Deadline-Constrained Many-Objective Workflow Scheduling for IoT Environments

Najwa Kouka, *Member, IEEE*, Sabrina De Capitani di Vimercati, *Fellow, IEEE*, Sara Foresti, *Senior Member, IEEE*, Vincenzo Piuri, *Life Fellow, IEEE*, and Pierangela Samarati, *Fellow, IEEE*

***Abstract*—Advanced data-intensive services and applications in the Internet of Things (IoT) scenarios require efficient access to external resources for data processing and computation. Fog computing addresses such a need by moving part of the activity closer to the edge, alleviating network congestion, and improving responsiveness. A major problem in scheduling the workflow required by the IoT applications for their execution in fog nodes is accounting for the many quality of service (QoS) requirements that should be guaranteed, as well as possible deadlines for workflow completion imposed by the applications. In this article, we present a novel meta-heuristic approach for solving a deadline-constrained many-objective workflow scheduling problem that enhances the arithmetic optimization algorithm (AOA) with dynamic evolutionary state estimation for selecting arithmetic operators and balancing exploration and exploitation in the search space. Our approach also accounts for parallelization in task allocation and includes a repairing mechanism for infeasible solutions. The extensive experimental evaluation on benchmarks of real-world workflows, comparing against alternative algorithms, demonstrates the effectiveness of our approach.**

***Index Terms*—Constrained many-objective optimization, Internet of Things (IoT) applications, workflow scheduling.**

I. Introduction

ADVANCEMENTS in pervasive and ubiquitous technologies have contributed to the development of a hyperconnected society where billions of connected devices are daily used for accessing data-intensive services and applications. Providing data and services conveniently available to users anytime and anywhere requires solutions to effectively use external storage and processing services from peripheral devices. In fact, the Internet of Things (IoT) applications may need to access remote data and external services. Also, despite the advancements in storage and processing power of peripheral devices, modern applications, increasingly requiring artificial intelligence (AI) and real-time processing, often demand more resources than individual devices can effectively provide. Computational offloading and scheduling have been emerging as a solution, allowing devices to rely on external resources for complex tasks, and thus reducing processing time and operational costs (Fig. 1). While cloud computing can handle data-intensive activities by providing virtually unlimited storage and computing power, the communication bandwidth between edge and cloud nodes introduces latency. By placing computation and data closer to the edge (i.e., peripheral devices), fog computing enables alleviating network congestion and improving responsiveness.

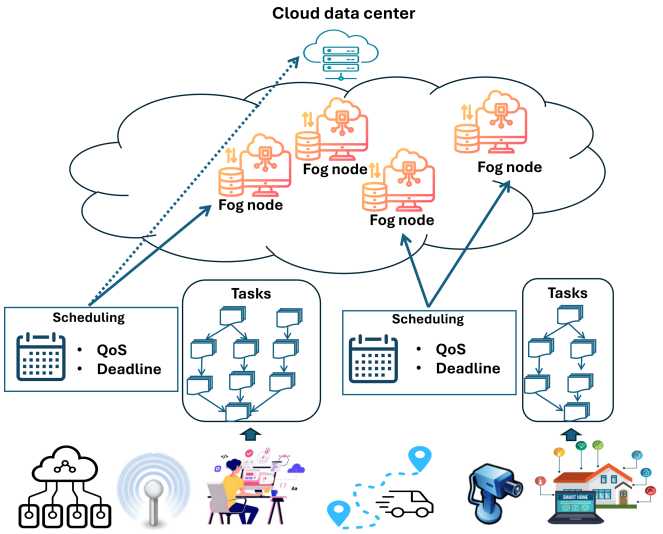

Fig. 1. Workflow scheduling in the IoT environments.

To make efficient use of fog/cloud resources in the IoT environments, it is crucial to provide solutions enabling effective dispatching of work (workflow scheduling) to nodes in the fog/cloud as better suited for applications. In fact, poor resource management can compromise the efficiency and effectiveness of applications and services, and can have a negative impact on the users' experience. Effectively responding to the problem of scheduling workflows in the IoT scenarios requires not only to provide effective use of external resources, but also to meet applications demand for quality of service (QoS), and for the satisfaction of deadline constraint for time-critical applications. Due to the complexity of the challenge and the difficulty in efficiently searching the space of solutions as the number of objectives increases, workflow scheduling is

Received 12 November 2025; revised 17 December 2025; accepted 4 January 2026. Date of publication 12 January 2026; date of current version 9 March 2026. This work was supported in part by the EC under projects Chips JU EdgeAI (101097300) and GLACIATION (101070141), by the Italian MUR under PRIN project POLAR (2022LA8XBH), and by project SERICS (PE00000014) under the MUR NRRP funded by the EU-NGEU. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Italian MUR. Neither the European Union nor the Italian MUR can be held responsible for them. *(Corresponding author: Vincenzo Piuri.)*

The authors are with the Università degli Studi di Milano, 20133 Milan, Italy (e-mail: vincenzo.piuri@unimi.it).

Digital Object Identifier 10.1109/JIOT.2026.3652002

frequently treated as an optimization problem. The richness of different objectives and requirements to be considered makes the problem even more complex. In fact, since different objectives often conflict with each other, the optimization of one objective can deteriorate the others.

In this article, we address the problem of workflow scheduling with QoS in the IoT environments. We formulate it as a deadline-constrained many-objective problem (to comprehensively capture different objectives) and propose a meta-heuristic for its solution. The proposed meta-heuristic significantly extends arithmetic optimization [1] with a dynamic evolutionary state estimation approach to balance exploration and exploitation in the search space of solutions. Also, our approach accounts for parallelization in task assignments to dynamically restrict the search space in the evolution process and reduce latency in workflow execution. The approach also includes a mechanism for repairing infeasible solutions, reallocating tasks that contribute to the deadline violation. The contribution of this article is multifold. First, we provide a simple yet comprehensive model capturing a rich set of QoS requirements and deadline constraint. Second, we formulate the problem of optimizing deadline-constrained workflow scheduling as a *constrained many-objective optimization problem* (CMaOP). Third, we provide a novel population-based meta-heuristic, called constrained many-objective arithmetic optimization algorithm II (CMaOAOAII), for adaptively solving the CMaOP problem. Fourth, we conduct an extensive experimental evaluation comparing our meta-heuristic with state-of-the-art algorithms. For the experimental evaluation, we use benchmark workflows of real-world applications, considering different numbers of tasks and varying (ranging from stricter to looser) deadline constraints. The results demonstrate the effectiveness of our approach in optimizing the constrained many-objective workflow scheduling problem.

The remainder of this article is organized as follows. Section II discusses related work. Section III introduces the workflow scheduling model. Section IV defines the objectives and the deadline constraint, and formulates the workflow scheduling problem as a CMaOP. Section V describes our CMaOAOAII meta-heuristic for solving the problem. Section VI presents and analyzes the experimental results. Section VII discusses challenges to be addressed for further enhancing the problem modeling and CMaOAOAII for its solution, representing opportunities for future work. Finally, Section VIII concludes this article and outlines future directions.

II. Related Work

Scheduling plays a crucial role in the IoT environments by ensuring the optimal utilization of resources. This is also testified by the several approaches that have been specifically developed for addressing this problem in the IoT environments. In particular, the line of work more closely related to our proposal is aimed at addressing the scheduling problem through meta-heuristic algorithms. While solutions for task scheduling (see [2], [3], [4]), similar to CMaOAOAII, aim at allocating tasks to available resources, they differ from our approach as they do not take dependencies among tasks into consideration for the allocation. The workflow scheduling problem, which takes task dependencies into account for allocation, has often been modeled as a multiobjective problem. In contrast to our approach, these solutions, therefore, support the optimization of only up to three objective functions (e.g., NSGA-II [5], ECRWSM [6], and DC-MOEA-DPDS [7]). Only a few meta-heuristic algorithms have extended the optimization problem to the consideration of more objective functions similar to CMaOAOAII. However, differing from CMaOAOAII, some of these meta-heuristics consider the workflow scheduling problem as unconstrained (see [8]). A few proposals, instead, consider scenarios, like the one considered in this article, which require the modeling and consideration of constraints (e.g., the deadline [9], [10] or security [11] constraints). CMaOAOAII, therefore, represents a meta-heuristic for the constrained many-objective workflow scheduling optimization problem. General-purpose constrained meta-heuristics for many-objective optimization problems typically handle high-dimensional objective spaces with constraints by incorporating feasibility into the dominance comparison (e.g., the constrained nondominated sorting genetic algorithm III (CNSGAIII) [12] and the dynamic CNSGAIII (DCNSGAIII) [13]). CMaOAOAII, differently from CNSGAIII and DCNSGAIII, leverages the peculiarities of the constrained workflow scheduling optimization problem, including the existence of dependencies among tasks, correlated QoS objectives, and an adaptive repair mechanism.

Another related line of work aims at addressing the scheduling problem in the IoT environments, leveraging modern AI techniques (e.g., deep neural networks, reinforcement learning, and statistical methods) for enhancing decision-making efficiency (see [14], [15], [16], [17], [18]). These approaches typically solve multiobjective optimization problems, providing interesting results. For instance, the solution in [17] adopts deep reinforcement learning for the IoT-based vehicle routing, optimizing task distribution among servers, improving time and energy consumption. CMaOAOAII differs from these approaches as it adopts an evolutionary-based technique for solving the workflow scheduling problem, while the others rely on different AI techniques.

Kouka et al. [19] introduced the idea of dynamic evolutionary state estimation for guiding the choice of the arithmetic operator based on convergence and diversity of solutions. The work includes only a preliminary experimental assessment, does not consider feasibility in evolutionary state estimation, does not integrate a strategy for available node selection to reduce the search space, and adopts a naive repairing technique. CMaOAOAII significantly advances on the idea, providing a model for the problem, and an advanced solution for dynamic state estimation and available node selection, as well as a novel hybrid repairing approach.

III. Workflow Scheduling and Background

A workflow consists of a set of tasks that process data to achieve a specific goal. Tasks have dependencies that determine the sequence in which they must be executed.

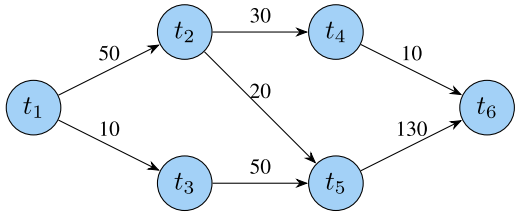

Fig. 2. Example of workflow.

A workflow can be represented as a *workflow graph* (workflow for short), formally defined as follows.

Definition 1 (Workflow): A *workflow* W is represented as a weighted directed acyclic graph $W = (T, E, w)$, where $T = \{t_1, \ldots, t_m\}$ is a set of vertices corresponding to the set of tasks, $E \subseteq T \times T$ is a set of edges corresponding to task dependencies, and $w{:}E \rightarrow \mathbb{R}$ is a weighted function representing the amount of data transferred between tasks.

