The Journal of Supercomputing (2023) 79:19019–19048
https://doi.org/10.1007/s11227-023-05325-w

A mathematical programming approach for resource allocation of data analysis workflows on heterogeneous clusters

Somayeh Mohammadi[1] · Latif PourKarimi[2] · Felix Droop[1] · Ninon De Mecquenem[3] · Ulf Leser[3] · Knut Reinert[1]

Accepted: 18 April 2023 / Published online: 23 May 2023

Abstract
Scientific communities are motivated to schedule their large-scale data analysis workflows in heterogeneous cluster environments because of privacy and financial issues. In such environments containing considerably diverse resources, efficient resource allocation approaches are essential for reaching high performance. Accordingly, this research addresses the scheduling problem of workflows with bag-of-task form to minimize total runtime (makespan). To this aim, we develop a mixed-integer linear programming model (MILP). The proposed model contains binary decision variables determining which tasks should be assigned to which nodes. Also, it contains linear constraints to fulfill the tasks requirements such as memory and scheduling policy. Comparative results show that our approach outperforms related approaches in most cases. As part of the post-optimality analysis, some secondary preferences are imposed on the proposed model to obtain the most preferred optimal solution. We analyze the relaxation of the makespan in the hope of significantly reducing the number of consumed nodes.

Keywords Heterogeneous cluster environments · Scheduling · Data analysis workflow · Mixed integer linear programming · Makespan minimization · Post-optimality analysis

1 Introduction

Data analysis workflows (DAWs) consist of tasks with control or data dependencies. The use of DAWs to help scientists organize the steps involved in large-scale, complex scientific processes is a well-known paradigm [1, 2]. Generally, the structure of a DAW is represented with a directed acyclic graph (DAG). Some workflows are known as bag-of-tasks (BoT) workflows which consist of a set of concurrent bags; each includes a large number of independent homogeneous tasks

Extended author information available on the last page of the article

[3]. Scientists often use a heterogeneous cluster infrastructure to run their DAWs where a set of computing nodes are interconnected with a high-speed network. This choice is due to privacy and financial issues and because it provides high-performance computing environments. Hence, large-scale scientific workflows can be run within a reasonable time [4]. In such an environment, the main goal is to schedule the tasks of the DAW to the computing resources so that the total run time (makespan) is minimized. Generally, scheduling a DAG in heterogeneous systems is a well-known problem in scheduling theory and is NP-Hard [5] because of the heterogeneity within the computing system and the precedence constraints between tasks.

The current research investigates the problem of BoT DAWs scheduling in heterogeneous cluster infrastructures by using mathematical programming. Mathematical models are applicable in scientific and engineering problems, especially those related to optimization. The main advantages of a mathematical model can be highlighted as follows [3, 6]:

- In a mathematical model, the objective function, decision variables, and the problem constraints are expressed using mathematical expressions. These models provide insight into the problem structure. Some critical information, such as post-optimality analysis or trade-offs, can be derived explicitly.
- The model can be adapted when the problem conditions change. Modifying the key features of the model (decision variables, constraints, or objective functions) is possible since the underlying mathematical programming engine and algorithms always remain the same. Therefore, maintaining mathematical optimization applications is easier than using heuristics.
- When we construct a mathematical optimization model for a scientific approach, this approach improves as related optimization algorithms improve.

The model adopted in this work is a *Mixed Integer Linear Programming* (MILP) model. It provides comprehensive, clear, and flexible descriptions of our assignment problem. This model, because of its linear structure, is also one of the simplest mathematical models that can be solved in a reasonable time through efficient algorithms [7]. A proper combination of compute nodes is selected for the DAW execution so the objective function is minimized, and problem constraints are satisfied. After solving the model, some post-optimality analyses are performed.

The contributions of the current study are highlighted as follows:

- The problem of DAW scheduling in a heterogeneous cluster environment is formulated as a MILP model to minimize makespan. We compared our approach to five well-known and novel approaches on three different cluster sizes for four different scientific workflows. Comparative results show that our approach outperforms related approaches in 75% of cases.
- As a part of the post-optimality analysis, choosing a preferred solution among the set of (possible) alternative optimal solutions is investigated.
- Relaxation of the makespan in the hope of significantly reducing the number of consumed nodes is probed in another post-optimality analysis.

The paper is organized as follows: Sect. 2 presents a review of the related work, and Sect. 3 illustrates the system model consisting of the infrastructure model, the workflow model and the problem statement. Section 4 introduces the proposed mathematical model. Section 5 discusses the numerical results, and Sect. 6 provides concluding remarks and plans for further studies.

2 Related works

Much research has been done on the problem of scientific DAW scheduling. In a cloud infrastructure, the objective is cost minimization because cloud providers offer a pay-as-you-go model for leasing resources, while in a cluster infrastructure the main goal is makespan minimization [4, 8].

Generally, workflows are scheduled using static or dynamic methods [9, 10]. Static scheduling approaches address the problem of assigning a set of tasks to compute resources in advance. It is assumed that accurate information about workflow and infrastructure resources can be obtained before scheduling, while dynamic scheduling approaches don't require such assumptions. In this paper, we propose a static scheduling approach for scientific DAWs.

Since obtaining an optimal solution for the DAWs scheduling problem is difficult within a reasonable time in most cases, heuristic-based algorithms have been widely used in the past few decades [8, 9]. Heuristics provide approximate solutions within polynomial time periods. These algorithms for the workflow resource allocation problem in an HPC system can be classified into three groups: list-based, clustering-based, and duplication-based scheduling [11, 12].

A list scheduling algorithm has two phases. The first phase consists of giving a priority to each task of the workflow. The second phase selects the computing node that will execute the task. List-based algorithms are superior to the other groups of heuristic algorithms [13]. HEFT [14] computes the priority or rank of each task t_i using an upward rank value which indicates the length of the critical path from task t_i to the exit task. In the task assignment phase, HEFT assigns each task to a node that allows completing the task the earliest. HEFT uses an insertion policy in which tries to insert a task at the earliest idle time between two already scheduled tasks on a computing node, if the slot is large enough to accommodate the task. The PEFT algorithm [15] is an effective list-based scheduling technique with a look-ahead feature. It predicts the impact of an assignment for all successor tasks of the current task without sacrificing the time complexity. This prediction is based on an "optimistic cost table (OCT)" which is used to rank tasks of the workflow, and to select a processor for a task. The IPPTS algorithm [16] introduced the "looking ahead" as a new feature in the task prioritization phase while considering the "out-degree" of a task, and used "downward-upward" approach to look ahead in the computing node selection phase. The IPEFT algorithm [17] calculates the priority of the tasks based on a pessimistic cost table and uses a critical node cost table to predict features. In [18], an estimation of distribution algorithm (EDA) enhanced by path relinking is proposed to minimize makespan. To describe the relative position relationships between two tasks, a specific probability model is created and the task processing permutations are generated by sampling such a model.

In our proposed method, it is assumed that the workflows are in the form of BoT; however, there is no such assumption in heuristic methods (e.g., HEFT) and they can be used in the most general form of workflows. Therefore, it is expected that heuristic methods provide a lower makespan allocation than the proposed method, but because the heuristic methods obtain sub-optimal solutions, this does not happen in some cases. On the other hand, the proposed method has other advantages over heuristic methods. Firstly, despite accepting the above-mentioned limitation, because the proposed method obtains an optimal solution, its solutions are competitive with the solutions of heuristic methods that operate without the above-mentioned limitation and obtain sub-optimal solutions. Secondly, the proposed method has a very important advantage that the heuristic method lacks. By using mathematical models, one can apply secondary criteria to the set of optimal solutions (expertise) and provide post-optimality analysis. For this purpose, we need an explicit representation of all optimal solutions. In mathematical models, this can be done by adding a constraint that guarantees optimality, whereas heuristic methods lack this capability. More precisely, in mathematical programming, the optimality condition is imposed by adding an equality or inequality to the set of constraints of the problem. Thus, the feasible space of the augmented problem is the same as the set of all the optimal solutions to the original problem. Therefore, any type of secondary preference can be applied to this set; equivalently, any secondary objective function can be optimized on it.

A clustering algorithm has three phases [19, 20]: clustering, cluster-merging, and task-ordering within clusters. In the clustering phase, the highly communicating tasks are grouped into clusters. Then, the clusters are merged to be mapped to the bounded number of computing nodes (cluster-merging phase). In the task-ordering phase, task priority is determined to obtain a schedule. The CMWSL algorithm [21] calculates the lower bound of the total execution time for each node by taking into account both the system and application characteristics. As a result, the number of nodes used for actual execution is adjusted to minimize the Worst Schedule Length (WSL). The actual task assignment and task clustering are then performed to minimize the schedule length until the total execution time in a task cluster exceeds the lower bound. The RBCA and DBCA algorithms [22] addressed runtime and dependency imbalance problems, respectively. The dependency imbalance may occur when a task requires data from other tasks clustered in different task clusters; thus, this delays the release of subsequent tasks and increases the overall makespan. The runtime imbalance occurs when the tasks are assigned to the different task clusters with a significant difference in execution time; thus, the execution of task clusters on the next level of the workflow will be delayed and it affects the overall makespan of the workflow. To avoid the dependency imbalance, DBCA clusters the tasks with high dependency correlation(similarity of two tasks in terms of data dependency) together while RBCA clusters the tasks into multiple task clusters with roughly equal workload summation.

Most clustering-based algorithms (e.g. DBCA and RBCA) select the optimal number of task clusters equal to the number of computing nodes. Furthermore, they try to reduce data transfer time by grouping highly communicating tasks into a common task cluster. Employing all computing nodes may not lead to minimum makespan. In our proposed approach, the highly communicating tasks are implicitly grouped by considering the data transfer time in the objective function. Moreover, unlike clustering-based algorithms, the proposed approach does not necessarily use all of the computing nodes. Notably, as a post-optimality analysis, the proposed approach can yield the minimum number of consumed nodes with the minimum makespan (See Sect. 5.2.1).

