by considering the solution of $(P1)$, where $w_i^{'} = x_i^{'} w_{i,0} + (1 - x_i^{'}) w_{i,1}$.

Proposition 1. *The relation between w_i^r and $w_i^{'}$ of each task t_i is given by $w_i^r \leqslant 2w_i^{'}$, $i \in \{1, \ldots, n\}$.*

Proof. From Lemma 4, $2w_i^{'} = 2x_i^{'} w_{i,0} + 2(1-x_i^{'}) w_{i,1} \geqslant x_i^r w_{i,0} + (1-x_i^r) w_{i,1} = w_i^r$.

Lemma 5. *For two successive tasks $(t_i, t_j) \in E$, if t_i and t_j are executed by two different processing elements, then $\zeta_{i,j}^{'} > \frac{1}{2}$.*

Proof. We have two cases:

1. If $\min\{\widetilde{x}_i^{'}, \widetilde{x}_j^{'}\} > \min\{1 - \widetilde{x}_i^{'}, 1 - \widetilde{x}_j^{'}\}$, then from $Con_{i,j}$ constraint, we have $\zeta_{i,j}^{'} = (1 - min\{x_i^{'}, x_j^{'}\})$:
   a. If $x_i^{'} < \frac{1}{2}$ and $x_j^{'} \geqslant \frac{1}{2}$, then $\zeta_{i,j}^{'} = (1 - x_i^{'}) > \frac{1}{2}$.
   b. If $x_i^{'} \geqslant \frac{1}{2}$ and $x_j^{'} < \frac{1}{2}$, then $\zeta_{i,j}^{'} = (1 - x_j^{'}) > \frac{1}{2}$.
2. If $\min\{\widetilde{x}_i^{'}, \widetilde{x}_j^{'}\} \leqslant \min\{1 - \widetilde{x}_i^{'}, 1 - \widetilde{x}_j^{'}\}$, then from $Con_{i,j}$ constraint, we have $\zeta_{i,j}^{'} = (1 - min\{1 - x_i^{'}, 1 - x_j^{'}\})$:
   a. If $1 - x_i^{'} < \frac{1}{2}$ and $1 - x_j^{'} \geqslant \frac{1}{2}$, then $\zeta_{i,j}^{'} = x_i^{'} > \frac{1}{2}$.
   b. If $1 - x_i^{'} \geqslant \frac{1}{2}$ and $1 - x_j^{'} < \frac{1}{2}$, then $\zeta_{i,j}^{'} = x_j^{'} > \frac{1}{2}$.

For each couple of tasks $(t_i, t_j) \in E$, we denote by $Cost_{i,j}^r$ the value given by $Cost_{i,j}^r = 0$ if $x_i^r = x_j^r$, $Cost_{i,j}^r = ct_{i,j}$ otherwise. Let $Cost_{i,j}^{'}$ be the value given by $Cost_{i,j}^{'} = \zeta_{i,j}^{'} ct_{i,j}$.

Proposition 2. *For each couple of tasks $(t_i, t_j) \in E$, the relation between $Cost_{i,j}^r$ and $Cost_{i,j}^{'}$ is given by $Cost_{i,j}^r < 2Cost_{i,j}^{'}$.*

Proof. If t_j and t_j are executed by the same processing element, $Cost_{i,j}^r = 0 \leqslant 2\zeta_{i,j}^{'} ct_{i,j}$, because $\zeta_{i,j}^{'} \geqslant 0$. If t_j and t_j are executed by two different processing elements, then $Cost_{i,j}^r = ct_{i,j}$. Then, from Lemma 5, $\zeta_{i,j}^{'} > \frac{1}{2}$, then $2\zeta_{i,j}^{'} > 1$, follows $Cost_{i,j}^r = ct_{i,j} \leqslant 2\zeta_{i,j}^{'} ct_{i,j} = 2Cost_{i,j}^{'}$.

Proposition 3. *For each two successive tasks $(t_i, t_j) \in E$, let be $l_{i,j}^r = w_i^r + Cost_{i,j}^r + w_j^r$ (resp. $l_{i,j}^{'} = w_i^{'} + Cost_{i,j}^{'} + w_j^{'}$) the length of (t_i, t_j) in PLS solution (resp. $(P1)$ solution). Then, we have $l_{i,j}^r < 2l_{i,j}^{'}$.*

Proof. From Proposition 1 and Proposition 2, $l_{i,j}^r = w_i^r + Cost_{i,j}^r + w_j^r < 2w_i^{'} + 2Cost_{i,j}^{'} + 2w_j^{'} = 2l_{i,j}^{'}$. Thus, $l_{i,j}^r < 2l_{i,j}^{'}$.