Given a workflow $W = (T, E, w)$, edge $(t_i, t_j) \in E$ implies that task t_j (*successor* of t_i) can start only when task t_i (*predecessor* of t_j) has completed and t_j has received input from t_i. A task can have multiple immediate predecessors and successors (i.e., multiple incoming and outgoing edges). In this case, a task can start only when the task has received input from all its predecessors. Tasks without predecessors are called *initial tasks* and tasks without successors are called *terminal tasks*. Fig. 2 illustrates an example of workflow, with initial task t_1, with immediate successors t_2 and t_3, and terminal task t_6, with immediate predecessors t_4 and t_5. For simplicity, but without loss of generality, we assume that a workflow is characterized by one initial task, denoted as t_{init}, and one terminal task, denoted as t_{ter}. Workflows with multiple initial (or terminal) tasks are accommodated by adding a dummy initial (or terminal) task, predecessor of all the initial (or successor of all the terminal) tasks. The edges from/to the dummy initial/terminal task have weight equal to zero.

Given a set N of *nodes*, which can be fog/cloud virtual or physical machines, scheduling a workflow $W = (T, E, w)$ implies allocating tasks in T to nodes in N.

The allocation of tasks to nodes is formally defined as follows.

Definition 2 (Allocation): Given a workflow $W = (T, E, w)$ and a set N of nodes, an *allocation* function for T over N is a function $\alpha{:}T \rightarrow N$ that associates with each task $t \in T$ a node $n \in N$.

We refer to the workflow together with the allocation of its tasks to nodes as a workflow schedule, formally defined as follows.

Definition 3 (Workflow Schedule): A *workflow schedule* is a 5-uple of the form $S = (T, E, w, N, \alpha)$, where $W = (T, E, w)$ is a workflow, N is a set of nodes available for its execution, and $\alpha{:}T \rightarrow N$ is a function allocating tasks in T to nodes in N for their execution.

Allocation of tasks to nodes has a clear effect on completion time, cost, and energy consumption of the workflow execution, which depend on the specific characteristics of the nodes to which tasks are allocated. In Section IV, we first introduce different objectives, that is, completion time (called makespan), load balance, economic cost, and energy consumption, and then introduce the deadline constraint together with a formulation of the workflow scheduling problem.

IV. Constrained Workflow Scheduling Problem

The completion time and cost of a workflow depend on the execution time and cost of the individual tasks and of the communication between them, which depend on the characteristics of the nodes to which the tasks are allocated. Therefore, we first define the *execution time* of a task on a node and the *transmission times* for transferring data from a task to its successors in the workflow (Section IV-A). We then define the performance and cost objectives (Section IV-B) as well as the deadline constraint (Section IV-C). Finally, we formally define the workflow scheduling problem as a many-objective constrained problem (Section IV-D).

A. Execution and Transmission Time

The execution time for a task t on a node n is expressed as the ratio between the *length of the task,* expressed as the number of instructions to be executed, denoted as $\texttt{length}(t)$, and the *processing power of the node,* expressed as instructions per second, denoted as $\texttt{pproc}(n)$. Analogously, the transmission time for a data transfer from a node n is expressed as the ratio between the amount of data to be transferred and the bandwidth of the node, denoted as $\texttt{bw}(n)$.

With allocation function α determining the nodes to which tasks are allocated, we then consider execution and transmission times entailed by an allocation as follows.

Given a workflow schedule $S = (T, E, w, N, \alpha)$, the execution time of a task $t \in T$ on node $\alpha(t)$, denoted as $\text{ET}^{\alpha}(t)$, is defined as the ratio between the length of the task and the processing power of the node to which the task is allocated

$$\text{ET}^{\alpha}(t) = \frac{\texttt{length}(t)}{\texttt{pproc}(\alpha(t))} \tag{1}$$

The transmission time between a task t and task t_s, with $(t, t_s) \in E$, denoted as $\text{TT}^{\alpha}(t, t_s)$, is the ratio between the amount of data that has to be transferred from t to t_s and the network bandwidth of the node to which t is allocated, if t_s is allocated to a different node; it is 0, otherwise

$$\text{TT}^{\alpha}(t, t_s) = \begin{cases} \frac{w(t,t_s)}{\texttt{bw}(\alpha(t))}, & \text{if } \alpha(t) \neq \alpha(t_s) \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Both the execution time and the transmission time are used in the definition of the performance and cost objectives, which we illustrate next.

B. Objectives

Objectives comprise both performance aspects and cost considerations. The performance of a workflow schedule is captured in terms of *makespan* (i.e., completion time) and *load balancing* (i.e., distribution of the work among different nodes). As for costs, in line with the ever increasing importance of green computing and the attention on carbon emission, we consider both *economic cost* (cost, for short) as well as *energy consumption* demanded by the computation and data transmission. We now illustrate each objective in more detail.

1) Makespan: An important criterion for the performance of the workflow execution is the *makespan*, defined as the total time required to complete the workflow. Since a task t can start only when it has received input from all its immediate predecessors, its completion time needs to consider, besides the time to execute the task itself, the time required for its predecessors to complete and transmit input to t. The makespan for a workflow schedule is then defined as follows.

Definition 4 (Makespan): Given a workflow schedule $S = (T, E, w, N, \alpha)$, the *makespan* of S, denoted as Makespan(S), is defined as

$$\text{Makespan}(S) = \max\{\text{FT}^{\alpha}(t) : t \in T\}$$

where $\text{FT}^{\alpha}(t) = \text{ST}^{\alpha}(t) + \text{ET}^{\alpha}(t)$ is the finish time of t on node $\alpha(t)$, with $\text{ST}^{\alpha}(t)$ the start time of task t and $\text{ET}^{\alpha}(t)$ the execution time of t on $\alpha(t)$ [see (1)]. The start time $\text{ST}^{\alpha}(t)$ is computed as the maximum between the ready time of node $\alpha(t)$ and $\text{FT}^{\alpha}(t_p) + \text{TT}^{\alpha}(t_p, t)$, for all $(t_p, t) \in E$, with $\text{TT}^{\alpha}(t_p, t)$ the transmission time from t_p on $\alpha(t_p)$ to t on $\alpha(t)$ [see (2)]. Note that the start time of initial task t_{init} is zero (i.e., $\text{ST}^{\alpha}(t_{init}) = 0$).

2) Load Balance: Another important aspect for the performance of a workflow is the *load balance*, meaning a balanced distribution of work among the nodes. We consider a set R of resources, offered by nodes and consumed by tasks, which comprise, for example, CPU, memory, and storage. We denote with $r_i(t)$ ($r_i(n)$, respectively) the amount of resource $r_i \in R$ required for executing task t (offered by node n, respectively). The load on a node entailed by a given allocation is, then, the amount of resources required for the execution of all the tasks allocated to the node. The load formulation can also weight various resource types differently. Representing as ϕ_i the weighting factor for resource r_i (with $\sum_{r_i \in R} \phi_i = 1$), the load on a node n entailed by an allocation α, denoted $L(n)$, is then formally defined as follows:

$$L(n) = \sum_{\substack{t \in T : \alpha(t) = n \\ r_i \in R}} \phi_i \cdot \frac{r_i(t)}{r_i(n)}. \tag{3}$$

The load balance of a schedule can then be achieved by minimizing the standard deviation of the load across the nodes, formally stated in the following definition.

Definition 5 (Load Deviation): Given a workflow schedule $S = (T, E, w, N, \alpha)$ and a set R of resources, the *load deviation* of S is defined in terms of standard deviation as

$$\text{Lsd}(S) = \sqrt{\frac{1}{|N|} \cdot \sum\nolimits_{n \in N} (L(n) - \text{Lavg}(S))^2}$$

where $L(n)$ is the load of node n [see (3)], and $\text{Lavg}(S) = \sum_{n \in N} L(n)/|N|$ is the average load of the nodes in N.

3) Economic Cost: The economic cost of a workflow schedule is defined as the sum, over all tasks, of the cost to execute a task and to communicate results to its immediate successors, if allocated to a different node. The cost of executing a task and of transmitting its results depends on the node to which the task is allocated. The economic cost of a workflow is formally defined as follows.

Definition 6 (Workflow Cost): Given a workflow schedule $S = (T, E, w, N, \alpha)$, the *workflow cost*, denoted Cost(S), is defined as

$$\text{Cost}(S) = \sum_{t \in T} \text{Ccomp}(\alpha(t)) \cdot \text{ET}^{\alpha}(t) + \sum_{(t, t_s) \in E} \text{Ccomm}(\alpha(t)) \cdot \text{TT}^{\alpha}(t, t_s)$$

with $\text{Ccomp}(\alpha(t))$ and $\text{Ccomm}(\alpha(t))$ the computation cost per time unit and the communication cost per time unit of node $\alpha(t)$, respectively.

4) Energy Consumption: The energy consumption of a workflow schedule is defined as the sum, over all tasks, of the energy consumption to execute a task and to communicate results to its immediate successors, if allocated to a different node. The energy consumption of executing a task and of transmitting its results depends on the node to which the task is allocated. The energy consumption of a workflow is formally defined as follows.

Definition 7 (Workflow Energy Consumption): Given a workflow schedule $S = (T, E, w, N, \alpha)$, the *workflow energy consumption*, denoted as Energy(S), is defined as

$$\text{Energy}(S) = \sum_{t \in T} \text{Ecomp}(\alpha(t)) \cdot \text{ET}^{\alpha}(t) + \sum_{(t, t_s) \in E} \text{Ecomm}(\alpha(t)) \cdot \text{TT}^{\alpha}(t, t_s)$$

with $\text{Ecomp}(\alpha(t))$ and $\text{Ecomm}(\alpha(t))$ the computation consumption per time unit and the communication consumption per time unit of node $\alpha(t)$, respectively.

C. Deadline Constraint

While the time for completing a workflow, that is, the makespan, is one of the factors considered in computing an optimal allocation, it is considered together with other objectives (i.e., load balancing, cost, and energy). However, a workflow may be time-critical and has a deadline for its completion. A deadline imposes an upper bound on the completion time of the workflow. Hence, notwithstanding the overall consideration of all the objectives (the many factors to be optimized), the task allocation must guarantee that the workflow execution is completed within the given deadline. A workflow schedule satisfies a deadline constraint if its makespan remains within the deadline, as formally stated in the following definition.

Definition 8 (Workflow Deadline Satisfaction): Given a workflow schedule $S = (T, E, w, N, \alpha)$, we say that S *satisfies* a deadline `deadline` iff $\text{Makespan}(S) \leq$ `deadline`.

D. Workflow Scheduling Problem

The workflow scheduling problem consists of identifying, for a given workflow W and a set N of nodes, an allocation function α that minimizes the makespan, load deviation, economic cost, and energy consumption of the workflow. In addition, the allocation must satisfy the deadline constraint. The problem is formally defined as follows.

Problem 1 (Workflow Scheduling): Let $W = (T, E, w)$ be a workflow, and `deadline` be a deadline constraint for W. Given a set N of nodes, compute an allocation of tasks to nodes $\alpha: T \rightarrow N$ such that

$$\min_{\alpha} \begin{cases} \text{Makespan}(S) & \text{(Definition 4)} \\ \text{Lsd}(S) & \text{(Definition 5)} \\ \text{Cost}(S) & \text{(Definition 6)} \\ \text{Energy}(S) & \text{(Definition 7)} \end{cases}$$

$$\text{s.t. Makespan}(S) \leq \texttt{deadline} \quad \text{(Definition 8)}$$

with $S = (T, E, w, N, \alpha)$ the workflow schedule.

Problem 1 is a CMaOP, and in Section V, we will illustrate our evolutionary approach to solve it. In the description of the approach, we will use $\mathcal{F}$ to denote the set of all objective functions (i.e., $\mathcal{F}$ = {Makespan, Lsd, Cost, Energy}).