In duplication-based algorithms, the main idea is to duplicate tasks on multiple computing nodes so that the results of the duplicated tasks are available on multiple nodes to trade computation time for communication time[11, 13]. At first, the TDCA algorithm [23] assigns tasks to computing nodes to generate several initial clusters. In the next step, a task duplication method is applied to modify the initial clusters to reduce the makespan iteratively, so new clusters are added if necessary. Then, TDCA merges some clusters and decreases the number of occupied computing nodes to obtain a preliminary schedule. Finally, a task insertion scheme is adopted which inserts a task to an idle time slot located before its successor tasks if this insertion could make the successor start earlier.

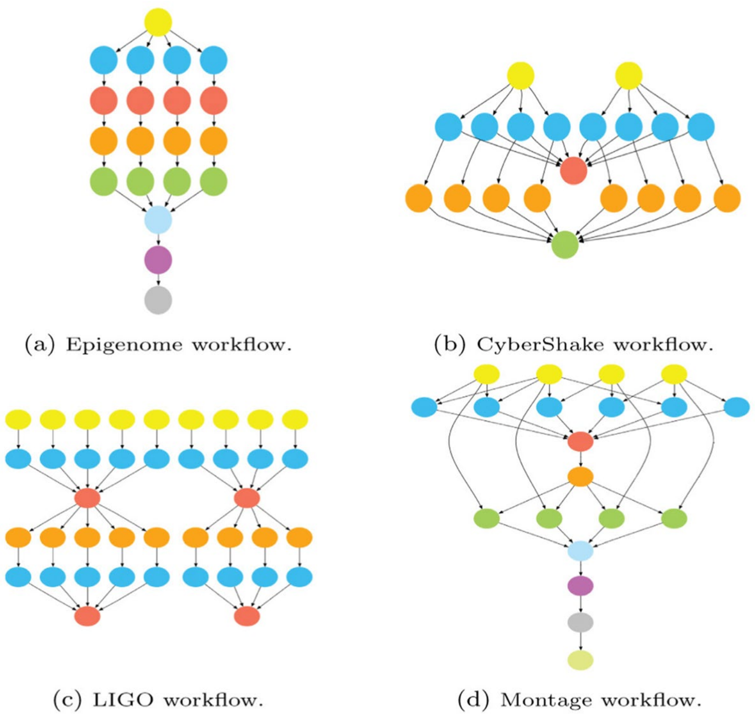
It is concluded from the above-mentioned discussion that achieving a low value of makespan requires a large number of iterations, which leads to a large amount of running time for the algorithm. As is expected the experimental results show that the proposed approach outperforms the TDCA method for all cases in the viewpoint of makespan.

3 System model

This section describes the infrastructure model and the workflow model. At the end, it introduces the problem statement.

3.1 Infrastructure model

We assume that the target computing environment is a heterogeneous cluster, which is a common configuration used for workflow execution [24, 25]. A heterogeneous cluster consists of multiple compute nodes that are different in the amount of cores, memory, and performance characteristics. Also, nodes are usually connected to a distributed file system via a fast interconnection network. We assume that each node k of the cluster is connected to the network via a dedicated link characterized by its bandwidth BW_k. Notably, our model takes data transfer between tasks into account, in addition to the individual task run times themselves (see Sect. 4). However, we do not model the effects of potential network contention due to too many concurrent data transfers (see, e.g., [26]). We plan to extend our model to include such effects as future work.



(a) Epigenome workflow.

(b) CyberShake workflow.

(c) LIGO workflow.

(d) Montage workflow.

Fig. 1 The general structure of scientific DAWs [27]

3.2 Workflow model

As previously stated, a workflow consists of a set of computational tasks with specific inter-dependencies represented by a DAG. Dependencies indicate data flow and execution priority between tasks. We consider the following four scientific workflows from various applied scientific fields:

- **Epigenome**: Biology
- **LIGO**: Gravitational physics
- **Montage**: Astronomy
- **CyberShake**: Earthquake science

The specifications of these workflows can be found in the Pegasus Workflow Generator.[1] A small structure of the workflows is shown in Fig. 1. Some task distribution patterns are commonly used in scientific workflows: pipeline, scatter, gather, broadcast, reuse, and distribute [27, 28]. In these patterns, every level of the DAG contains a group of parallel homogeneous tasks or a single task namely a Bag-of-Task(BoT). A bag includes tasks located in the same level of DAG which

[1] https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator.

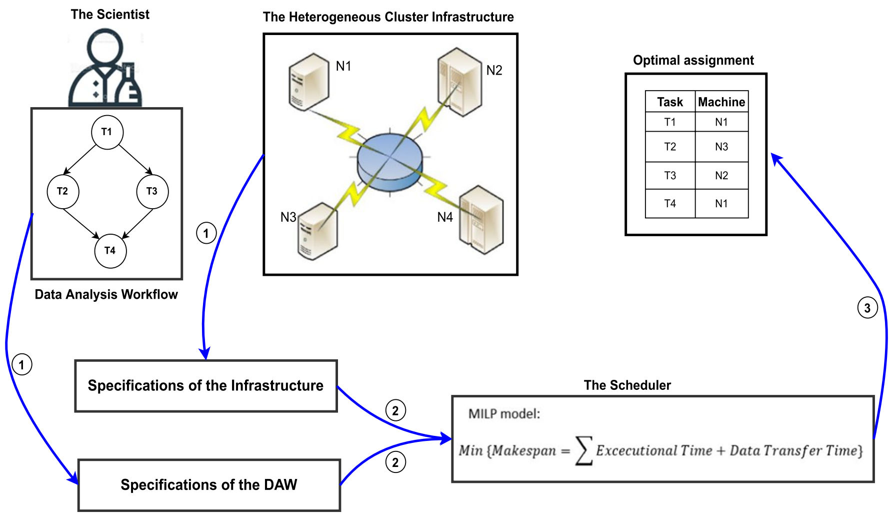

The Scientist
T1
T2
T3
T4
Data Analysis Workflow
The Heterogeneous Cluster Infrastructure
N1
N2
N3
N4
Optimal assignment
Task | Machine
T1 | N1
T2 | N3
T3 | N2
T4 | N1
1
1
2
2
3
Specifications of the Infrastructure
Specifications of the DAW
The Scheduler
MILP model:
Min {Makespan = ∑ Excecutional Time + Data Transfer Time}

Fig. 2 Overview of the system model

have approximately the same characteristics (e.g. runtime and required memory). Studies of the real-world characteristics of parallel workloads show that between 34% and 89% of applications running on parallel systems have a BoT structure [29]. Given that no data dependency exists between the tasks in a bag, they can be executed parallelly. In this study, we propose a model that is applicable to those workflows that have BoT structures. Such partitioning enables us to consider a few bags instead of many tasks individually [3, 30].

3.3 Problem statement

This study addresses the scheduling of scientific workflow applications in heterogeneous cluster environments through mathematical programming with an objective of makespan minimization. We assumed that m computing nodes are available in a cluster environment and that the workflow application consists of n bag-of-tasks at the n levels. We formulated this scheduling problem as an MILP model.

Figure 2 provides an overview of the system model. During the first step, specifications of the cluster infrastructure and the workflow are collected. It can be done by existing approaches to profile the cluster nodes performance [4, 24] and analyzing existing task performance metrics from historical workflow executions. In the second step, these specifications are submitted to the scheduler as input for an MILP mode. Then, by solving the MILP model, optimal assignment is obtained. Finally, the optimal assignment is used to run the DAW in the cluster.

4 The proposed mathematical model

A mathematical planning study involves: providing a mathematical model for the problem being evaluated, solving the model, and performing a post-optimality analysis. In the proposed model, the objective function and constraints are expressed by linear functions of decision variables. The decision variables contain non-negative integer, binary, and real variables. Thus, the proposed model is an MILP model. Due to the simple structure of the model, efficient computational algorithms and software programs are available to solve it. In the following, we explain how to create constraints and express the objective function as a function of decision variables.

The input data for the mathematical model are described in Table 1. Also, this table explains decision variables existing in the proposed model. The workflow characteristics (e.g., number of levels and tasks in each level) and the infrastructure specifications (e.g., number of cores and size of memory) are available.

The makespan of a workflow includes executional and data transfer times and the objective function minimizes the makespan. The model has binary and real decision variables. For example, assignment of tasks to computing nodes of the cluster is determined by binary variables. Also, some other auxiliary variables have been used in the proposed model. In addition, the model fulfills constraints on the workflow requirements and infrastructure resource limitations. For example, it ensures that each task is assigned to only one computing node and the memory requirement is fulfilled, etc. The objective function, constraints, and decision variables are as follows:

$$\text{(MILP model)} \quad \text{Total Run-Time } = \min MS_n \tag{1}$$

subject to:

$$\sum_{k=1}^{m} x_{ijk} = 1 \qquad \forall i \in \{1,2,\dots,n\}, \forall j \in J_i \tag{2}$$

$$\sum_{j \in J_i} x_{ijk} \le \gamma_{ik} \cdot \delta_i \qquad \forall i \in \{1,2,\dots,n\}, \forall k \in \{1,2,...,m\} \tag{3}$$

$$\sum_{j \in J_i} x_{ijk} \ge \gamma_{ik} \qquad \forall i \in \{1,2,\dots,n\}, \forall k \in \{1,2,\cdots,m\} \tag{4}$$

$$\gamma_{ik} \cdot Mem_i \le mem_k \qquad \forall i \in \{1,\dots,n\}, \forall k \in \{1,\cdots,m\} \tag{5}$$

$$Z_i \ge \gamma_{i-1,k} - \gamma_{i,k} \qquad \forall i \in \{2,\dots,n\}, \forall k \in \{1,2,\dots,m\} \tag{6}$$

$$Z_i \ge \gamma_{i,k} - \gamma_{i-1,k} \qquad \forall i \in \{2,\dots,n\}, \forall k \in \{1,2,\dots,m\} \tag{7}$$