Theorem 2. *Let $\widehat{C}_{max}$ be the solution obtained by using PLS algorithm, then $\widehat{C}_{max} < 6C_{max1'}^{\star}$.*

Proof. From Proposition 3, the length of each path L from $G(V, E)$ is given by $length(L)^r = \sum_{(t_i,t_{i+1})\in L} l^r_{i,i+1} \leqslant 2\sum_{(t_i,t_{i+1})\in L} l^{'}_{i,i+1} = 2length(L)^{'}$, where $length(L)^r$ (resp. $length(L)^{'}$) is the length of L in PLS solution (resp. $(P1)$ solution). Furthermore, the workload of the tasks assigned to Pe_1 (resp. Pe_2) is given by $\sum_{i=1}^{n} x_i^r w_{i,0} = 2\sum_{i=1}^{n} x_i^{'} w_{i,0}$ (resp. $\sum_{i=1}^{n}(1 - x_i^r)w_{i,0} = 2\sum_{i=1}^{n}(1 - x_i^{'})w_{i,0}$). Thus, the value of the critical path and the workloads on Pe_1 and Pe_2 will be at most doubled compared to the lower bounds. Finally, the interaction between the longest paths and the workload on each processing element has been studied in [15], such that if we have these properties, then $\widehat{C}_{max} < 6C^{\star}_{max1'}$.

Theorem 3. *The ratio between the solution $\widehat{C}_{max}$ obtained by PLS algorithm and the optimal scheduling solution $C^{\star}_{max}$ is given by $\dfrac{\widehat{C}_{max}}{C^{\star}_{max}} < 6\alpha$.*

Proof. From Lemma 3, we have $C^{\star}_{max1'} \leqslant \alpha C^{\star}_{max}$. Then, $\dfrac{\widehat{C}_{max}}{C^{\star}_{max}} < \dfrac{6C^{\star}_{max1'}}{C^{\star}_{max}} \leqslant \dfrac{6\alpha C^{\star}_{max}}{C^{\star}_{max}} \leqslant 6\alpha$.

5 Numerical Results

We compare here the performance of PLS (Polynomial List Scheduling) algorithm to HEFT (Heterogeneous Earliest Finish Time) and LS (List Scheduling) algorithm using benchmarks generated by Turbine [19].

The benchmark is composed of ten parallel DAG applications. We denote by *test_i* instance number i. We generate 10 different applications for each *test_i* with $i \in \{1, \ldots, 10\}$. The execution times of the tasks are generated randomly over an interval $[w_{min}, w_{max}]$, w_{min} has been fixed at 5 and w_{max} at 70. The number of successors of each task is generated randomly over an interval $[d_{min}, d_{max}]$, d_{min} has been fixed at 1 and d_{max} at 10. The communication rate for each arc was generated on an interval $[ct_{min}, ct_{max}]$, we set ct_{min} to 35 and ct_{max} to 80.

To study the performance of our method, we compared the ratio between each makespan value obtained by PLS algorithm with HEFT and LS algorithm, the optimal solution obtained by *CPLEX* and the lower bound $C_{max1'}$ obtained by $(P1)$. Table 2 shows the average results obtained on 10 instances given in column Inst of each application size given in the second column using *CPLEX*, HEFT, PLS and LS algorithms. We show the average time that was needed to *CPLEX* to provide the optimal solution using (*Opt*). We only have the result for the first two instances due to the large running time for instances with more than 60 tasks ($> 4h$). Then, we show the results obtained by HEFT, PLS and LS algorithm. GAP columns give the average ratio between the makespan obtained by each method compared to $C_{max1'}$ using the following formula: $GAP = \frac{\text{method makespan} - C_{max1'}}{C_{max1'}} \times 100$. Time columns show the average time that was needed for each method to provide a solution. Best columns present

the number of instances where each algorithm provides better or the same solution than other methods. A line Average is added at the end of each table which represents the average of the values each column.

Table 2. *CPLEX*, HEFT, LS and PLS algorithms results.

Inst	Number	*CPLEX*		HEFT			LS algorithm			PLS algorithm		
	of tasks	Optimal	Time	GAP	Time	Best	GAP	Time	Best	GAP	Time	Best
test_1	10	✓	0.35 s	33.23%	0.0016 s	2	20.84%	0.25 s	7	21.24%	0.008 s	7
test_2	30	✓	59.58 s	38.11%	0.0049 s	4	52.34%	0.600 s	0	43.19%	0.028 s	5
test_3	60	X	X	25.89%	0.009 s	4	24.86%	0.29 s	6	24.81%	0.081 s	7
test_4	100	X	X	15.80%	0.017 s	0	6.85%	0.198 s	8	6.46%	0.184 s	9
test_5	200	X	X	14.56%	0.044 s	0	1.34%	0.51 s	7	1.08%	0.68 s	6
test_6	400	X	X	11.80%	0.19 s	0	0.25%	1.72 s	7	0.31%	2.52 s	6
test_7	500	X	X	11.50%	0.26 s	0	0.16%	1.84 s	6	0.11%	2.33 s	9
test_8	600	X	X	11.53%	0.61 s	0	0.32%	2.06 s	6	0.237%	2.09 s	7
test_9	800	X	X	11.78%	1.40 s	0	0.14%	3.15 s	7	0.13%	4.09 s	6
test_10	1000	X	X	12.11%	1.90 s	0	0.052%	4.32 s	7	0.06%	5.26 s	7
Average	/	/	/	18.63%	0.44 s	6%	10.71%	1.49 s	61%	9.76%	1.72 s	69%

For the running time, HEFT algorithm requires less time than PLS and LS algorithms to provide a solution. PLS algorithm is the most efficient method with a gap of 9.76% and 69% of better solutions compared to other methods. Its average running time is 1.72 s, which is slightly higher than the running time of LS algorithm.