V. CMAOAOAII

Our approach for solving the workflow scheduling problem (Problem 1), called CMaOAOAII, is a population-based many-objective optimization method that leverages the arithmetic optimization algorithm (AOA) [1]. CMaOAOAII starts from a randomly generated population of candidate solutions (allocations). It, then, performs an evolutionary state estimation process (Fig. 3), resulting in solutions approximating the Pareto front of the many-objective optimization problem. This optimization process consists of the application of arithmetic operators to the solutions in the population for examining the search space through *exploration* and *exploitation*. Different from AOA, which selects the arithmetic operator for each solution at random at each iteration, our approach selects the arithmetic operator based on the *convergence* and *diversity* of the solutions.

In this section, we first illustrate solution encoding, population initialization, and solution archiving (Section V-A). Then, we discuss our evolutionary state estimation approach and how exploration and exploitation are applied to evolve the population (Section V-B). Finally, we illustrate the dynamic evolutionary state estimation, which also includes the repair of solutions that violate the deadline constraint (Section V-C).

A. Solution Encoding/Archival and Population Initialization

A solution to the workflow scheduling problem is encoded as a pair $\langle Y, A \rangle$ of matrices. Both Y and A have a row for each task $t_i \in T$ and a column for each node $n_j \in N$. Y represents a position of a candidate solution and is a matrix of real values on which the evolutionary process operates. More specifically, entries in Y are real values in the range $[L, U]$, with L and U the specified lower and upper bounds, respectively. A is a binary matrix, derived from Y, which represents the allocation of tasks to nodes (i.e., $A_{i,j} = 1$ implies $\alpha(t_i) = n_j$). In the following, we refer to an allocation interchangeably by using the allocation matrix A or the corresponding allocation function α.

The binarization of Y into A operates considering each row Y_i, which corresponds to task t_i, together with the set of nodes N_i available for its allocation (we will elaborate on N_i in the illustration of the process). Node $n_j \in N_i$ for which $A_{i,j}$ is to be set to 1 is determined as a j such that $Y_{i,j}$ is the largest value for which sigmoid($Y_{i,j}$) is greater than a randomly extracted value (different for each j) between 0 and 1. If no j such that $n_j \in N_i$ satisfies such condition, a j for which $Y_{i,j}$ has the largest value is chosen. The pseudocode of the *Binary* function is reported with other procedures and functions in Fig. 4. In the pseudocode, variable amax records the largest $Y_{i,j}$ encountered whose sigmoid satisfied the condition against the random value; variable max records the largest $Y_{i,j}$ encountered (up to where amax first changed its value), while jmax records the last j for which amax (or max if amax never changed) changed its value.

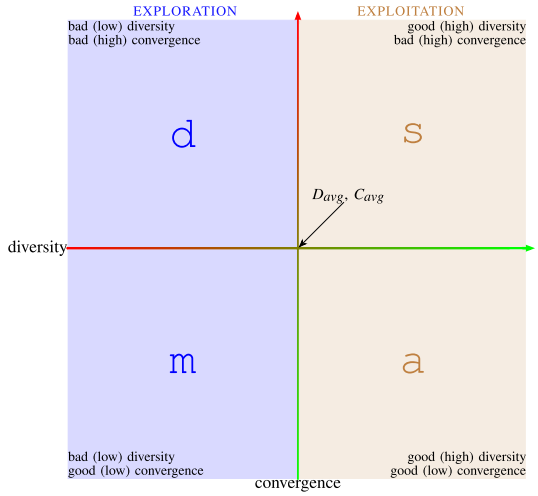

Fig. 3. Evolutionary state estimation.

CMaOAOAII maintains a population $\mathbb{P}$ of solutions subject to the evolutionary process as well as an archive $\mathbb{S}$ of best solutions found in the evolution. Set $\mathbb{P}$ is initialized with k solutions, whose Y is obtained by randomly extracting values between $[L, U]$ for its cells, and A is its binarization considering all nodes in N. Archive $\mathbb{S}$ is updated as population $\mathbb{P}$ evolves. For efficiency reason, and to avoid redundancy, a threshold (maxSol) is set on the number of solutions in the archive $\mathbb{S}$, and only solutions that are not dominated by other solutions in the archive are maintained. Among feasible solutions, dominance is defined according to Pareto; it is based on the makespan, otherwise. More precisely, a solution $\langle Y, A \rangle$ *dominates* another solution $\langle Y', A' \rangle$, denoted as $\langle Y, A \rangle \succ \langle Y', A' \rangle$, if either $\langle Y', A' \rangle$ is feasible and Pareto dominance on the objective functions holds or $\langle Y', A' \rangle$ is infeasible and has a makespan not smaller than $\langle Y, A \rangle$. (Note that this applies regardless of whether $\langle Y, A \rangle$ is, or is not, unfeasible.) Formally, $\langle Y, A \rangle \succ \langle Y', A' \rangle$, iff either: 1) $\langle Y', A' \rangle$ is feasible: $\forall f \in \mathcal{F}$, $f(A) \leq f(A')$, and $\exists f \in \mathcal{F}$ s.t. $f(A) < f(A')$ and 2) $\langle Y', A' \rangle$ is infeasible: *Makespan*$(A) <$ *Makespan*(A').

Here and in the following, with a slight abuse of notation, $f(A)$, with $f \in \mathcal{F}$ an objective function to be minimized (see Problem 1), is used as a shorthand for $f(S)$.

B. Population Evolution

The optimization process of CMaOAOAII operates according to an evolutionary strategy, iteratively applying arithmetic

CMaOAOAII

MAIN

$\mathbb{P} := \emptyset$; $\mathbb{S} := \emptyset$ /* population and archive of solutions */
for $l := 1, \ldots, k$ **do** /* population initialization */
  $A := [0]_{|T|\times|N|}$ /* initialize the assignment matrix to 0 */
  for $i := 1, \ldots, |T|$ **do**
    for $j := 1, \ldots, |N|$ **do**
      $Y_{i,j} :=$ *random*(L, U) /* random generation of a solution */
    $A_i :=$ **Binary**(Y_i, N) /* identify assignment for t_i */
  $\mathbb{P} := \mathbb{P} \cup \{\langle Y, A\rangle\}$ /* insert solution in the population */
  Update_Archive(Y,A)
for $g := 1, \ldots,$ *maxIter* **do** /* optimization process */
  $D_{avg} :=$ *avg*$\{D(Y,A) : \langle Y, A\rangle \in \mathbb{P}\}$ /* compute avg diversity */
  $C_{avg} :=$ *avg*$\{C(Y,A) : \langle Y, A\rangle \in \mathbb{P}\}$ /* compute avg converge */
  for each $\langle Y, A\rangle \in \mathbb{P}$ **do**
    $\langle Y', A'\rangle :=$ **Update_Position**(Y, A, C_{avg}, D_{avg})
    if *Makespan*(A') > `deadline` **then**
      $\langle Y', A'\rangle :=$ **Repair**(Y', A')
    if $\langle Y', A'\rangle \succ \langle Y, A\rangle$ **then**
      $\langle Y, A\rangle := \langle Y', A'\rangle$ /* keep the new solution */
      Update_Archive(Y,A)

REPAIR(Y,A)

if (*Makespan*(A) − `deadline`) $\leq$ *threshold* · `deadline` **then**
  Let $\langle Y^1, A^1\rangle, \langle Y^2, A^2\rangle, \langle Y^3, A^3\rangle$ be three randomly extracted solutions ordered wrt their makespan
  $Y^m := Y^1 + SF \times (Y^2 - Y^3)$ /* perform mutation */
  $A^r := [0]_{|T|\times|N|}$ /* reset assignment matrix */
  for $i := 1, \ldots, |T|$ **do**
    for $j := 1, \ldots, |N|$ **do**
      if CR > *random*(0,1) OR $i =$ *random*(1,$|T|$) **then**
        $Y^r_{i,j} := Y^m_{i,j}$
      else $Y^r_{i,j} := Y_{i,j}$
    $A^r_i :=$ **Binary**(Y^r_i, N) /* identify assignment */
else
  Let α be the allocation function corresponding to A
  $Q := \{t_{ter}\}$
  while $Q \neq \emptyset$ **do**
    Let t_i be the task with minimum TTC in Q
    $Q := Q \setminus \{t_i\}$
    efinish := $FT^{\alpha}(t_i)$ /* earliest finish time of task t_i */
    node := $\alpha(t_i)$ /* best node for the reallocation of t_i */
    for each $n \in N$ **do**
      Let α' be such that $\alpha'(t_p)$=$\alpha(t_p)$, $\alpha'(t_i)$=n
      ready := *max*$\{FT^{\alpha}(t_j) : j < i$ AND $\alpha(t_j)$=$n\}$
      estart := *max*$\{FT^{\alpha}(t_p) + TT^{\alpha'}(t_p, t_i) : (t_p,t_i)\in E\}$
      finish := *max*$\{$*ready*, *estart*$\}$ + $ET^{\alpha'}(t_i)$
      if *finish* < *efinish* **then**
        efinish := *finish*
        node := n
    $\alpha(t_i)$:= *node*
    for each $t_p \in T$ s.t. $(t_p,t_i)\in E$ **do**
      if $FT^{\alpha}(t_p) + TT^{\alpha}(t_p, t_i) + TTC^{\alpha}(t_i)$ > `deadline` **then**
        $Q := Q \cup \{t_p\}$
  for $i := 1, \ldots, |T|$ **do** /* construct $\langle Y^r, A^r\rangle$ from α */
    for $j := 1, \ldots, |N|$ **do**
      if $\alpha(t_i) = n_j$ **then**
        $A^r_{i,j} := 1$; $Y^r_{i,j} := U$
      else $A^r_{i,j} := 0$; $Y^r_{i,j} := L$
return(Y^r, A^r)

BINARY(Y_i, N_i)

max := *amax* := $L - 1$
for each $n_j \in N_i$ **do** /* consider nodes in random order */
  if *sigmoid*$(Y_{i,j})$ > *random*(0,1) AND $Y_{i,j} >$ *amax* **then**
    amax := $Y_{i,j}$; *jmax* := j
  if *amax* = $L - 1$ AND $Y_{i,j} >$ *max* **then**
    max := $Y_{i,j}$; *jmax* := j
$A_i := [0]_{|N|}$ /* reset assignment for the task */
$A_{i,jmax} := 1$ /* t_i is allocated to n_j */
return(A_i)

UPDATE_ARCHIVE(Y, A)

if $\nexists \langle Y', A'\rangle \in \mathbb{S} : \langle Y', A'\rangle \succ \langle Y, A\rangle$ **then**
  $\mathbb{S} := \mathbb{S} \cup \{\langle Y, A\rangle\}$ /* insert not dominated */
  for each $\langle Y', A'\rangle \in \mathbb{S}$ **do**
    if $\langle Y, A\rangle \succ \langle Y', A'\rangle$ **then**
      $\mathbb{S} := \mathbb{S} \setminus \{\langle Y', A'\rangle\}$ /* delete dominated solutions */
  if $|\mathbb{S}| >$ *maxSol* **then**
    Let $\langle Y', A'\rangle \in \mathbb{S}$ with minimum crowding distance
    $\mathbb{S} := \mathbb{S} \setminus \{\langle Y', A'\rangle\}$