Table 1 Models parameters/variables and their description

Notation of parameters	Description
$CN = \{node1, node2, ..., node_m\}$	Cluster node set
m	The number of nodes of the cluster
CN_k	kth node of the cluster
mem_k	Accessible memory size of CN_k for running tasks
C_k	The number of cores CN_k
P_k	The performance of each core of CN_k (e.g. floating point operations per second)
BW_k	The bandwidth of network for CN_k
$A = \{B1, B2, ..., B_n\}$	The set of bags of the workflow
n	The number of level(Bags) of the workflow
B_i	i_{th} bag of the workflow
δ_i	The number of tasks contained in B_i
W_i	Required amount of computation for one task of B_i(e.g. number of floating point operations)
IDS_i	Input data size for B_i
ODS_i	Output data size for B_i
J_i	The set of tasks of B_i
Mem_i	Required memory for each task of B_i
TSK_{ij}	jth task of B_i
Notation of decision variables	**Description**
x_{ijk}	1 iff Tsk_{ij} is assigned to CN_k, otherwise 0
γ_{ik}	1 iff at least one task of B_i is assigned to CN_k, otherwise 0
Z_i	0 iff all tasks of B_i and B_{i-1} are assigned to a common node of the cluster, otherwise 1
Θ_{ik}	1 iff Z_i=1 and $\gamma_{ik} = 1$, otherwise 0
y_{ijk}	1 iff Z_i=1 and $\gamma_{ik} = 1$ and $x_{ijk} = 1$, otherwise 0
u_{ijk}	1 iff Z_{i+1}=1 and $\gamma_{ik} = 1$ and $x_{ijk} = 1$, otherwise 0
R_i	The time for transferring of required input data for B_i by CN_k from the shared storage of the cluster
WR_i	The time for transferring of output data of tasks of B_i by CN_k to the shared storage of the cluster
MS_i	The makespan of B_i
ST_i	The start time of executing of B_i
ET_i	The executional time of tasks of B_i

$$\Theta_{ik} \leq \frac{1}{2}(\gamma_{ik} + Z_i) \qquad \forall i \in \{2, \dots, n\}, \forall k \in \{1, \dots, m\} \tag{8}$$

$$\Theta_{ik} \geq (\gamma_{ik} + Z_i) - 1 \qquad \forall i \in \{2, \dots, n\}, \forall k \in \{1, \dots, m\} \tag{9}$$

$$y_{ijk} \leq \frac{1}{3}(x_{ijk} + \gamma_{ik} + Z_i) \qquad \forall i \in \{2, \dots, n\}, \forall j \in J, \forall k \in \{1, \dots, m\} \tag{10}$$

$$y_{ijk} \geq (x_{ijk} + \gamma_{ik} + Z_i) - 2 \qquad \forall i \in \{2, \dots, n\}, \forall j \in J, \forall k \in \{1, \dots, m\} \tag{11}$$

$$u_{ijk} \leq \frac{1}{3}(x_{ijk} + \gamma_{ik} + Z_{i+1}) \quad \forall i \in \{1, \dots, n-1\}, \forall j \in J, \forall k \in \{1, \dots, m\} \tag{12}$$

$$u_{ijk} \geq (x_{ijk} + \gamma_{ik} + Z_{i+1}) - 2 \quad \forall i \in \{1, \dots, n-1\}, \forall j \in J, \forall k \in \{1, \dots, m\} \tag{13}$$

$$R_i \geq \frac{IDS_i}{BW_k} \cdot \Theta_{ik} \qquad \forall i \in I_1, \forall k \in \{1, \dots, m\} \tag{14}$$

$$R_i \geq \frac{IDS_i}{BW_k} \cdot \sum_{j \in J_i} y_{ijk} \qquad \forall i \in I_2, \forall k \in \{1, \dots, m\} \tag{15}$$

$$WR_i \geq \left(\frac{ODS_i}{BW_k} \cdot \Theta_{ik} \right) \qquad \forall i \in I_1, \forall j \in J, \forall k \in \{1, \dots, m\} \tag{16}$$

$$WR_i \geq \left(\frac{ODS_i}{BW_k} \cdot \sum_{J} u_{ijk} \right) \qquad \forall i \in I_2, \forall k \in \{1, \dots, m\} \tag{17}$$

$$ET_i \geq \frac{\left(\sum_{j \in J_i} x_{ijk} \right) \cdot W_i}{P_k \cdot C_k} \qquad \forall i \in \{1, 2, \dots, n\}, \forall k \in \{1, 2, \dots, m\} \tag{18}$$

$$ST_{i+1} = ST_i + ET_i + R_i + WR_i \quad \forall i \in \{1, \dots, n-1\} \tag{19}$$

$$MS_i \geq ST_i + ET_i + R_i + WR_i \qquad \forall i \in \{1, \dots, n\} \tag{20}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n\}, \forall j \in J, \forall k \in \{1, 2, \dots, m\} \tag{21}$$

$$y_{ijk} \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n\}, \forall j \in J, \forall k \in \{1, 2, \dots, m\} \tag{22}$$

$$u_{ijk} \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n-1\}, \forall j \in J, \forall k \in \{1, 2, \dots, m\} \tag{23}$$

$$\gamma_{ik} \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n\}, \forall k \in \{1, 2, \dots, m\} \tag{24}$$

$$\Theta_{ik} \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n\}, \forall k \in \{1, 2, \dots, m\} \tag{25}$$

$$Z_i \in \{0, 1\} \qquad \forall i \in \{1, 2, \dots, n\} \tag{26}$$

$$MS_i \in \mathbb{R}^+ \qquad \forall i \in \{1, 2, \dots, n\} \tag{27}$$

$$ST_i \in \mathbb{R}^+ \qquad \forall i \in \{1, 2, \dots, n\} \tag{28}$$

$$ET_i \in \mathbb{R}^+ \qquad \forall i \in \{1, 2, \dots, n\} \tag{29}$$

$$WR_i \in \mathbb{R}^+ \qquad \forall i \in \{1, 2, \dots, n\} \tag{30}$$

$$R_i \in \mathbb{R}^+ \qquad \forall i \in \{1, 2, \dots, n\} \tag{31}$$

Below, the roles of the constraints and objective function of the model are explained in detail:

- The objective function, presented in Eq. (1), indicates the makespan of the last bag which is equal to the makespan of the workflow, including execution and data transferring times. The model minimizes this objective function.
- Constraints (2) state that each task must be assigned to only one node of the cluster.
- Constraints (3) and (4) guarantee that all tasks in B_i must be assigned. Precisely, $\gamma_{ik} = 1$ ($\gamma_{ik} = 0$) iff $\sum x_{ijk} > 0$ ($\sum x_{ijk} = 0$).

  If multiple tasks are assigned to a node, the node will execute them sequentially. Therefore, the memory constraint should be satisfied for each task individually. Constraints (5) guarantee that each task of each bag is assigned to the nodes that can satisfy its memory requirements.
- Constraints (6) and (7) show that if all tasks of B_i and B_{i-1} are assigned to a common node of the cluster, then $Z_i = 0$. Thus, data transfer time must be disregarded from the analysis.

  Remark: Constraints (6) and (7) can guarantee $Z_i = 1$ when tasks of bag_i and bag_{i-1} are assigned to different nodes and if they are assigned to the same node, both constraints lead to $Z_i \geq 0$. In mathematical models, some constraints are intentionally ignored because they are automatically satisfied in optimality, since the mathematical models obtain global optimal solution. In addition, this ignoring leads to a simplification of the problem and reduces the number of computations. The proposed model takes advantage of this property of mathematical models. More precisely, the constraints (8)– (13) are linearized forms of the decision variables Θ_{ik}, y_{ijk}, and u_{ijk}. On the other hand, the constraints (14)–(17) show the effect of the decision variable Z_i in the calculation of the data transfer time. Since *makespan* = *execution time* + *data transfer time*, the optimal value of the objective function leads to the minimization of data transfer time. Therefore, in order to keep optimality, when two bags i and $i-1$ are assigned to the same node, Z_i must be zero.

- Constraints (8) and (9) determine the value of decision variable Θ_{ik}. Indeed, $\Theta_{ik} = 1$ iff Z_i=1 and $\gamma_{ik} = 1$, otherwise $\Theta_{ik} = 0$. Precisely, these constraints are equivalent to the nonlinear equation: $\Theta_{ik} = \gamma_{ik} \cdot Z_i$.
- Constraints (10) and (11) determine the value of decision variable y_{ijk}. Indeed, $y_{ijk} = 1$ iff $Z_i = 1$ and $\gamma_{ik} = 1$ and $x_{ijk} = 1$, otherwise $y_{ijk} = 0$. Precisely, these constraints are equivalent to the nonlinear equation: $y_{ijk} = Z_i \cdot \gamma_{ik} \cdot x_{ijk}$.
- Constraints (12) and (13) determine the value of decision variable u_{ijk}. Indeed, $u_{ijk} = 1$ iff Z_{i+1}=1 and $\gamma_{ik} = 1$ and $x_{ijk} = 1$, otherwise $u_{ijk} = 0$. Precisely, these constraints are equivalent to the nonlinear equation: $u_{ijk} = Z_{i+1} \cdot \gamma_{ik} \cdot x_{ijk}$.
- Constraints (14) determine the value of data transfer time of input data for B_i which i is in $I1$. We divide the bag's index into two groups, I1 and I2—I1 for the bags that tasks have a same input data and I2 for others bags.
- Constraints (15) determine the value of data transfer time of input data for B_i which i is in $I2$.
- Constraints (16) determine the value of data transfer time of output data for B_i which i is in $I1$.
- Constraints (17) determine the value of data transfer time of output data for B_i which i is in $I2$.
- Constraints (18) determine the value of execution time of B_i. The tasks of B_i can be assigned to different nodes of the cluster simultaneously. $\frac{(\sum_{j\in J_i} x_{ijk})\cdot W_i}{P_k \cdot C_k}$ denotes just the execution time of those tasks assigned to CN_k. Thus, the execution time of B_i is max $\frac{(\sum_{j\in J_i} x_{ijk})\cdot W_i}{P_k \cdot C_k}$.
- Constraint (19) determines the value of start time of B_i and guarantees that the model fulfills the sequence of two sequential bags.
- Constraint (20) calculates the makespan of B_i.