6 Conclusion and Perspectives

This paper presents an efficient algorithm to solve the problem of scheduling parallel applications on hybrid platforms with communication delays. The objective is to minimise the total execution time (makespan).

After modelling the problem, we proposed a three-phase algorithm; the first two phases consist in solving linear formulations to find the type of processor assigned to execute each task. In the third phase, we compute the start execution time of each task to generate a feasible schedule. Tests on large instances close to reality demonstrated the efficiency of our method comparing to other methods and shows the limits of solving the problem with a solver such as *CPLEX*.

A proof of the performance guarantee for PLS algorithm was initiated. In future works, we will focus on finding the value of α to have a fixed bound on the ratio between $\widehat{C}_{max}$ and $C^{\star}_{max}$. Tests on real applications and an extension to more general heterogeneous platforms is also planned.

References

1. Shen, L., Choe, T.-Y.: Posterior Task scheduling algorithms for heterogeneous computing systems. In: Daydé, M., Palma, J.M.L.M., Coutinho, Á.L.G.A., Pacitti, E., Lopes, J.C. (eds.) VECPAR 2006. LNCS, vol. 4395, pp. 172–183. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71351-7_14

2. Benoit, A., Pottier, L., Robert, Y.: Resilient co-scheduling of malleable applications. Int. J. High Perform. Comput. Appl. **32**(1), 89–103 (2018)
3. Ullman, J.D.: Np-complete scheduling problems. J. Comput. Syst. Sci. **10**(3), 384–393 (1975)
4. Imreh, C.: Scheduling problems on two sets of identical machines. Computing **70**(4), 277–294 (2003)
5. Marchal, L., Canon, L.-C., Vivien, F.: Low-cost approximation algorithms for scheduling independent tasks on hybrid platforms. Ph.D. thesis, Inria-Research Centre Grenoble-Rhône-Alpes (2017)
6. Kedad-Sidhoum, S., Monna, F., Mounié, G., Trystram, D.: A family of scheduling algorithms for hybrid parallel platforms. Int. J. Found. Comput. Sci. **29**(01), 63–90 (2018)
7. Kedad-Sidhoum, S., Monna, F., Trystram, D.: Scheduling tasks with precedence constraints on hybrid multi-core machines. In: IPDPSW, pp. 27–33. IEEE (2015)
8. Amaris, M., Lucarelli, G., Mommessin, C., Trystram, D.: Generic algorithms for scheduling applications on hybrid multi-core machines. In: Rivera, F.F., Pena, T.F., Cabaleiro, J.C. (eds.) Euro-Par 2017. LNCS, vol. 10417, pp. 220–231. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64203-1_16
9. Topcuoglu, H., Hariri, S., Min-you, W.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
10. Boeres, C., Rebello, V.E.F., et al.: A cluster-based strategy for scheduling task on heterogeneous processors. In: 16th Symposium on Computer Architecture and High Performance Computing, SBAC-PAD 2004, pp. 214–221. IEEE (2004)
11. Yang, T., Gerasoulis, A.: DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Trans. Parallel Distrib. Syst. **5**(9), 951–967 (1994)
12. Garey, M.R., Johnson, D.S.: Complexity results for multiprocessor scheduling under resource constraints. SIAM J. Comput. **4**(4), 397–411 (1975)
13. Khan, M.A.: Scheduling for heterogeneous systems using constrained critical paths. Parallel Comput. **38**(4–5), 175–193 (2012)
14. Kushwaha, S., Kumar, S.: An investigation of list heuristic scheduling algorithms for multiprocessor system. IUP J. Comput. Sci. **11**(2) (2017)
15. Aba, M.A., Zaourar, L., Munier, A.: Approximation algorithm for scheduling applications on hybrid multi-core machines with communications delays. In: 2018 IEEE IPDPSW, pp. 36–45. IEEE (2018)
16. Zaourar, L., Aba, M.A., Briand, D., Philippe, J.-M.: Modeling of applications and hardware to explore task mapping and scheduling strategies on a heterogeneous micro-server system. In: IPDPSW, pp. 65–76. IEEE (2017)
17. IBM: Ibm ilog cplex v12.5 user's manual for cplex. http://www.ibm.com
18. Aba, M.A., Pallez, G., Munier-Kordon, A.: Scheduling on two unbounded resources with communication costs (2019)
19. Bodin, B., Lesparre, Y., Delosme, J.-M., Munier-Kordon, A.: Fast and efficient dataflow graph generation. In: Proceedings of the 17th International Workshop on Software and Compilers for Embedded Systems. ACM (2014)