UPDATE_POSITION(Y, A, C_{avg}, D_{avg})

if *Makespan*(A) > `deadline` **then** /* infeasible solution */
  if $D(Y,A) < D_{avg}$ AND $C(Y,A) > C_{avg}$ **then**
    OP := `d` /* bad (low) diversity and bad (high) convergence */
  else OP := `m` /* good (high) diversity or good (low) convergence */
else /* feasible solution */
  if $D(Y,A) < D_{avg}$ **then** /* bad (low) diversity, exploration */
    if $C(Y,A) > C_{avg}$ **then**
      OP := `d` /* bad (high) convergence */
    else OP := `m` /* good (low) convergence */
  else /* good (high) diversity, exploitation */
    if $C(Y,A) > C_{avg}$ **then**
      OP := `s` /* bad (high) convergence */
    else OP := `a` /* good (low) convergence */
$A' := [0]_{|T|\times|N|}$ /* initialize the assignment matrix to 0 */
for $i := 1, \ldots, |T|$ **do**
  $N_i :=$ **Compute_Available_Nodes**(t_i, A)
  for each $n_j \in N_i$ **do**
    $Y'_{i,j}$:= apply OP to $Y_{i,j}$ according to Eq. 4
  $A'_i :=$ **Binary**(Y'_i, N_i) /* identify assignment */
return(Y', A')

COMPUTE_AVAILABLE_NODES(t_i, A)

Available := N /* initialize the set of available nodes for t */
Let α be the allocation function corresponding to A
for each $t_j \in T$ s.t. (*level*(t_j) = *level*$(t_i)-1$) OR (*level*(t_j) = *level*(t_i) AND $j \leq i$) **do**
  Let α' be such that $\alpha'(t_p)$=$\alpha(t_p)$, $\alpha'(t_i)$=$\alpha(t_j)$
  if $FT^{\alpha}(t_j) > max\{FT^{\alpha}(t_p) + TT^{\alpha'}(t_p, t_i) : (t_p, t_i)\in E\}$ **then**
    Available := *Available* $\setminus$ $\alpha(t_j)$
if *Available* = $\emptyset$ **then**
  Available := N
return(*Available*)

Fig. 4. Pseudocode of CMaOAOAII.

operators on the population of solutions. Similar to AOA [1], the evolution of solutions leverages arithmetic operators (i.e., addition, subtraction, multiplication, or division). These operators enforce exploration and exploitation, with the goal of finding solutions closer to, and well-distributed along, the Pareto front. Different from AOA, in our approach, the choice of the arithmetic operator to be applied for the evolution of each solution is driven by diversity and convergence of solutions in the population (instead of being chosen randomly). The *diversity* of a solution is measured as its minimum Euclidean distance with respect to the other solutions in the normalized objective space (i.e., the space with a dimension for each objective function). Formally, $D(Y,A) = \min\{\|\theta(A) - \theta(A_i)\|_2 : A_i \in \mathbb{P}\}$, where $\theta(A_i) = 1/\sum_{f\in\mathcal{F}} f(A_i) \times F(A_i)$, $F(A_i)$ is the vector of normalized objective functions, and $\|.\|_2$ is the Euclidean distance (2-norm). The *convergence* of a

solution is measured as its Euclidean distance, in the objective space, from the ideal solution Z^* (i.e., the solution having, for each objective function, the best value reached by solutions in $\mathbb{S}$). Formally, $C(Y,A) = \|F(A) - F(Z^*)\|_2$, with F the vector of normalized objective functions, where normalization is computed using the ideal point Z^* and the nadir point (i.e., the solution having, for each objective function, the worst value reached by any solution in $\mathbb{S}$).

We first define the application of operators to evolve solutions and then illustrate our evolutionary state estimation guided by diversity and convergence.

1) Arithmetic Operators: Solutions evolve through the application of arithmetic operators. The application of operator $\text{OP} \in \{a, s, m, d\}$ to solution $\langle Y, A\rangle$ consists in updating the values in Y as follows:

$$Y_{i,j} = \begin{cases} (\phi - 1) \times Y_{i,j} \times \cos(2\pi l) & \\ +\phi \times \text{Best}_{i,j} \, \text{OP}(\text{MOP} + \epsilon) \times \ \text{step}, & \text{if OP} \ \in \{m, d\} \\ Y_{i,j} \ \text{OP} \ (\text{MOP} \times \ \text{step} \ ), & \text{if OP} \ \in \{a, s\} \end{cases} \tag{4}$$

where $\phi = 1 - 1/(1 + e^{5(2g/\text{maxIter}} - l))$ is a nonlinear factor regulating the dependence from *Best* (which is, considering two randomly extracted solutions in $\mathbb{S}$, the one having the higher crowding distance [20]); ϵ is a small positive constant necessary to prevent division by 0; $step = (U-L) \times \mu + L$, with μ a control parameter that influences the step size; and MOP is the math optimizer probability that controls the magnitude of the position update. Such probability is computed as $\text{MOP} = 1-(g/\text{maxIter})^{1/\kappa}$ with κ a sensitivity parameter that governs the accuracy of the exploitation process over iterations, l a random value, and g the current iteration of the evolutionary process.

2) Evolutionary State Estimation: The selection of the most suited arithmetic operator OP for evolving a solution $\langle Y, A\rangle$ is based on its diversity $D(Y,A)$ and convergence $C(Y,A)$, with respect to the average diversity D_{avg} and average convergence C_{avg} of the solutions in population $\mathbb{P}$. The goal is to evolve in a population that enjoys *high diversity* (i.e., it is well spread along the Pareto front) and *low convergence* (i.e., it is close to the Pareto front). Fig. 3 illustrates our evolutionary state estimation, where the x-axis represents the diversity, the y-axis represents the convergence, and the cross point between the two represents where their average values meet. As visible from the figure, diversity regulates whether *exploration* or *exploitation* should be applied, while convergence regulates the specific exploration/exploitation operator to be applied. More specifically, at each iteration of the evolution for each solution $\langle Y, A\rangle$, if the diversity of the solution is lower than the average diversity of the population, then exploration is applied; else exploitation is applied. The rationale is that a low value of diversity indicates that the solution is in a crowded area and exploration enables to move away toward a better distribution. By contrast, when diversity is high, the solution is in a noncrowded area, and exploitation enables to evolve in nearby positions to possibly further enhance the quality of the solution and move it closer to the Pareto front. When exploring, if convergence is bad (higher than average), division is applied; else multiplication is applied. The rationale is to enforce a greater jump to move closer to the Pareto front in the first case, while enforcing a smaller jump to move away from the crowded area but remaining close to the Pareto front in the second case. Similarly, when exploiting, if convergence is bad (high), subtraction is applied; else addition is applied. Again, the rationale is to remain in the not well-explored area but remain close (or move toward) the Pareto front. A special case is the evolution of infeasible solutions for which exploitation is substituted with exploration through multiplication: when both diversity and convergence are bad, then division is applied; else, in all other three cases, multiplication is applied. The rationale for substituting multiplication to addition/subtraction applied in the exploitation is to perform a deviation on the solutions to explore feasible regions in the solution space.

C. Workflow Scheduling Approach

Our workflow scheduling approach uses evolutionary state estimation as described above to evolve solutions, which then translate corresponding to allocations of tasks in the workflow. Clearly, the order in which tasks are considered needs to reflect dependencies (intuitively walking forward in the workflow) and account for parallel paths in the workflow. Therefore, we define the order for tasks considering their *upward rank* (rank, for short) in the workflow. Given an allocation α, the upward rank of a task t captures the highest time, along any path, to complete the workflow from t. Hence, the rank of the terminal task is its execution time. The rank of any other task is its execution time plus the maximum, over all tasks t_s successors of t, of the sum of the rank of t_s and plus the time for transmitting t's results to t_s. Formally, $\text{rank}^{\alpha}(t) = \text{ET}^{\alpha}(t) + \max\{\text{rank}^{\alpha}(t_s) + \text{TT}^{\alpha}(t, t_s) : (t, t_s) \in E\}$.. Since at the start time allocation α is still to be defined, for establishing the initial order of tasks, we consider a *rank* assuming all tasks are allocated to a single node. To simplify notation, we assume such an order to be captured by the index associated with the task. Hence, we consider tasks in T (and consequently rows in matrices Y and A) to be ordered in nonincreasing order of ranks, that is, for each $t_i, t_j \in T$: $i < j$ reflects that $\text{rank}(t_i) \geq \text{rank}(t_j)$ holds.

We are now ready to illustrate the working of our CMaOAOAII approach.

1) Main Process: The pseudocode of CMaOAOAII is illustrated in Fig. 4. The algorithm (*Main*) starts by randomly generating k solutions for population $\mathbb{P}$ and saving them (lines 1–9) in the archive of solutions $\mathbb{S}$. As described in Section V-A, the archive of solutions (updated by procedure *Update_Archive*) maintains the best maxSol (with $\text{maxSol} \leq k$) solutions found and keeps only solutions that do not dominate each other. If the addition of a new solution exceeds the maxSol threshold, a solution in $\mathbb{S}$ having the lowest crowding distance (and thus providing poor diversity contribution) is removed (lines 6–8 in *Update_Archive*). After such initialization, maxIter iterations are performed evolving the population. Population evolution works as described in Section V-B, first determining the average diversity and convergence of the population (lines 11 and 12) and then calling procedure *Update_Position* to evolve each solution

$\langle Y, A\rangle$ in $\mathbb{P}$ into a new solution $\langle Y', A'\rangle$. If the solution $\langle Y', A'\rangle$ produced by the evolution violates the deadline, a *Repair* procedure (we elaborate on it as follows) is called on it (line 16). If $\langle Y', A'\rangle > \langle Y, A\rangle$ (i.e., $\langle Y', A'\rangle$ is better), then $\langle Y, A\rangle$ is set to $\langle Y', A'\rangle$ in $\mathbb{P}$ (lines 17 and 18). Finally, the procedure *Update_Archive* is called to add $\langle Y, A\rangle$ to the archive of solutions $\mathbb{S}$ (line 19). The process is repeated for all the maxIter iterations, resulting in maxSol best solutions in archive $\mathbb{S}$.

2) Update_Position: For each solution on which it is called, the function first determines (lines 1–13) the arithmetic operator OP to be applied for evolution as described in Section V-B. Then, after initializing the binary allocation matrix (line 14), for each task t_i (considering tasks in the defined order), it first determines (by calling procedure *Compute_Available_Nodes*) the set of nodes N_i that could be available to serve the task with no delay. Hence, for each node $n_j \in N_i$, it applies operator OP to cell $Y_{i,j}$ making it evolve in $Y'_{i,j}$ (lines 17 and 18). Finally, procedure *Binary* is called to retrieve the binarization of Y'_i considering available nodes N_i (line 19) and the resulting solution $\langle Y', A'\rangle$ is returned (line 20).

3) Compute_Available_Nodes: The function is called to determine the nodes that could be *Available* to serve a task t_i with no delay, with the aim of maximizing parallel execution and avoiding waiting time. It considers all tasks t_j preceding t_i in the initial rank and either at the same level (i.e., the same length of longest path from initial task t_{init}) as t_i or at the level immediately preceding the one of t_i (line 3). It, then, discards from the available nodes, any node to which one of such t_j has been allocated if the finish time of t_j is greater than the time at which t_i could be ready to start if assigned to the same node as t_j (hence, the α' at lines 4 and 5 differing from α by assuming t_i allocated at the same node as t_j). If the control discards all nodes (i.e., *Available* becomes empty), then all nodes are to be considered (lines 7 and 8). Finally, the set of *Available* nodes is returned (line 9).