A mathematical model related to a decision-making problem includes decision variables that represent the number of certain resources for use or the level of certain activities. The value of decision variables is specified during the problem-solving process. The decision variables in the model are defined in Eqs. (21)–(31).

For a cluster node having C_k processors(cores), parallel execution time T_p of a task t_i can be calculated using Amdahl's law [31]:

$$T_p(t_i, C_k) = \left(\alpha + \frac{1-\alpha}{C_k}\right) \cdot T_p(t_i, 1)$$

where $T_p(t_i, 1)$ is the execution time of t_i in single processor and, α is the fraction of t_i that cannot be parallelized. We assume that tasks of the workflow can be parallelized completely, so: $T_p(t_i, C_k) = \frac{T_p(t_i,1)}{C_k}$

5 Experimental results

The proposed mathematical model was implemented by CPLEX Solver[2] version 20.1.0. To perform experiments, we used a system with an *Intel core* i7-6500U CPU, a 2.5 GHz 8 GB RAM. We performed numerical experiments for the Epigenome, LIGO, Montage, and CyberShake workflows, whose specifications were taken from the Pegasus Workflow Generator.

In this section, the proposed approach is compared with related ones. We presented an example to illustrate the performance of the proposed MILP model and other related approaches. Then, two post-optimality analysis scenarios were presented. Finally, the running time of the MILP model and other related works were reported.

5.1 Comparing the proposed approach with related methods

For evaluating performance, we compare our approach to several related approaches. As previously stated, heuristic-based algorithms have been widely proposed to address the problem of DAWs scheduling in the past few decades [8, 9], therefore we selected well-known, efficient (HEFT, CPOP) and state-of-the-art (RBCA, DBCA, TDCA) heuristic algorithms to evaluate our approach. Moreover, these algorithms have been selected from all three groups of heuristic-based scheduling algorithms (list-based, clustering-based, and duplication-based).

For the comparison fairly, we assume the following for all approaches:

- The communication time is zero when two dependent task t_i and t_j are scheduled on the same node.
- Any task can be started only after completing all precedent tasks and receiving input data.
- Any tasks must be assigned to one node only once.
- All tasks are non-preemptive tasks.
- A computing node of the cluster can run only one single task at any time.

In the following, the algorithms HEFT, CPOP, RBCA, DBCA and TDCA are briefly described for comparison with our proposed MILP approach.

- **Heterogeneous Earliest Finish Time (HEFT):** The HEFT [14] algorithm is a well-known static and list-based algorithm for the workflow scheduling problem. It has two main phases: task prioritization phase and task assignment phase. In the task prioritization phase the algorithm computes the priority or rank of each task t_i using an upward rank value. $rank_u(t_i)$ calculated by Eq. 32 which indicates the length of the critical path from task t_i to the exit task where $\bar{W}_i$ is the average execution time of task t_i on all nodes, $\bar{C}_{i,j}$ is the

[2] https://www.ibm.com/products/ilog-cplex-optimization-studio.

average communication time between tasks t_i and t_j across all pairs of nodes and $rank_u(t_{\text{exit}}) = \bar{W}_{\text{exit}}$. In the task assignment phase, HEFT assigns each task to a node that allows completing the task the earliest. It is mentionable that the HEFT algorithm also uses an insertion policy in which tries to insert a task in at the earliest idle time between two already scheduled tasks on a computing node, if the slot is large enough to accommodate the task.

$$rank_u(t_i) = \bar{W}_i + \max_{t_j \in successor(t_i)} \{rank_u(t_j) + \bar{C}_{i,j}\} \tag{32}$$

- **Critical Path on Processor(CPOP):** The CPOP algorithm [14] has the same phases as the HEFT algorithm with different strategies. In the task prioritization phase, both the upward and downward rank values are considered. $rank_d(t_i)$ calculated by Eq. 33 represents the longest distance from the entry task to the task t_i, not including the computation cost of the t_i. The priority of each task t_i is assigned with the summation of upward and downward ranks $(rank_d(t_i) + rank_u(t_i))$. The length of the critical path is equal to the priority value of the entry task, and the entry task is selected as the first critical task. Then, the immediate successor task of the selected task that has the highest priority value is selected as the next critical task. This process continues until the exit task. In the computing node selection phase, the critical path node is the node that minimizes the computation time of all tasks on the critical path. A non-critical task is assigned on a node that minimizes the earliest execution finish time of the task with considering the insertion scheduling policy.

$$rank_d(t_i) = \max_{t_j \in predecessor(t_i)} \{rank_d(t_j) + \bar{W}_j + \bar{C}_{i,j}\} \tag{33}$$

- **Runtime Balance Clustering Algorithm (RBCA):** The RBCA algorithm [22] is one of the most novel static and cluster-based algorithms for the workflow scheduling problem. Although cloud computing is considered as the infrastructure, this approach is comparable with our approach, because, like our approach, the objective of RBSA is minimizing the overall makespan of the workflow. It also applies horizontal task clustering for a workflow that merges the tasks of the same horizontal level in a workflow into a specific number of task clusters. In [22], it has been explained that runtime imbalance occurs when the tasks are assigned to the different task clusters with a large difference in execution time. In such a status, the execution of task clusters on the next level of the workflow will be delayed, thus affecting the overall makespan of the workflow. RBCA tries to balance the run time of defined task clusters using a backtracking strategy. It minimizes the value of *TBM*(Time Balance Measurement) metric. *TBM* is calculated by Eq. 34 which represents the maximum summation of runtime among defined task clusters where the tasks of a level of the workflow are clustered into m task cluster $(C_1, \dots, C_m)$; $taskTimeSum(C_i)$ represents the sum of runtime of all tasks in cluster C_i. RBCA chooses the number of compute nodes as the number of task clusters.

$$TBM = \max\{taskTimeSum(C_i)\} \qquad i = 1, \dots, m \tag{34}$$

- **Dependency Balance Clustering Algorithm (DBCA):** The DBCA algorithm is a new static and cluster-based algorithm presented in [22] like RBCA. It also applies horizontal task clustering and considers the number of compute nodes as the number of task clusters. DBCA tries to tackle over dependency imbalanced problem. The dependency imbalance may occur when a task requires the data from other tasks clustered in different task clusters. This delays the release of following tasks and thus increases the overall makespan. To avoid the dependency imbalance, DBCA clusters the tasks with high dependency correlation(similarity of two tasks in terms of data dependency) together.

  The dependency correlation between two tasks t_i and t_j is calculated by Eq. 35 where $succ(t_i)$ represents the set of immediate successors of task t_i, and $|succ(t_i)|$ represents the number of immediate successors of task t_i.

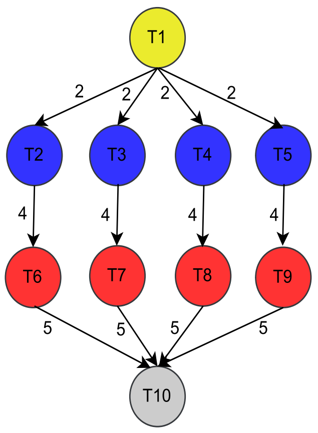
$$\begin{aligned} cor(t_i, t_j) &= \frac{|succ(t_i) \cap succ(t_j)|}{\sqrt{|succ(t_i) \cdot succ(t_j)|}} \\ &= \sqrt{\frac{|succ(t_i) \cap succ(t_j)|}{|succ(t_i)|}} \cdot \sqrt{\frac{|succ(t_i) \cap succ(t_j)|}{|succ(t_j)|}} \end{aligned} \tag{35}$$

- **Task Duplication-based Clustering Algorithm (TDCA):** The TDCA algorithm [23] includes four phases which is an improved extension of the TANH algorithm [32]. In the first phase, it assigns tasks to computing nodes to generate several initial clusters. In the second phase, a task duplication method is applied to modify the initial clusters to reduce the makespan, so new clusters are added if necessary. In the third phase, TDCA merges some clusters and decreases the number of occupied computing nodes to obtain a preliminary schedule. Finally, in the last phase, a task insertion scheme, introduced in DCPD [33], is adopted which inserts a task to an idle time slot located before its successor tasks if this insertion could make the successor start earlier.

5.1.1 An illustrative example

To illustrate the performance of the proposed approach compared to related work, we use the workflow in Fig. 3, which is roughly similar in structure to the Epigenome workflow. Assume that the cluster infrastructure consists of three compute nodes N1, N2, and N3 with different computing power. The average data transfer time required between two tasks is shown by the values on the edges. The execution time of the tasks on each of the nodes in the cluster is also shown. Figure 4 shows how our MILP approach and the heuristic algorithms schedule the workflow of Fig. 3. It also shows the scheduling length (makespan).

Regarding the upward ranking of the HEFT, tasks of the same level have the same priority. Therefore, the tasks of a level can be prioritized in ascending order



Computation Times

	N1	N2	N3
T1	90	100	50
T2, T3, T4, T5	45	75	25
T6, T7, T8, T9	35	70	20
T10	70	100	40

Fig. 3 A workflow with 10 tasks and computation time of tasks on three nodes of a cluster

of their number. So, the scheduling order of the tasks for the HEFT algorithm is $T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9, T_{10}$.

The CPOP algorithm identifies four critical paths of equal weight which are $\{T_1 - T_2 - T_6 - T_{10}\}$, $\{T_1 - T_3 - T_7 - T_{10}\}$, $\{T_1 - T_4 - T_8 - T_{10}\}$, and $\{T_1 - T_5 - T_9 - T_{10}\}$. On the other hand, the critical node is N_3 because it minimizes the execution time of the tasks on the critical path. All tasks are scheduled on N_3 and the makespan is 270 if all four paths are considered critical path. If only one of them is considered the critical path, the makespan is 225.

The DBCA and RBCA algorithms give the same result. This is because the levels in the DAG are completely homogeneous. This homogeneity covers both the runtime and the dependencies of the tasks. Since they cluster and assign the tasks of each level of the DAG based on the number of nodes, all nodes are used.

The TDCA algorithm assigns all tasks to node N3. Since N3 is better than the other nodes for all tasks, all schedules from the other nodes are merged onto the

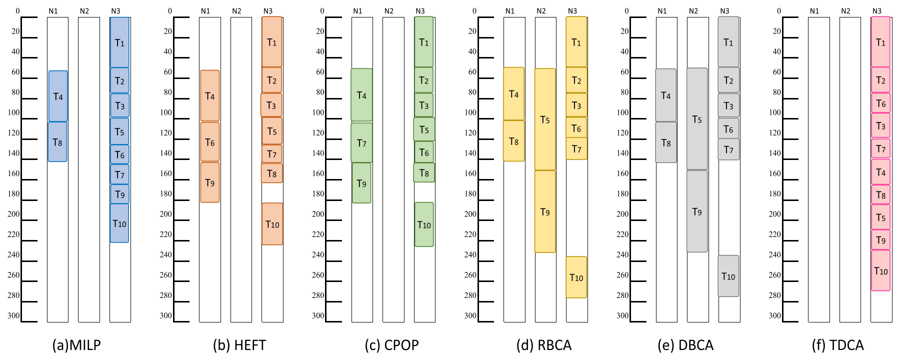


Fig. 4 Scheduling of workflow in Fig. 3 with **a** MILP (makespan=211), **b** HEFT(makespan=222), **c** CPOP (makespan=225), **d** RBCA (makespan=242), **e** DBCA (makespan=242), **f** TDCA (makespan=270)

Table 2 Obtained makespan of different approaches for scientific workflows on the small cluster

Approach	Epigenome			LIGO			Montage			Cybershake		
	24	47	100	30	50	100	25	50	100	30	50	100
MILP	**149.4**	262.23	**531.96**	**16.895**	28.51	**47.53**	**1.02**	**2**	3.35	**7.56**	**8.78**	14.42
HEFT	155.73	**261.79**	536.2	17.76	**28.39**	50.26	1.47	2.06	**3.17**	8.54	9.86	**13.05**
CPOP	**149.4**	262.55	534.82	17.76	**28.39**	53.52	1.53	2.23	3.54	8.82	10.33	14.46
RBCA	181.54	342.33	984.6	18.99	47.3	94.58	1.68	2.97	5.19	12.26	13.09	18.6
DBCA	181.54	342.33	984.6	18.99	47.3	94.58	1.68	2.97	5.19	12.26	13.09	18.6
TDCA	181.97	382.63	643.39	21.27	30.08	57.12	1.54	2.35	3.46	13.43	13.79	16.01

The bold values mean the best results

best node. The algorithm may work better for cluster infrastructures where no single node is best for all tasks, so different tasks may have different preferred nodes.

5.1.2 Comparison results

To compare the MILP model with other approaches, we performed experiments on the Epigenome, LIGO, Montage, and CyberShake workflows with four workflow sizes (about 30, 50, 100, and 1000 tasks). We looked into three cluster sizes, namely small, medium, and large (consist of 4, 8, and 16 computing nodes, respectively) with the following parameters:

- The small cluster:
  $P[CN_k] = \{15, 10, 25, 20\}$
  $C[CN_k] = \{10, 6, 4, 8\}$
  $BW[CN_k] = 1000$
  $mem[CN_k] = \{5000, 2000, 4000, 8000\}$
- The medium cluster:
  $P[CN_k] = \{10, 25, 15, 15, 30, 12, 10, 30\}$
  $C[CN_k] = \{14, 4, 10, 8, 4, 12, 16, 6\}$
  $BW[CN_k] = 1000$
  $mem[CN_k] = \{1000, 1050, 5000, 2000, 4000, 8000, 6000, 3000\}$
- The large cluster:
  $P[CN_k] = \{10, 25, 15, 15, 30, 12, 10, 30, 25, 10, 15, 30, 25, 12, 10, 20\}$
  $C[CN_k] = \{14, 4, 10, 8, 14, 12, 16, 6, 4, 12, 8, 6, 8, 10, 14, 8\}$
  $BW[CN_k] = 1000$
  $mem[CN_k] = \{1000, 1050, 5000, 2000, 4000, 8000, 6000, 3000, 1100, 1250, 5500, 2300, 4300, 8100, 6200, 3900\}$

The numerical results are provided in Tables 2, 3, 4 and Figs. 5, 6, 7. It should be noted that the all bars in Figs. 5, 6, 7 are extracted from the data in Tables 2, 3, 4 which are normalized using the maximum value of makespan for different workflow sizes.

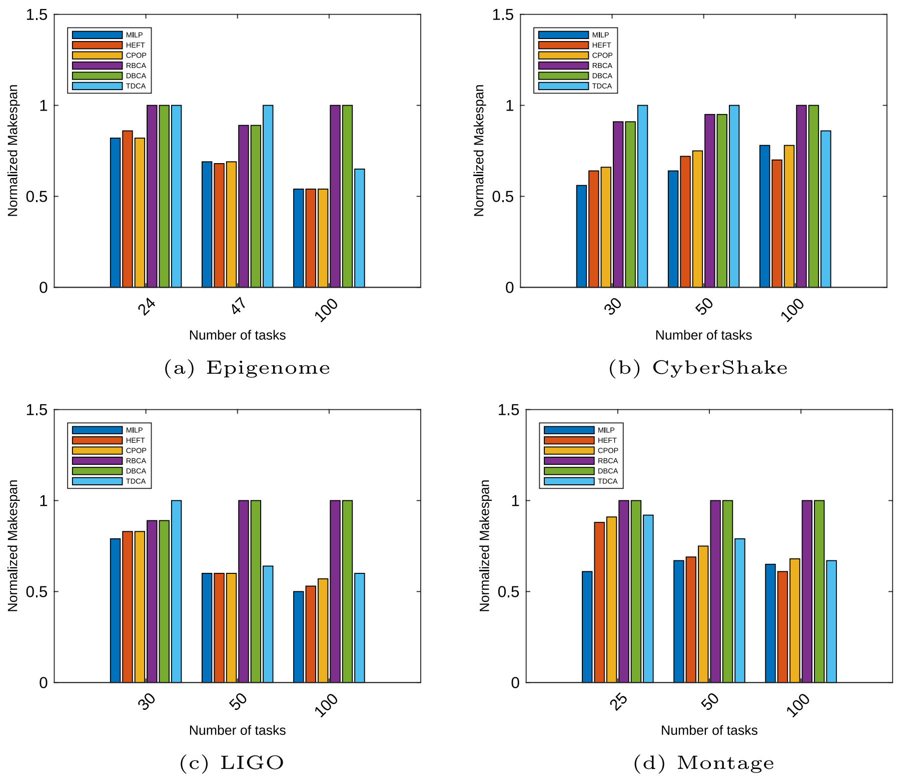


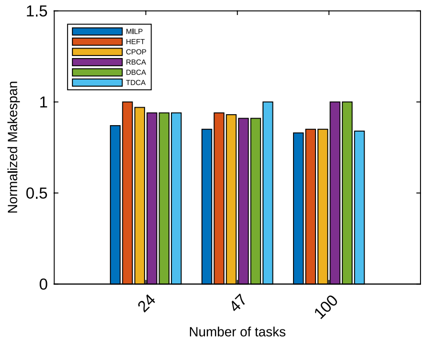
Fig. 5 Normalized makespan for the workflows with different sizes on the small cluster

Table 3 Obtained makespan of different approaches for scientific workflows on the medium cluster

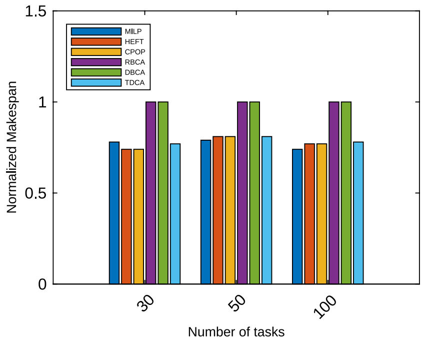
Approach	Epigenome			LIGO			Montage			Cybershake		
	24	47	100	30	50	100	25	50	100	30	50	100
MILP	**81.5**	**129.43**	**255.03**	8.05	**11.05**	**19.06**	**0.94**	**1.2**	**1.21**	7.95	**8.32**	**8.55**
HEFT	93.36	144.26	262.48	8.06	13.23	23.66	1.15	1.37	1.58	**7.47**	8.49	8.98
CPOP	90.4	142.06	262.48	**7.97**	13.21	24.82	1.15	1.47	2.08	**7.47**	8.49	8.99
RBCA	87.44	139.13	307.71	**7.97**	13.21	28.42	1.15	1.42	2.09	10.15	10.52	11.62
DBCA	87.44	139.13	307.71	**7.97**	13.21	28.42	1.15	1.42	2.09	10.15	10.52	11.62
TDCA	87.5	152.79	259.61	8.77	14.53	28.27	1.08	1.48	2.2	7.84	8.49	9.09

The bold values mean the best results

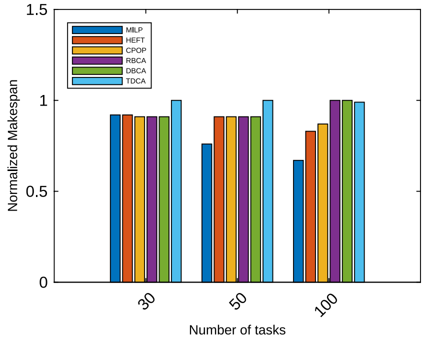
As previously stated in Sect. 2, in contrast to heuristic methods, we assumed that the workflows are in the form of BoT. Therefore, the heuristic methods are expected to give a lower makespan than that of our proposed method. But as seen in the experimental results of makespan reported in Tables 2, 3 and 4, in 75% of cases, our