4) Repair: Function repair is called for each solution, produced by the evolution, which violates the deadline (lines 15 and 16 of the main process) to attempt its repair. The function distinguishes the case where deadline violation is limited, that is, within a given *threshold* (e.g., 10% in our experiments), or is more severe. If the deadline violation is within the threshold, it attempts repairing by applying the differential evolution (DE) algorithm [21] (lines 1–10). More specifically, it extracts three random solutions from archive $\mathbb{S}$ (or from $\mathbb{P}$, if $\mathbb{S}$ has less that three solutions) and computes their mutation $Y^m = Y^1 + \text{SF} \times (Y^2 - Y^3)$, where SF is a scale factor and Y^1, Y^2, and Y^3 are the extracted solutions in decreasing order of makespan. The DE algorithm computes the repaired solution Y^r by applying the crossover operation and, for each task t_i and node n_j, setting $Y^r_{i,j}$ to $Y^m_{i,j}$, if CR > random(0,1) or i = random$(1, |T|)$, and to $Y_{i,j}$ otherwise, with CR $\in [0, 1]$ the crossover probability. If the deadline violation exceeds the threshold, repairing is performed by traversing the workflow backward and attempting reallocation of tasks that are on a *critical path* (i.e., a path that contributes to the violation of the deadline). Tasks to be possibly reallocated are included in a set Q, initialized with the terminal node (line 13). The repair process extracts from Q the task t_i with minimum time to complete (TTC, which is the difference between the finish time of the final task and the finish time of the considered task) and initializes its earliest finish time *efinish* and assigned node *node* as per the current allocation (lines 17 and 18). Then, for each node $n \in N$, it computes the following values. The time at which the node could be *ready* to serve the task (which is the maximum finish time of all tasks preceding t_i in the original ranking, line 21). The earliest time (*estart*, line 22) at which t_i could have all its input to start at n (which is the maximum sum, across all predecessor t_p of t_i of the finish time of t_p and the time for transmitting data resulting from t_p and input to t_i). The *finish* time (line 23) at which t_i would finish if allocated to n (which is the maximum between the two times just calculated, summed to the execution time of t_i at node n). If such finish time is smaller than the earliest finish time recorded (*efinish*), the earliest finish time and the allocated node are updated (lines 24–26). At the end of the cycle, t_i's allocation is changed to *node*, which is the node that guarantees the earliest finish time (line 27). Then, each predecessor t_p of t_i for which the sum of the finish time of t_p, the transmission time from t_p to t_i, and the TTC of t_i is greater than the deadline is added to the set Q of tasks to be reallocated (lines 28–30). The process repeats until set Q is empty. At the end, the repaired solution is constructed based on the new computed allocation α (lines 31–35) and returned (line 36).

VI. Experimental Evaluation

We conducted a series of experiments to assess the effectiveness of CMaOAOAII in solving the deadline-constrained many-objective workflow scheduling problem. In the following, we first illustrate the algorithms used for comparison, the considered workflows, and parameter setting (Section VI-A). We, then, report the results of our experiments, measuring the feasibility and success rate, and the quality of the computed solutions (Section VI-B).

A. Experimental Configurations

To analyze the effectiveness of CMaOAOAII, we compare it against existing solutions considering benchmark workflows. In the following, we describe the algorithms, the benchmark workflows, as well as the considered experimental settings.

1) Algorithms: We compared CMaOAOAII against four existing algorithms that have demonstrated competitive performance in constrained many-objective optimization.

1) *CMaOAOA* [19] introduced dynamic evolutionary state estimation for navigating the search space, but does not consider feasibility in the estimation, does not integrate a strategy for available node selection, and adopts a naive repairing technique.
2) *Many-objective particle swarm optimization (MaOPSO)* [9] effectively handles many-objective problems. It dynamically adjusts reference points throughout the search to maintain solution diversity and convergence, and it is employed for solving constrained workflow scheduling.

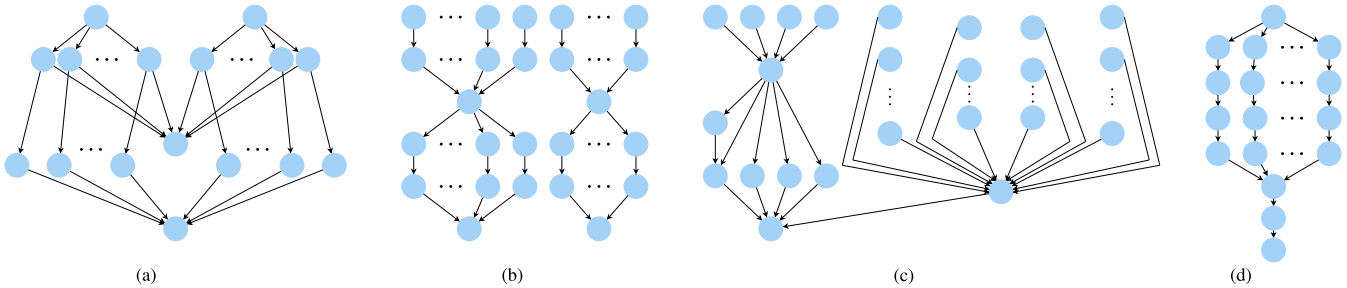

Fig. 5. Topologies of (a) CyberShake, (b) Inspiral, (c) SIPHT, and (d) Epigenomics workflows.

3) *CNSGAIII* [12] is widely used extension of NSGAIII, adapted for constrained problems. It employs reference-point-based environmental selection along with constraint dominance, making it a relevant benchmark for constrained multiobjective and many-objective optimization problems.
4) *DCNSGAIII* [13] dynamically handles constraints. It considers constraints as objective functions and uses ϵ-constrained dominance in its environmental selection process to promote feasible solution generation.

For the comparison, we implemented all algorithms using the jMetal framework [22].

2) Workflows: We performed experiments on the workflows simulated using the following four real-world benchmark workflow applications[1] [23].

1) *CyberShake* used by the Southern California Earthquake Center to evaluate seismic hazard in specific geographic areas.
2) *LIGO inspiral analysis* (Inspiral, for short) used by the Laser Interferometer Gravitational Wave Observatory (LIGO) to analyze data resulting from the coalescence of compact binary systems (e.g., binary neutron stars and black holes).
3) *SIPHT* developed by a bioinformatics project at Harvard to search for untranslated RNAs (sRNAs) within bacterial replicons in the NCBI database
4) *Epigenomics* used by the Epigenome Center to automate different operations in genome sequence processing.

Fig. 5 provides an illustration of the topology of the workflows for each application, following the description in [23]. In our experiments, we considered small-scale (24–30 tasks), medium-scale (46–60 tasks), and large-scale (100 tasks) workflows. The exact number of tasks depends on the structure of each workflow application, more precisely: 30, 50, and 100 for CyberShake and Inspiral; 30, 60, and 100 for SIPHT; and 24, 46, and 100 for Epigenomics. For each workflow, the deadline was set to $\texttt{deadline} = \text{MF} + (\text{MS} - \text{MF}) \times \lambda$, with MF and MS the minimum and maximum execution times of the workflow and λ a scaling factor ranging between 0.005 and 0.155, with an increasing step of 0.05. The values of MF and MS have been obtained assuming to allocate each task to the fastest node (for MF) and to the slowest node (for MS) as determined by the HEFT algorithm [24] (a well-known heuristic that quickly computes a near-optimal makespan).

[1] https://pegasus.isi.edu/workflow_gallery/

TABLE I
CHARACTERISTICS OF NODES

Parameter	Value
CPU rate `cpu` (MIPS)	$[370, \ldots, 4500]$
Computation cost *Ccomp* ($ per hour)	$[0.067, \ldots, 0.8]$
Communication cost *Ccomm* ($ per hour)	$[0.655, \ldots, 0.75]$
Processing power `pproc` (Watt per hour)	$[250, \ldots, 500]$

3) Parameters: Table I summarizes the characteristics of the nodes considered in our experimental evaluation. The number of nodes was set according to the workflow size: five nodes for small-scale workflows, ten nodes for medium-scale workflows, and 15 nodes for large-scale workflows. The experimental results reported in the following have been obtained as the average over 20 independent runs. Each run considered a population of $k = 50$ solutions and maxIter = 100 iterations of the optimization process. The range $[L, U]$ for the values in matrix Y was set to $[-5, 5]$, which ensures that the transformed values produced by the sigmoid function remain in the range [0,1].

For CMaOAOA and CMaOAOAII, we configured the parameters as follows: $\mu = 0.499$ (according to [25]), $\kappa = 2$ (based on empirical results), CR = 0.9 (according to [26]), and SF = 0.2 (based on empirical results). For MaOPSO, CNSGAIII, and DCNSGAIII, we considered the parameter settings described in [9], [12], and [13], respectively.

B. Experimental Results

To analyze the effectiveness of CMaOAOAII, we measured the number and quality of the computed solutions in terms of the following objectives.

1) *Feasibility Rate:* Fraction of independent runs in which the algorithm generates at least a feasible solution.
2) *Success Rate:* Percentage of feasible solutions in the population generated by the last iteration (i.e., when g = maxIter).
3) *Hypervolume* (HV): Measure of the quality of the approximation in terms of both diversity and convergence [27]. It is computed as the volume of the region in the objective space that is dominated by the Pareto front approximation and bounded by a reference point. The reference point is set to $1.1\times$ the maximum objective values observed across the approximated Pareto front.

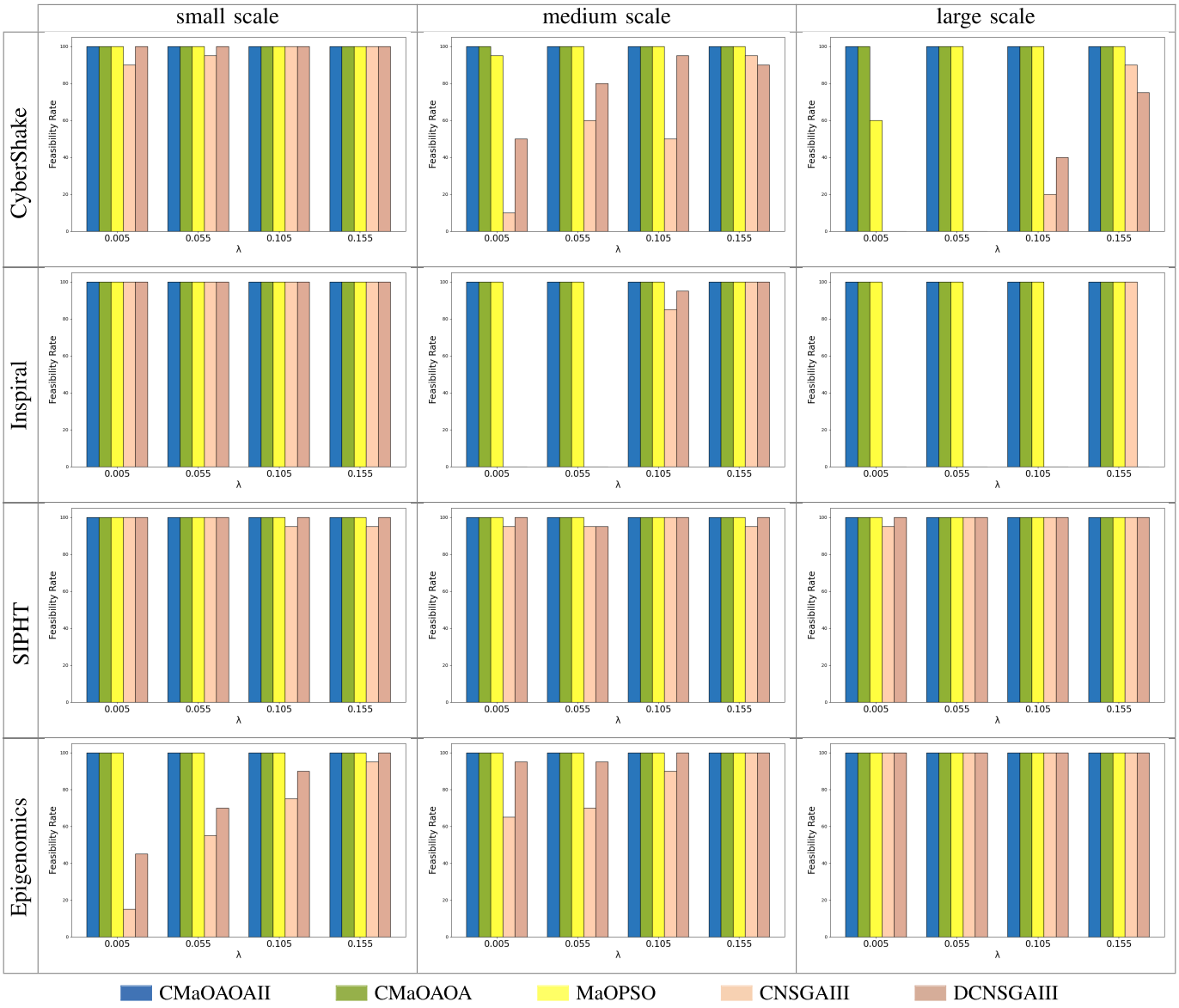

Fig. 6. Feasibility rate for CyberShake, Inspiral, SIPHT, and Epigenomics workflows varying deadline factor λ.

A higher HV value signifies a better distribution of solutions that are both diverse and well-converged toward the Pareto front, indicating a better approximation to the Pareto front.

1) Feasibility Rate: Fig. 6 reports the feasibility rate varying the deadline factor λ for different workflows. As it is visible from the figure, CMaOAOAII and CMaOAOA consistently produce feasible solutions across all the runs for all the tested workflows. MaOPSO computes feasible solutions across all the runs for most of the tested workflows, failing to find feasible solutions for few runs of CyberShake (medium- and large-scale workflows) with strict deadline constraints. The results in the figure confirm that, as expected, the feasibility rate generally increases as the deadline constraint becomes looser (i.e., higher values of λ) across all workflows.

2) Success Rate: Fig. 7 reports the success rate varying the deadline factor λ for different workflows. As expected, the success rate generally grows with λ. CMaOAOAII and CMaOAOA provide more feasible solutions than all the other algorithms in most of the considered configurations and scale well, with a success rate remaining consistently high under strict deadline even for medium and large workflows.

3) Hypervolume: Besides the rate of feasibility and success, it is important to analyze the quality of the solutions produced, meaning how well they approximate the Pareto front. As a matter of fact, a smaller number of good quality solutions is to be preferred to a higher number of solutions with poorer quality. Fig. 8 shows the comparison of the HV values (average and standard deviation) obtained with the five compared algorithms, varying λ, the workflow benchmark, and the number of tasks. For each tested configuration, we highlight with a darker background the best solution, and with a light background the second-best one. Value "Infeasible" means that the algorithm did not return feasible solutions in any of the runs. For each tested configuration, we report, for each algorithm, a symbol

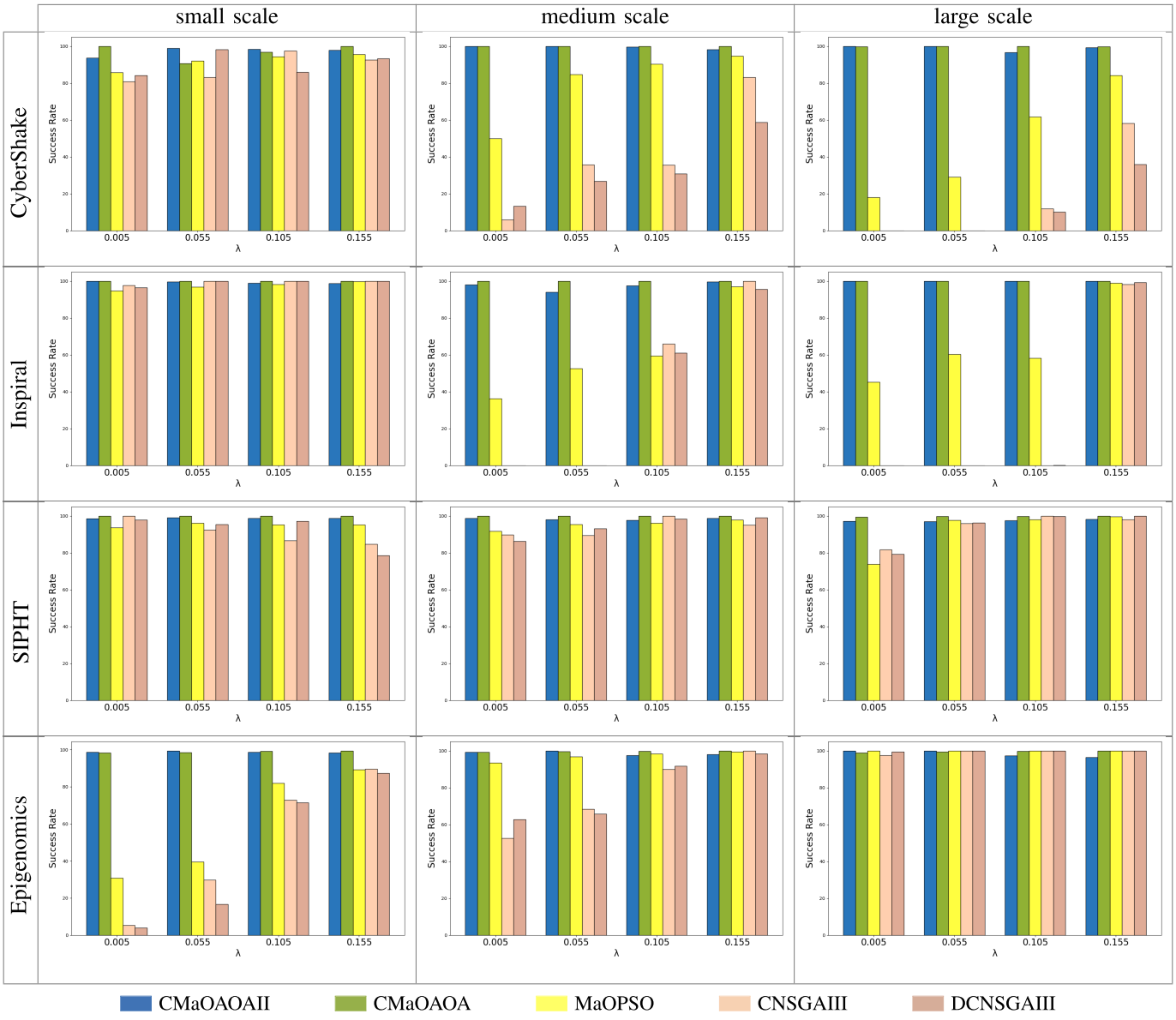

Fig. 7. Success rate for CyberShake, Inspiral, SIPHT, and Epigenomics workflows varying deadline factor λ.

(+, =, or −) indicating whether the results obtained with CMaOAOAII are statistically better, not significantly different, or worse than the one of the compared algorithm based on the results of a Wilcoxon signed-rank test [28] with a significance level of 0.05. The last row in the table summarizes the results of the Wilcoxon signed-rank test, indicating the number of configurations where CMaOAOAII performed better, equally, or worse than the algorithm to which the column refers. The results in the figure show that CMaOAOAII obtains the highest HV value in almost all the tested configurations, with CMaOAOA achieving the second best in the majority of them, outperforming the other tested algorithms. The superiority of CMaOAOAII is also confirmed by the results of the Wilcoxon signed-rank test: in all the tested configurations, CMaOAOAII obtains statistically better results than CNSGAIII and DCNSGAIII, and better than, or equal to, CMaOAOA and MaOPSO.

In summary, the experimental evaluation confirms the competitive performance of CMaOAOAII in addressing the constrained workflow scheduling problem.

VII. Discussion

CMaOAOAII effectively considers many objectives (i.e., makespan, load balance, economic cost, and energy consumption) together with the deadline constraint. In the following, we illustrate promising opportunities for further investigations, aimed at enriching and extending CMaOAOAII to address also other aspects.

A. Security and Privacy

A workflow may include tasks that perform critical operations or manage sensitive data. Such critical tasks require special care, and their allocation must be determined by considering possible restrictions or pReferences on the nodes