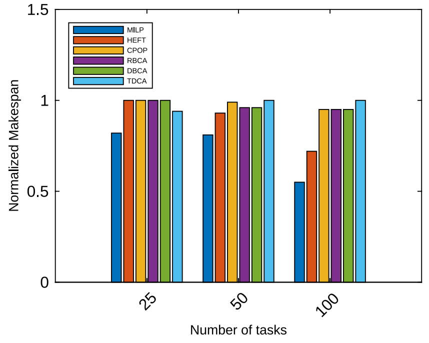
(a) Epigenome



(b) CyberShake



(c) LIGO



(d) Montage

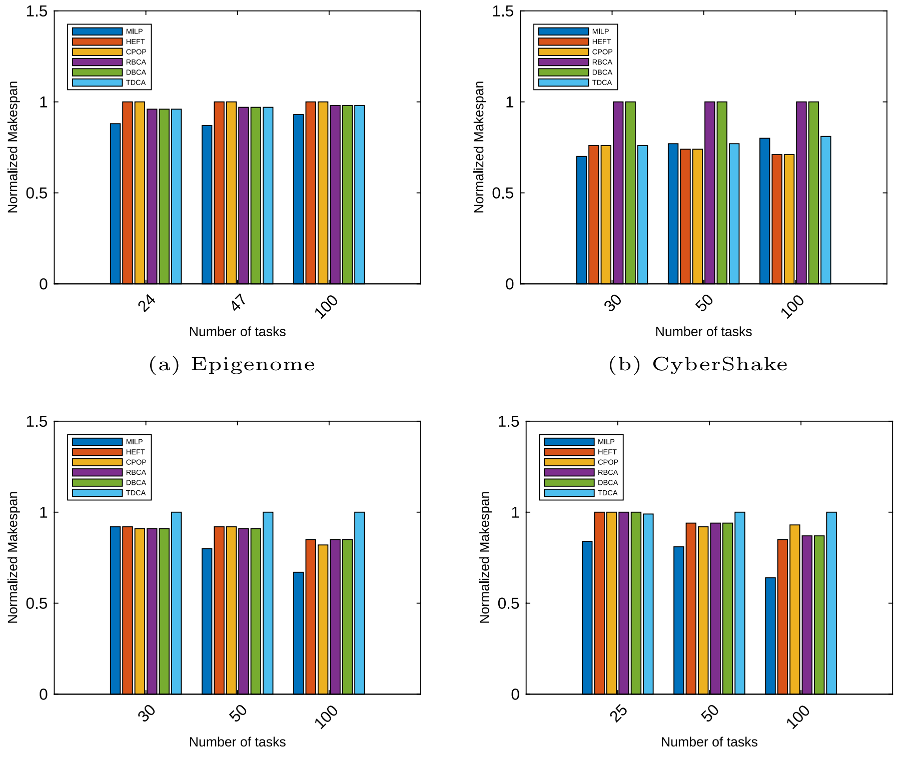
Fig. 6 Normalized makespan for the workflows with different sizes on the medium cluster

Table 4 Obtained makespan of different approaches for scientific workflows on the large cluster

Approach	Epigenome			LIGO			Montage			Cybershake		
	24	47	100	30	50	100	25	50	100	30	50	100
MILP	**70.05**	**87.39**	**146.32**	6.74	**7.05**	**10.65**	**0.88**	**1.01**	**1.04**	**6.17**	7.02	7.58
HEFT	79.95	100.04	157.38	6.71	8.06	13.54	1.05	1.17	1.38	6.72	**6.73**	**6.74**
CPOP	79.95	100.04	157.38	**6.65**	8.04	13.09	1.05	1.15	1.51	6.72	**6.73**	**6.74**
RBCA	76.98	97.17	154.42	**6.65**	7.96	13.59	1.05	1.17	1.42	8.81	9.14	9.48
DBCA	76.98	97.17	154.42	**6.65**	7.96	13.59	1.05	1.17	1.42	8.81	9.14	9.48
TDCA	77.03	97.19	154.49	7.31	8.76	15.99	1.04	1.25	1.63	6.72	7.08	7.69

The bold values mean the best results

approach outperforms to other approaches. It should be noted that in almost all other cases (except one), our approach is ranked second in terms of makespan.



(a) Epigenome (b) CyberShake

(c) LIGO (d) Montage

Fig. 7 Normalized makespan for the workflows with different sizes on the large cluster

5.2 Post-optimality analysis

In the theory of decision-making, although optimizing the objective function is the main issue, it is not necessarily the end of the story: employing a post-analysis can optimize other parameters. In this section, we present two interesting cases of post-optimality analysis.

5.2.1 Choosing a preferred solution among the set of alternative optimal solutions

As a post-optimality analysis, choosing a preferred solution among the set of alternative optimal solutions referred to as the preferred optimal solution is important from the decision-maker's perspective.

In our case, this can be done in the form of Phase 2 optimization. After solving Model MILP in the first phase and calculating the optimal value of makespan (denoted by t^*), in phase 2 we optimize the objective function expressing the secondary preferences on the set of alternative optimal phase solutions of Phase 1.

Phase 1: Solving the MILP model as follows:

$$\begin{aligned} t^* &= \min MS_n \\ &\text{subject to:} \qquad Eqs.\ (2)-(31) \end{aligned}$$

One of these secondary preferences (which can be important here) is to choose a type of assignment in which the least possible number of nodes are used while getting the best makespan. In a cluster environment that includes a limited number of computing nodes, the fewer nodes we utilize to run applications, the less energy will be consumed, and the more users can start running their applications. For this purpose, for each node CN_k, we define a binary variable η_k as follows:

$$\eta_k = \begin{cases} 1 & if \quad \sum_i \sum_j x_{ijk} > 0 \\ 0 & O.W \end{cases} \tag{36}$$

According to the definition of decision variables in Phase 1, we want to hold the following proposition for each index k:

$$\sum_i \sum_j x_{ijk} > 0 \Longleftrightarrow \eta_k = 1 \qquad \forall k \in \{1, 2, ..., m\} \tag{37}$$

It can be shown that in optimality (in optimal solutions) of the following model the above-mentioned proposition holds:

Phase 2: Solving the MILP-ON model

$$\text{(MILP-ON model)} \quad \min \sum_k \eta_k \tag{38}$$

$$\begin{aligned} &\text{subject to:} \qquad Eqs.\ (2)-(31) \\ &MS_n \leqslant t^* + \epsilon \end{aligned} \tag{39}$$

$$\sum_i \sum_j x_{ijk} \le M \cdot \eta_k \qquad \forall k \in \{1, 2, ..., m\} \tag{40}$$

$$\eta_k \in \{0, 1\} \qquad\qquad \forall k \in \{1, 2, ..., m\}, \tag{41}$$

where M is a large enough positive number that its value can be any number greater than or equal to the number of bags of the workflow multiplied by the number of tasks. We define binary quantity η_{ik} in the sense that $\eta_{ik} = 1$ if only if at least one task of B_i is assigned to the CN_k equivalently to Eq. 42. Also, we define positive integer quantity NN_i as Eq. 43 that calculates the number of used nodes of the cluster for each B_i.

$$\sum_j x_{ijk} > 0 \Longleftrightarrow \eta_{ik} = 1 \qquad \forall i \in \{1, 2, ..., n\}, \forall k \in \{1, 2, ..., m\} \tag{42}$$

$$NN_i = \sum_k \eta_{ik} \qquad \forall i \in \{1, 2, ..., n\} \tag{43}$$

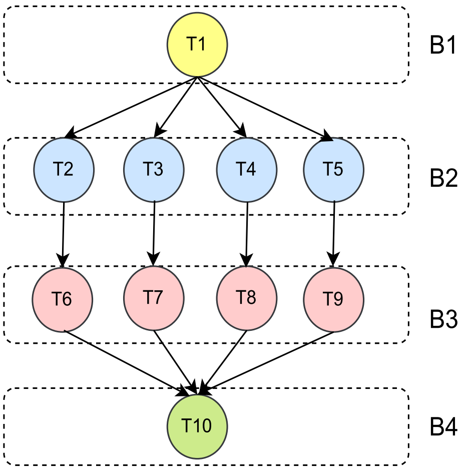
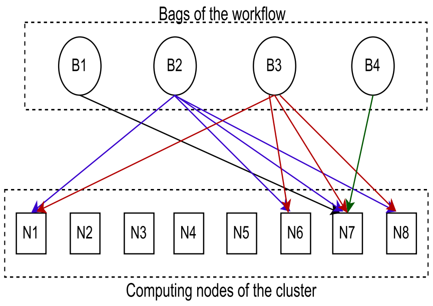
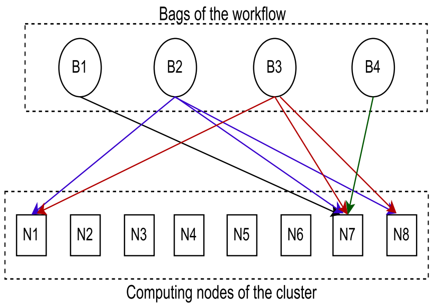


Fig. 8 A workflow with 10 tasks and four bag-of-tasks



(a) Optimal assignment of Phase1



(b) Optimal assignment of Phase1

Fig. 9 Two different optimal assignments with the same makespan

We conducted an experiment for the workflow of Fig. 8 which has four bags ($B_1, ..., B_4$). The simulated cluster environment consists of 8 computing nodes ($k = 8$). Parameters of the DAW and computing nodes are as the following:

- $P[CN_k] = \{100, 80, 60, 80, 40, 100, 200, 190\}$
- $C[CN_k] = 1$
- $BW[CN_k] = 1000$
- $mem[CN_k] = \{1000, 1050, 5000, 2000, 4000, 8000, 6000, 3000\}$
- $W[T_i] = \{1000, 500, 500, 500, 500, 400, 400, 400, 400, 800\}$
- $Mem[T_i] = \{5000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 1000, 4000\}$
- $IDS[T_i] = \{0, 10, 10, 10, 10, 20, 20, 20, 20, 20\}$
- $ODS[T_i] = \{10, 20, 20, 20, 20, 20, 20, 20, 20, 0\}$

Solving the MILP model in Phase 1, the optimal makespan is 18.05. In the obtained optimal solution, assignments are as depicted in Fig. 9a. Solving the

MILP-ON model in Phase 2, assignments of the optimal solution are as depicted in Fig. 9b with the same makespan. In the second alternative optimal solution, the number of consumed nodes for B_2 and B_3 is three which is less than that of the first optimal solution(four nodes). In general, not all the optimal solutions of phase 1 might be obtained, and indeed, it is not necessary to do that. However, in the second phase, the most preferred optimal solution can be found.

5.2.2 Relaxation of the makespan in the hope of significantly reducing the number of consumed nodes

Sometimes in decision-making, slight appeasement in one goal (even if it is our main goal) may lead to a significant improvement in other goals or preferences. In such cases, by relaxing(increasing here) the mentioned objective function at different levels, various values are obtained for other goals and preferences. These results show a kind of trade-off between considered goals. The results of this trade-off are provided for the decision-maker in the form of a table called the trade-off table for making a preferred decision. There are two main motivations for relaxing the makespan:

Table 5 Makespan relaxation for Epigenome workflow ($t^* = 97.85$)

The number of consumed nodes in each bag(level).

	Relaxed Makespan						
#Bag	97.85	97.90	97.95	103	110	130	163
1	1	1	1	1	1	1	1
2	6	1	1	1	1	1	1
3	6	3	1	1	1	1	1
4	6	6	6	5	4	3	2
5	6	6	6	5	4	3	2
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1
Sum	28	20	18	16	14	12	10

b Makespan relaxation VS. number of consumed node improvement.

Δ	$t^* + \Delta$	$\eta(\%)$	$\theta(\%)$	Gain-to-Loss $(\frac{\theta}{\eta})$
0.05	97.9	0.05	28.57	559.14
0.1	97.95	0.10	35.71	349.46
5.15	103	5.26	42.86	8.14
12.15	110	12.42	50.00	4.03
32.15	130	32.86	57.14	1.74
65.15	163	66.58	64.29	0.97

1. As makespan increases, the number of occupied nodes decreases; consequently, the amount of energy consumption decreases.
2. Increasing the makespan reduces the number of nodes occupied, so more users can run their programs.

We conducted experiments on the Epigenome, LIGO, Montage, and CyberShake workflows with 100 tasks. The simulated cluster environment consists of sixteen compute nodes with the following parameters:

- $P[CN_k] = \{15, 10, 20, 5, 10, 20, 30, 25, 18, 26, 17, 25, 30, 21, 10, 30\}$
- $C[CN_k] = \{20, 32, 28, 16, 10, 30, 32, 8, 12, 28, 24, 12, 18, 6, 8, 16\}$
- $mem[CN_k] = \{50, 100, 200, 850, 150, 100, 250, 300, 50, 100, 200, 850, 150, 100, 250, 300\}$
- $BW[CN_k] = 1000$

The numerical results are provided in Tables 5, 6, 7, 8. t^* denotes the optimal makespan obtained by solving the MILP model, θ is the percentage of reduction in the number of nodes consumed, η is the percentage of increase in the makespan and Δ is the amount of relaxing the makespan. From the numerical results reported in Tables 5, 6, 7, 8, the following points can be highlighted:

Table 6 Makespan relaxation for LIGO workflow ($t^* = 59.98$)

(a) The number of consumed nodes in each bag(level).

	Relaxed Makespan						
#Bag	59.98	60.30	60.80	62.30	65	70	76
1	2	2	2	1	1	1	1
2	2	2	2	2	2	2	1
3	2	2	1	1	1	1	1
4	2	2	2	2	1	1	1
5	2	2	2	2	2	1	1
6	2	1	1	1	1	1	1
Sum	12	11	10	9	8	7	6

(b) Makespan relaxation VS. number of consumed node improvement.

Δ	$t^* + \Delta$	$\eta(\%)$	$\theta(\%)$	Gain-to-Loss ($\frac{\theta}{\eta}$)
0.32	60.3	0.53	8.33	15.62
0.82	60.8	1.37	16.6	12.19
2.32	62.3	3.87	25	6.46
5.02	65	8.37	33.33	3.98
10.02	70	16.71	41.67	2.49
16.02	76	26.71	50	1.87

- A very small relaxation in makespan leads to a considerable reduction in the number of consumed nodes. For example in Table 5b just relaxing 0.05 percent in makespan leads to a 28.57 percent reduction in the number of consumed nodes. This fact is depicted in Fig 10. Thus, the ratio of gain (saved nodes) to loss (increasing makespan) is 559.14.
- As shown in Tables 5, 6, 7, 8, as the amount of relaxation in the makespan increases, the rate of improvement in the number of nodes consumed decreases so that a more value of gain-to-loss ratio occurs near the optimal value of makespan.

5.3 Running time

In this subsection, we express the running time of the MILP, HEFT, CPOP, RBCA, DBCA and TDCA approaches. Table 9 shows the running time of the approaches for the Epigenome, Montage, Cybershake and LIGO workflows for a different number of tasks. As can be seen, in most cases, the running time of the TDCA is better than the other ones. Moreover, the running time of the MILP approach is about a second in most cases while the running time of heuristic approaches is less than one second in all cases. Nevertheless, the running time of MILP is negligible compared with the workflow execution time.

Table 7 Makespan relaxation for Cybershake workflow ($t^* = 49.50$)

(a) The number of consumed nodes in each bag(level).

	Relaxed Makespan						
#Bag	49.50	49.51	49.52	49.53	49.56	49.90	52
1	2	2	2	2	2	1	1
2	2	2	2	2	2	2	1
3	1	1	1	1	1	1	1
4	9	5	3	2	1	1	1
5	1	1	1	1	1	1	1
Sum	15	11	9	8	7	6	5

(b) Makespan relaxation VS. number of consumed node improvement.

Δ	$t^* + \Delta$	$\eta(\%)$	$\theta(\%)$	Gain-to-Loss ($\frac{\theta}{\eta}$)
0.01	49.51	0.02	26.67	1320
0.02	49.52	0.04	40	990
0.03	49.53	0.06	46.67	770
0.06	49.56	0.12	53.33	440
0.4	49.9	0.81	60	74.25
2.5	52	5.05	66.67	13.2

Table 8 Makespan relaxation for Montage workflow ($t^* = 0.46$)

(a)The number of consumed nodes in each bag(level).

	Relaxed Makespan						
#Bag	0.46	0.48	0.50	0.53	0.55	0.60	0.70
1	10	9	7	6	5	4	3
2	10	9	7	6	5	4	3
3	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1
Sum	27	25	21	19	17	15	13

(b)Makespan relaxation VS. number of consumed node improvement.

Δ	$t^* + \Delta$	$\eta(\%)$	$\theta(\%)$	Gain-to-Loss ($\frac{\theta}{\eta}$)
0.02	0.48	4.35	7.41	1.70
0.04	0.5	8.70	22.22	2.56
0.07	0.53	15.22	29.63	1.95
0.09	0.55	19.57	37.04	1.89
0.14	0.6	30.43	44.44	1.46
0.24	0.7	52.17	51.85	0.99
0.34	0.8	73.91	59.26	0.80

6 Conclusion and future works

We proposed a mathematical model (MILP model) for BoT scientific workflow scheduling in heterogeneous cluster environments which minimizes the makespan. The problem was formulated as an MILP model and solved by the CPLEX solver in a reasonable time. In a mathematical model, the objective function and the problem's constraints are expressed using some explicit mathematical expressions of decision variables.

This provides a useful insight into the structure of the problem. Some critical information, such as post-optimality analysis or trade-offs, can be derived explicitly. We compared our approach against related approaches. In our proposed approach, it is assumed that the workflows are in the form of BoT; however, there is no such assumption in comparative heuristic methods and they can be used in the most general form of workflows. Therefore, heuristic approaches are expected to give a lower makespan than that of our proposed approach, but since they give sub-optimal solutions, this does not happen in most cases, so that our approach outperforms other approaches in 75% of the cases. It should be noted that in almost all other cases (except one), our approach is ranked second in terms of makespan.

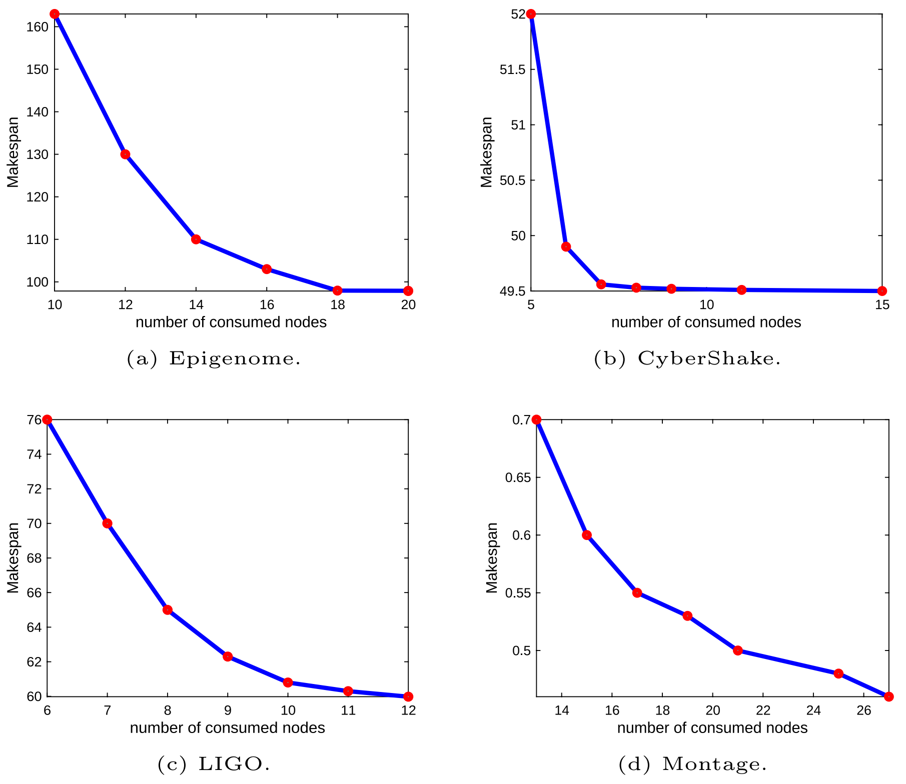


Fig. 10 The effect of makespan relaxation on the total number of consumed nodes

Table 9 Running time of the approaches for the workflows in seconds with different sizes