λ	Workflow	CMaOAOAII	CMaOAOA		MaOPSO		CNSGAIII		DCNSGAIII	
0.005	CyberShake_30	$2.802e+09_{1.20e+09}$	$2.303e+09_{8.35e+08}$	(=)	$1.397e+09_{7.47e+08}$	(+)	$6.742e+07_{7.41e+07}$	(+)	$1.030e+08_{9.75e+07}$	(+)
	CyberShake_50	$1.515e+10_{1.61e+09}$	$7.128e+09_{5.52e+09}$	(+)	$8.684e+09_{4.26e+09}$	(+)	$4.784e+07_{1.46e+08}$	(+)	$1.292e+08_{1.50e+08}$	(+)
	CyberShake_100	$2.282e+10_{4.63e+09}$	$2.236e+10_{1.03e+10}$	(=)	$1.103e+10_{9.87e+09}$	(+)	Infeasible	(+)	Infeasible	(+)
	Inspiral_30	$4.792e+10_{1.56e+10}$	$3.134e+10_{5.58e+09}$	(+)	$2.051e+10_{6.45e+09}$	(+)	$9.411e+09_{1.84e+09}$	(+)	$9.050e+09_{2.54e+09}$	(+)
	Inspiral_50	$2.909e+10_{9.65e+09}$	$1.440e+10_{4.17e+09}$	(+)	$2.183e+10_{9.92e+09}$	(+)	Infeasible	(+)	Infeasible	(+)
	Inspiral_100	$9.183e+11_{1.63e+11}$	$5.332e+11_{6.56e+10}$	(+)	$9.926e+11_{1.76e+11}$	(=)	Infeasible	(+)	Infeasible	(+)
	Sipht_30	$1.849e+10_{3.48e+09}$	$1.554e+10_{8.62e+09}$	(+)	$1.505e+09_{1.43e+09}$	(+)	$2.444e+08_{2.91e+08}$	(+)	$4.293e+08_{3.24e+08}$	(+)
	Sipht_60	$9.510e+11_{1.14e+11}$	$2.961e+11_{2.30e+11}$	(+)	$1.683e+11_{6.84e+10}$	(+)	$2.173e+10_{1.66e+10}$	(+)	$2.521e+10_{1.59e+10}$	(+)
	Sipht_100	$3.572e+12_{4.44e+12}$	$1.762e+12_{1.76e+11}$	(=)	$1.961e+11_{2.91e+11}$	(+)	$1.339e+11_{5.74e+10}$	(+)	$1.165e+11_{5.42e+10}$	(+)
	Epigenomics_24	$9.320e+11_{2.37e+11}$	$6.419e+11_{1.55e+11}$	(+)	$4.331e+11_{1.66e+11}$	(+)	$1.293e+10_{3.13e+10}$	(+)	$2.932e+10_{3.38e+10}$	(+)
	Epigenomics_46	$6.003e+12_{1.04e+12}$	$6.003e+12_{1.04e+12}$	(=)	$6.386e+12_{2.79e+12}$	(=)	$1.257e+11_{1.17e+11}$	(+)	$2.007e+11_{8.01e+10}$	(+)
	Epigenomics_100	$1.675e+16_{1.36e+15}$	$1.644e+16_{3.57e+15}$	(=)	$5.978e+15_{1.84e+15}$	(+)	$1.918e+14_{5.46e+13}$	(+)	$2.049e+14_{1.01e+14}$	(+)
0.055	CyberShake_30	$9.974e+08_{3.12e+08}$	$8.679e+08_{2.85e+08}$	(=)	$4.496e+08_{2.50e+08}$	(+)	$9.159e+07_{1.11e+08}$	(+)	$1.699e+08_{5.84e+07}$	(+)
	CyberShake_50	$2.059e+10_{6.18e+09}$	$1.144e+10_{1.70e+09}$	(+)	$1.256e+10_{4.44e+09}$	(+)	$2.786e+08_{3.39e+08}$	(+)	$4.580e+08_{3.84e+08}$	(+)
	CyberShake_100	$2.621e+10_{2.32e+09}$	$2.793e+09_{2.16e+09}$	(+)	$1.629e+10_{7.42e+09}$	(+)	Infeasible	(+)	Infeasible	(+)
	Inspiral_30	$6.211e+10_{1.54e+10}$	$3.274e+10_{5.75e+09}$	(+)	$2.047e+10_{7.66e+09}$	(+)	$5.186e+09_{1.43e+09}$	(+)	$5.745e+09_{1.83e+09}$	(+)
	Inspiral_50	$1.639e+12_{1.56e+11}$	$5.588e+10_{3.86e+10}$	(+)	$1.232e+11_{4.29e+10}$	(=)	Infeasible	(+)	Infeasible	(+)
	Inspiral_100	$3.571e+12_{5.84e+11}$	$2.841e+12_{3.37e+11}$	(+)	$3.561e+12_{2.52e+11}$	(=)	Infeasible	(+)	Infeasible	(+)
	Sipht_30	$5.639e+10_{1.62e+10}$	$3.460e+10_{1.38e+10}$	(+)	$5.770e+09_{5.29e+09}$	(+)	$3.520e+09_{2.02e+09}$	(+)	$4.564e+09_{1.99e+09}$	(+)
	Sipht_60	$1.876e+11_{6.84e+10}$	$9.497e+10_{9.77e+10}$	(+)	$2.640e+10_{1.42e+10}$	(+)	$4.334e+10_{2.07e+10}$	(+)	$4.040e+10_{1.72e+10}$	(+)
	Sipht_100	$4.726e+12_{7.28e+12}$	$4.146e+11_{9.56e+10}$	(+)	$1.858e+11_{1.56e+11}$	(+)	$2.368e+11_{1.59e+11}$	(+)	$2.576e+11_{1.46e+11}$	(+)
	Epigenomics_24	$1.653e+11_{4.92e+10}$	$5.557e+10_{1.97e+10}$	(+)	$4.421e+10_{2.22e+10}$	(+)	$8.777e+09_{1.06e+10}$	(+)	$8.831e+09_{7.91e+09}$	(+)
	Epigenomics_46	$2.699e+12_{7.90e+11}$	$1.358e+12_{4.97e+11}$	(+)	$2.056e+12_{8.89e+11}$	(+)	$3.487e+11_{3.63e+11}$	(+)	$2.339e+11_{2.27e+11}$	(+)
	Epigenomics_100	$1.094e+16_{4.49e+15}$	$1.022e+16_{4.92e+15}$	(=)	$4.498e+15_{1.05e+15}$	(+)	$9.158e+13_{4.62e+13}$	(+)	$8.772e+13_{3.05e+13}$	(+)
0.105	CyberShake_30	$1.062e+10_{1.68e+09}$	$8.533e+09_{1.68e+09}$	(+)	$5.063e+09_{1.83e+09}$	(+)	$1.133e+09_{3.62e+08}$	(+)	$1.173e+09_{1.99e+08}$	(+)
	CyberShake_50	$3.004e+10_{1.26e+10}$	$2.802e+10_{4.10e+09}$	(=)	$2.973e+10_{7.12e+09}$	(=)	$2.065e+08_{3.53e+08}$	(+)	$7.879e+08_{6.05e+08}$	(+)
	CyberShake_100	$2.360e+10_{2.10e+09}$	$2.278e+10_{6.43e+09}$	(=)	$1.533e+10_{5.11e+09}$	(+)	$7.722e+07_{2.91e+08}$	(+)	$5.233e+08_{9.70e+08}$	(+)
	Inspiral_30	$1.171e+11_{1.78e+10}$	$5.886e+10_{1.05e+10}$	(+)	$4.160e+10_{1.79e+10}$	(+)	$7.606e+09_{2.57e+09}$	(+)	$7.040e+09_{4.52e+09}$	(+)
	Inspiral_50	$2.073e+12_{1.73e+11}$	$1.056e+12_{7.69e+10}$	(+)	$1.708e+12_{2.04e+11}$	(+)	$2.177e+09_{3.61e+09}$	(+)	$4.400e+09_{4.55e+09}$	(+)
	Inspiral_100	$7.802e+12_{3.33e+11}$	$6.187e+12_{4.67e+11}$	(+)	$6.665e+12_{9.09e+11}$	(+)	Infeasible	(+)	$3.769e+09_{1.64e+10}$	(+)
	Sipht_30	$2.063e+11_{2.52e+10}$	$9.748e+10_{2.17e+10}$	(+)	$5.498e+10_{2.29e+10}$	(+)	$3.799e+09_{1.53e+09}$	(+)	$4.025e+09_{2.26e+09}$	(+)
	Sipht_60	$5.315e+11_{1.26e+11}$	$9.441e+10_{1.11e+11}$	(+)	$8.234e+10_{7.15e+10}$	(+)	$3.591e+10_{2.34e+10}$	(+)	$5.742e+10_{3.15e+10}$	(+)
	Sipht_100	$5.807e+12_{7.48e+12}$	$1.763e+12_{3.77e+11}$	(+)	$8.508e+11_{1.22e+12}$	(+)	$1.133e+11_{1.34e+11}$	(+)	$1.855e+11_{1.30e+11}$	(+)
	Epigenomics_24	$1.081e+12_{2.23e+11}$	$5.329e+11_{1.20e+11}$	(+)	$4.136e+11_{1.89e+11}$	(+)	$3.512e+09_{6.99e+09}$	(+)	$2.974e+09_{6.73e+09}$	(+)
	Epigenomics_46	$2.947e+13_{2.54e+12}$	$2.054e+13_{2.15e+12}$	(+)	$2.335e+13_{3.56e+12}$	(+)	$2.297e+11_{2.37e+11}$	(+)	$3.879e+11_{1.79e+11}$	(+)
	Epigenomics_100	$2.311e+15_{4.47e+14}$	$2.204e+15_{9.85e+14}$	(=)	$1.590e+15_{4.21e+14}$	(+)	$1.780e+15_{2.66e+14}$	(+)	$1.760e+15_{3.14e+14}$	(+)
0.155	CyberShake_30	$5.978e+09_{7.60e+08}$	$4.647e+09_{7.40e+08}$	(+)	$3.270e+09_{1.03e+09}$	(+)	$6.263e+07_{7.24e+07}$	(+)	$1.339e+08_{8.36e+07}$	(+)
	CyberShake_50	$5.741e+10_{1.13e+10}$	$3.917e+10_{5.75e+09}$	(+)	$4.489e+10_{6.99e+09}$	(+)	$5.423e+08_{4.45e+08}$	(+)	$1.400e+09_{9.94e+08}$	(+)
	CyberShake_100	$1.158e+11_{7.65e+09}$	$1.189e+11_{2.47e+10}$	(=)	$9.071e+10_{1.72e+10}$	(+)	$7.895e+09_{3.22e+09}$	(+)	$6.837e+09_{4.05e+09}$	(+)
	Inspiral_30	$1.190e+11_{1.57e+10}$	$6.526e+10_{9.25e+09}$	(+)	$5.123e+10_{2.00e+10}$	(+)	$5.040e+09_{2.03e+09}$	(+)	$6.349e+09_{3.52e+09}$	(+)
	Inspiral_50	$1.321e+12_{2.17e+11}$	$5.312e+11_{6.87e+10}$	(+)	$1.177e+12_{1.88e+11}$	(+)	$5.362e+10_{1.56e+10}$	(+)	$5.809e+10_{7.58e+09}$	(+)
	Inspiral_100	$8.157e+12_{4.11e+11}$	$7.162e+12_{5.21e+11}$	(+)	$7.110e+12_{7.21e+11}$	(+)	$3.527e+11_{6.85e+10}$	(+)	$3.406e+11_{7.46e+10}$	(+)
	Sipht_30	$1.463e+11_{1.93e+10}$	$4.715e+10_{1.59e+10}$	(+)	$2.773e+10_{2.29e+10}$	(+)	$1.275e+10_{8.54e+09}$	(+)	$2.330e+10_{1.25e+10}$	(+)
	Sipht_60	$3.237e+11_{5.81e+10}$	$7.844e+10_{9.71e+10}$	(+)	$3.917e+10_{3.74e+10}$	(+)	$4.660e+10_{2.81e+10}$	(+)	$5.237e+10_{3.22e+10}$	(+)
	Sipht_100	$6.074e+12_{5.96e+12}$	$1.562e+12_{4.62e+11}$	(+)	$2.229e+12_{3.92e+12}$	(+)	$1.038e+12_{4.09e+11}$	(+)	$1.147e+12_{5.39e+11}$	(+)
	Epigenomics_24	$1.176e+12_{2.66e+11}$	$5.906e+11_{8.04e+10}$	(+)	$4.017e+11_{1.88e+11}$	(+)	$1.557e+10_{1.31e+10}$	(+)	$1.691e+10_{1.05e+10}$	(+)
	Epigenomics_46	$4.392e+13_{3.60e+12}$	$2.881e+13_{4.90e+12}$	(+)	$3.421e+13_{7.14e+12}$	(+)	$5.568e+11_{5.26e+11}$	(+)	$5.011e+11_{2.63e+11}$	(+)
	Epigenomics_100	$4.693e+14_{1.78e+14}$	$2.025e+14_{6.53e+13}$	(+)	$2.721e+14_{9.42e+13}$	(+)	$1.209e+14_{1.58e+14}$	(+)	$8.648e+13_{9.16e+13}$	(+)
Better/Equal/Worse			37/11/0		43/5/0		48/0/0		48/0/0	

Fig. 8. HV of CMaOAOAII, CMaOAOA, MaOPSO, CNSGAIII, and DCNSGAIII varying deadline factor λ, workflow benchmark, and the number of tasks.

that can execute them [29]. For instance, tasks processing sensitive information should be allocated to nodes that are compliant with specific regulations, use encryption for storing or transferring data, or are characterized by a level of trust higher than a threshold. In addition to these task-level constraints, other constraints can also be defined at the workflow level. For instance, a constraint can require some tasks to be allocated to different nodes to prevent their joint visibility to the same node (similar to what is done in [30]). For workflows with critical tasks, it is, therefore, fundamental to extend the modeling of the workflow scheduling problem to take into account also security and privacy constraints in the computation of an allocation.