Approach	Epigenome			LIGO			Montage			Cybershake		
	24	47	100	30	50	100	25	50	100	30	50	100
MILP	1	2.01	9	0.854	0.870	0.900	1	1.245	6	0.789	0.923	1.2
HEFT	0.167	0.218	0.220	0.054	0.335	**0.001**	0.248	0.105	**0.001**	0.130	0.453	0.247
CPOP	0.096	0.144	0.315	0.076	0.121	0.239	0.411	0.182	**0.001**	0.231	0.174	0.327
RBCA	**0.017**	0.029	0.046	0.022	0.027	0.054	0.133	0.050	0.169	0.027	0.025	0.047
DBCA	0.025	0.046	0.100	0.037	0.054	0.103	0.076	0.140	0.854	0.043	0.069	0.182
TDCA	0.893	**0.001**	**0.013**	**0.003**	**0.008**	0.025	**0.003**	**0.007**	0.035	**0.002**	**0.006**	**0.028**

The bold values mean the best results

We also investigated choosing a preferred solution among the set of (possible) alternative optimal solutions as a part of the post-optimality analysis. As another part of the post-optimality analysis, relaxation of the makespan is explored in the hope of significantly reducing the number of consumed nodes.

The proposed mathematical model also provides an explicit framework for imposing any possible secondary preferences which have not been considered here and may be needed in the future. In the future, we intend to extend our proposed approach to take into account communication congestion and study its effect on the scheduling of a workflow in heterogeneous cluster environments.

Acknowledgements Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as FONDA (Project 414984028, SFB 1404).

Funding Open Access funding enabled and organized by Projekt DEAL. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as FONDA (Project 414984028, SFB 1404).

Availability of data and materials Not applicable.

Declarations

Ethics approval and consent to participate Not applicable.

Conflict of interest All authors declare that they have no conflict of interest.

Author contributions SM contributed to conceptualization, methodology, software, validation, formal analysis, and writing—original draft. LP contributed to conceptualization, methodology, formal analysis, and writing—original draft. FD contributed to software and validation. NDM contributed to validation and writing—review & editing. UL contributed to writing—review & editing, and investigation. KR contributed to formal analysis, resources, and writing—review & editing.

References

1. Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K (2013) Characterizing and profiling scientific workflows. Fut Gener Comput Syst 29(3):682–692
2. Abdi S, PourKarimi L, Ahmadi M, Zargari F (2018) Cost minimization for bag-of-tasks workflows in a federation of clouds. J Supercomput 74:2801–2822
3. Mohammadi S, Pedram H, PourKarimi L (2018) Integer linear programming-based cost optimization for scheduling scientific workflows in multi-cloud environments. J Supercomput 74:4717–4745
4. Bader J, Thamsen L, Kulagina S, Will J, Meyerhenke H, Kao O (2021, December) Tarema: Adaptive resource allocation for scalable scientific workflows in heterogeneous clusters. In: 2021 IEEE International Conference on Big Data (Big Data), pp 65-75. IEEE
5. Weiss G (1995) Scheduling: Theory, algorithms, and systems.JSTOR
6. 4 Key Advantages of Using Mathematical Optimization Instead of Heuristics, 2020 April, https://www.gurobi.com/resources/4-key-advantages-of-using-mathematical-optimization-instead-of-heuristics/, last accese:05/04/2023

7. Taha HA (2014) Integer programming: theory, applications, and computations. Academic Press
8. Versluis L, Iosup A (2021) A survey of domains in workflow scheduling in computing infrastructures: Community and keyword analysis, emerging trends, and taxonomies. Fut Generat Comput Syst 123:156–177
9. Wu F, Wu Q, Tan Y (2015) Workflow scheduling in cloud: a survey. J Supercomput 71:3373–3418
10. Bader J, Lehmann F, Groth A, Thamsen L, Scheinert D, Will J, ... Kao O (2022, November) Reshi: Recommending resources for scientific workflow tasks on heterogeneous infrastructures. In: 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC), pp 269-274. IEEE
11. Selvi S, Manimegalai D (2017) DAG scheduling in heterogeneous computing and grid environments using variable neighborhood search algorithm. Appl Artif Intell 31(2):134–173
12. He S, Wu J, Wei B, Wu J (2023) Algorithms for tree-shaped task partition and allocation on heterogeneous multiprocessors. J Supercomput 1-31
13. Maurya AK, Tripathi AK (2018) On benchmarking task scheduling algorithms for heterogeneous computing systems. J Supercomput 74(7):3039–3070
14. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parall Distribute Syst 13(3):260–274
15. Arabnejad H, Barbosa JG (2013) List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans Parall Distribute Syst 25(3):682–694
16. Djigal H, Feng J, Lu J, Ge J (2020) IPPTS: an efficient algorithm for scientific workflow scheduling in heterogeneous computing systems. IEEE Trans Parall Distrib Syst 32(5):1057–1071
17. Zhou N, Qi D, Wang X, Zheng Z, Lin W (2017) A list scheduling algorithm for heterogeneous systems based on a critical node cost table and pessimistic cost table. Concurr Comput Pract Exper 29(5):e3944
18. Wu CG, Wang L, Wang JJ (2021) A path relinking enhanced estimation of distribution algorithm for direct acyclic graph task scheduling problem. Knowl Based Syst 228:107255
19. Jedari B, Dehghan M (2009) Efficient DAG scheduling with resource-aware clustering for heterogeneous systems. Comput Inf Sci 2009:249–261
20. Wang H, Sinnen O (2018) List-scheduling versus cluster-scheduling. IEEE Trans Parall Distrib Syst 29(8):1736–1749
21. Kanemitsu H, Hanada M, Nakazato H (2016) Clustering-based task scheduling in a large number of heterogeneous processors. IEEE Trans Parall Distrib Syst 27(11):3144–3157
22. Yu D, Ying Y, Zhang L, Liu C, Sun X, Zheng H (2020) Balanced scheduling of distributed workflow tasks based on clustering. Knowl Based Syst 199:105930
23. He K, Meng X, Pan Z, Yuan L, Zhou P (2018) A novel task-duplication based clustering algorithm for heterogeneous computing environments. IEEE Trans Parall Distrib Syst 30(1):2–14
24. Bader J, Lehmann F, Thamsen L, Will J, Leser U, Kao O (2022, July) Lotaru: Locally estimating runtimes of scientific workflow tasks in heterogeneous clusters. In: Proceedings of the 34th International Conference on Scientific and Statistical Database Management pp 1-12
25. Sukhoroslov O (2021) Toward efficient execution of data-intensive workflows. J Supercomput 77(8):7989–8012
26. Sukhoroslov O (2019) An experimental study of data transfer strategies for execution of scientific workflows. In: Parallel Computing Technologies: 15th International Conference, PaCT 2019, Almaty, Kazakhstan, August 19-23, 2019, Proceedings 15, pp 67-79. Springer International Publishing
27. Bharathi S, Chervenak A, Deelman E, Mehta G, Su MH, Vahi K (2008, November) Characterization of scientific workflows. In 2008 third workshop on workflows in support of large-scale science pp 1-10. IEEE
28. Costa LB, Yang H, Vairavanathan E, Barros A, Maheshwari K, Fedak G, Al-Kiswany S (2015) The case for workflow-aware storage: an opportunity study. J Grid Comput 13:95–113
29. Minh TN, Nam T, Epema DH (2013) Parallel workload modeling with realistic characteristics. IEEE Trans Parall Distrib Syst 25(8):2138–2148
30. Malawski M, Figiela K, Bubak M, Deelman E, Nabrzyski J (2015) Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. Scientif Program 2015:5–5
31. Rodgers DP (1985) Improvements in multiprocessor system design. ACM SIGARCH Comput Architecture News 13(3):225–231
32. Bajaj R, Agrawal DP (2004) Improving scheduling of tasks in a heterogeneous environment. IEEE Trans Parall Distrib Syst 15(2):107–118

33. Liu CH, Li CF, Lai KC, Wu CC (2006, July) A dynamic critical path duplication task scheduling algorithm for distributed heterogeneous computing systems. In: 12th International Conference on Parallel and Distributed Systems-(ICPADS'06), Vol. 1, pp 8-pp. IEEE

Authors and Affiliations

Somayeh Mohammadi[1] · Latif PourKarimi[2] · Felix Droop[1] · Ninon De Mecquenem[3] · Ulf Leser[3] · Knut Reinert[1]

✉ Somayeh Mohammadi
somayeh.mohammadi@fu-berlin.de

Latif PourKarimi
l.pourkarimi@razi.ac.ir

Felix Droop
felix.droop@fu-berlin.de

Ninon De Mecquenem
mecquenn@informatik.hu-berlin.de

Ulf Leser
leser@informatik.hu-berlin.de

Knut Reinert
Knut.Reinert@fu-berlin.de

1 Department of Mathematics and Computer Science, Freie Universität, Berlin, Germany

2 Department of Mathematics and Computer Science, Razi University, Kermanshah, Iran

3 Department of Mathematics and Computer Science, Humboldt-Universität zu Berlin, Berlin, Germany