B. Node/Device Parameters

Our model explicitly considers the processing power and bandwidth associated with nodes, which impact the execution and transmission time of tasks when allocated to nodes. The model could be extended by considering other node parameters (e.g., for scenarios where nodes correspond to devices that may have limited memory or storage capacity). Our approach could be extended in this direction in a way similar to how security/privacy constraints can be considered. For instance, resource demands (e.g., storage or memory) can be associated with tasks and resource capacity associated with nodes. The allocation should ensure that tasks are allocated to nodes that have the resources required by the task. If balance in the use of resources is sought (e.g., similar to what is done for load balance), the model could be extended with objectives related to the optimization and balancing of multiple resource-related objectives.

C. Multiple Workflows

CMaOAOAII considers the problem of allocating tasks in a workflow. An interesting future direction of investigation is the consideration of multiple workflows, by the presence

of different users or applications that need to independently execute multiple workflows on the same set of nodes. When multiple workflows (fully or partially) share the same set of nodes, the workflow scheduling problem would need to consider additional constraints, which are needed to guarantee that workflows do not interfere with each other. For instance, additional constraints can include the consideration of priorities among workflows or restrictions on their allocation (e.g., to enforce a separation/confinement of workflows with respect to nodes to which they are allocated).

D. Very Large-Scale Workflows

Similar to other approaches in the literature, CMaOAOAII has been designed and tested to operate with medium-to-large-scale workflows. The consideration of scenarios characterized by a huge number of tasks composing the workflow requires specific attention because the solution space is expected to grow considerably with the number of tasks. Therefore, its exploration for the identification of a set of solutions that well approximates the Pareto front becomes more and more computationally expensive. It would be interesting to test, and possibly adapt, CMaOAOAII to effectively and efficiently operate in very large-scale scenarios.

VIII. Conclusion

We addressed the problem of workflow scheduling to enable the IoT devices to efficiently rely on external services for data access and computation. Responding to the richness and diversity of requirements to be taken into account to meet applications demand, we formulated the problem as a CMaOP. The many objectives capture different aspects related to QoSs, such as makespan, load balance, economic cost, and energy consumption. The deadline captures the possible constraint on completion time for time-critical applications. We, then, proposed a novel meta-heuristic evolutionary computing approach for its solution. The extensive experimental evaluation, including comparison with alternative approaches in the literature and considering different real-world workflows, confirms the validity of our approach. This article leaves space for future work, including along the directions of investigation discussed in Section VII.

References

[1] L. Abualigah, A. Diabat, S. Mirjalili, M. A. Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Comput. Methods Appl. Mech. Eng.*, vol. 376, Apr. 2021, Art. no. 113609.

[2] A. Ali et al., "Multiobjective Harris hawks optimization-based task scheduling in cloud-fog computing," *IEEE Internet Things J.*, vol. 11, no. 13, pp. 24334–24352, Jul. 2024.

[3] X. Chen et al., "A WOA-based optimization approach for task scheduling in cloud computing systems," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3117–3128, Sep. 2020.

[4] H. Li, J. Liu, L. Yang, L. Liu, and H. Sun, "An improved arithmetic optimization algorithm for task offloading in mobile edge computing," *Cluster Comput.*, vol. 27, no. 2, pp. 1667–1682, Apr. 2024.

[5] Q. Jiang, X. Xin, T. Zhang, and K. Chen, "Energy-efficient task scheduling and resource allocation in edge-heterogeneous computing systems using multiobjective optimization," *IEEE Internet Things J.*, vol. 12, no. 17, pp. 36747–36764, Sep. 2025.

[6] Z. Fang, H. Yu, G. Fan, Z. Li, and J. Zhang, "Energy, cost and reliability-aware workflow scheduling on multi-cloud systems: A multi-objective evolutionary approach," *IEEE Trans. Netw. Service Manage.*, vol. 22, no. 5, pp. 4788–4805, Oct. 2025.

[7] X. Cai, Y. Zhang, M. Li, L. Wu, W. Zhang, and J. Chen, "Dynamic deadline constrained multi-objective workflow scheduling in multi-cloud environments," *Expert Syst. Appl.*, vol. 258, Dec. 2024, Art. no. 125168.

[8] S. Qin, D. Pi, and Z. Shao, "A two-stage learning-driven many-objective memetic algorithm for solving the workflow scheduling problem in cloud environment," *IEEE Trans. Services Comput.*, vol. 18, no. 5, pp. 2574–2587, Sep. 2025.

[9] S. Saeedi, R. Khorsand, S. Ghandi Bidgoli, and M. Ramezanpour, "Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing," *Comput. Ind. Eng.*, vol. 147, pp. 1–23, Sep. 2020.

[10] L. Ye, L. Yang, Y. Xia, and X. Zhao, "A cost-driven intelligence scheduling approach for deadline-constrained IoT workflow applications in cloud computing," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 16033–16047, May 2024.

[11] X. Ye, S. Liu, Y. Yin, and Y. Jin, "User-oriented many-objective cloud workflow scheduling based on an improved knee point driven evolutionary algorithm," *Knowl.-Based Syst.*, vol. 135, pp. 113–124, Nov. 2017.

[12] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, Aug. 2014.

[13] R. Jiao, S. Zeng, C. Li, S. Yang, and Y.-S. Ong, "Handling constrained many-objective optimization problems via problem transformation," *IEEE Trans. Cybern.*, vol. 51, no. 10, pp. 4834–4847, Oct. 2021.

[14] Z. Wang, W. Zhan, H. Duan, G. Min, and H. Huang, "Deep-reinforcement-learning-based continuous workflows scheduling in heterogeneous environments," *IEEE Internet Things J.*, vol. 12, no. 10, pp. 14036–14050, May 2025.

[15] B. Mao, X. Zhou, J. Liu, and N. Kato, "On a cooperative deep reinforcement learning-based multi-objective routing strategy for diversified 6G metaverse services," *IEEE Trans. Veh. Technol.*, vol. 73, no. 9, pp. 14092–14096, Sep. 2024.

[16] Y. Huang, H. Xu, Q. Ran, W. Wei, and H. Gao, "GRWS: A deep reinforcement learning method with graph attention networks for flexible workflow scheduling in industrial manufacturing scenarios," *IEEE Internet Things J.*, early access, 2025.

[17] B. Mao, J. Qiu, and N. Kato, "On an intelligent task offloading model to jointly optimize latency and energy for electric connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 73, no. 4, pp. 6024–6028, Apr. 2024.

[18] B. Mao, Y. Kawamoto, and N. Kato, "AI-based joint optimization of QoS and security for 6G energy harvesting Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7032–7042, Aug. 2020.

[19] N. Kouka, S. De Capitani di Vimercati, V. Piuri, and P. Samarati, "An arithmetic optimization algorithm with dynamic state estimation for deadline-constrained workflow scheduling," in *Proc. Genetic Evol. Comput. Conf. Companion*, Malaga, Spain, Jul. 2025, pp. 379–382.

[20] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[21] P. Sharma, H. Sharma, S. Kumar, and J. C. Bansal, "A review on scale factor strategies in differential evolution algorithm," in *Proc. Soft Comput. Problem Solving*, Dec. 2017, pp. 925–943.

[22] J. J. Durillo and A. J. Nebro, "JMetal: A Java framework for multi-objective optimization," *Adv. Eng. Softw.*, vol. 42, no. 10, pp. 760–771, Oct. 2011.

[23] E. Deelman et al., "The evolution of the pegasus workflow management software," *Comput. Sci. Eng.*, vol. 21, no. 4, pp. 22–36, Jul. 2019.

[24] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[25] L. Peng, C. Sun, and W. Wu, "Effective arithmetic optimization algorithm with probabilistic search strategy for function optimization problems," *Data Sci. Manage.*, vol. 5, no. 4, pp. 163–174, Dec. 2022.

[26] J. Montgomery and S. Chen, "An analysis of the operation of differential evolution at high and low crossover rates," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–8.

[27] L. While, L. Bradstreet, and L. Barone, "A fast way of calculating exact hypervolumes," *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 86–95, Feb. 2012.
[28] W. Conover, *Practical Nonparametric Statistics*. Hoboken, NJ, USA: Wiley, 1999.
[29] S. De Capitani Di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati, "Supporting user requirements and preferences in cloud plan selection," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 274–285, Jan. 2021.
[30] S. De Capitani di Vimercati, S. Foresti, G. Livraga, V. Piuri, and P. Samarati, "Security-aware data allocation in multicloud scenarios," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2456–2468, Sep. 2021.

Najwa Kouka (Member, IEEE) is a Post-Doctoral Researcher with the Università degli Studi di Milano, Milan, Italy. Her research interests are in optimization problems (multiobjective, many-objective optimization, and dynamic optimization), and the application of evolutionary algorithms in real-world scenarios. More details can be found at https://kouka.di.unimi.it

Sabrina De Capitani di Vimercati (Fellow, IEEE) is a Professor with the Università degli Studi di Milano, Milan, Italy. She has been a Visiting Researcher with SRI International, Menlo Park, CA, USA, and George Mason University, Fairfax, VA, USA. She has published more than 240 articles in journals, conference proceedings, and books. Her research interests are in data security and privacy. More details can be found at https://decapitani.di.unimi.it

Sara Foresti (Senior Member, IEEE) is a Professor with the Università degli Studi di Milano, Milan, Italy. She has been a Visiting Researcher with George Mason University, Fairfax, VA, USA. She has published more than 120 articles in journals, conference proceedings, and books. Her research interests are in data security and privacy. More details can be found at https://foresti.di.unimi.it

Vincenzo Piuri (Life Fellow, IEEE) is a Professor with the Università degli Studi di Milano, Milan, Italy. He has been a Visiting Researcher with The University of Texas at Austin, Austin, TX, USA, and George Mason University, Fairfax, VA, USA. He has published more than 350 articles in journals, conference proceedings, and book chapters. His main research interests are signal and image processing, biometrics, intelligent systems, and digital and signal processing architectures.

Dr. Piuri is a fellow of IFIP. More details can be found at https://piuri.di.unimi.it

Pierangela Samarati (Fellow, IEEE) is a Professor with the Università degli Studi di Milano, Milan, Italy. She has been a Visiting Researcher with Stanford University, Stanford, CA, USA, SRI International, Menlo Park, CA, USA, and George Mason University, Fairfax, VA, USA. She has published more than 300 articles in journals, conference proceedings, and books. Her main research interests are in data protection, security, and privacy.

Dr. Samarati is a fellow of ACM and IFIP. More details can be found at https://samarati.di.unimi.it

Open Access funding provided by 'Università degli Studi di Milano' within the CRUI CARE Agreement