Towards Efficient Allocation of Tasks in DAG-based Workflows across Federated Fog Systems

Gayani Gupta[1], Amisha Gupta[2], Mohsen Amini Salehi[1], Antonio Fernández Anta[3],
Sanjukta Bhowmick[1], Jose Aguilar[3], Mathieu Kourouma[4]
[1]University of North Texas, USA
[2]University of California, Berkeley, USA
[3]IMDEA Software Institute & IMDEA Networks Institute, Spain
[4]Southern University and A&M College, Louisiana, USA
[1]{gayani.gupta,mohsen.aminisalehi,sanjukta.bhowmick}@unt.edu, [2]amisha.gupta@berkeley.edu
[3]{antonio.fernandez,jose.aguilar}@imdea.org, [4]mathieu_kourouma@subr.edu

***Abstract*—DAG-based (Directed Acyclic Graph) applications, deployed on fog computing platforms (mini-data centers), are at the core of modern industrial systems—especially those in remote locations, which are often subjected to harsh, disaster-prone conditions and frequently face unreliable or no cloud connectivity. When disasters occur and multiple time-sensitive DAGs need to be processed, the inherent inelasticity of local fog systems arises as a barrier to meeting latency constraints. To achieve cloud-like elasticity in such resource-limited environments, we propose resource allocation strategies based on Mixed-Integer Linear Programming (MILP) to optimally "partition" the DAG and "approximately process" its tasks in a federated fog system. To make our approximate computing strategy feasible for large-scale DAGs, we developed a second method based on the greedy approach. Our experimental results demonstrate that our strategies establish elasticity across federated fog systems, as well as minimize the approximation level while meeting the deadline constraints of the DAGs.**

I. Introduction and Motivation

The Fourth Industrial Revolution (Industry 4.0) has accelerated the adoption of autonomous, data-driven workflows in industrial systems, demanding low-latency and serverless processing in remote environments such as offshore oil fields [1], space stations [2], and underwater robotic platforms [3]. These systems are characterized by intermittent connectivity and scarce human resources, making cloud-dependent solutions in them impractical. However, many critical industrial applications, such as disaster response, predictive maintenance, and real-time environmental monitoring, must be executed locally under strict latency and resource constraints to ensure system safety and operational efficiency [4], [5].

Fog computing has emerged as a localized processing solution, enabling edge devices to handle complex, real-time tasks without relying on the cloud [6]. However, standalone fog nodes often lack the computational elasticity to process multiple concurrent workflows during emergencies such as oil spills or equipment failures [7]. For instance, Figure 1 illustrates the workflow for disaster response in the event of a fire, where multiple tasks must be executed within a deadline. These include tasks for data pre-processing, fire detection, location mapping, alarm notification, evacuation planning simulation, fire expansion simulation, and granular surveillance planning. The dependencies among these tasks form a Directed Acyclic Graph (DAG), where each node represents a distinct task. Efficient and timely execution of the DAG requires intelligent resource allocation strategies to ensure all tasks meet their stringent deadlines. To address the resource limitations of individual fog nodes, we leverage federated fog computing, where nearby fog systems collaborate to provide shared computational resources and improve elasticity [8]. This setup allows efficient execution of DAG-based industrial workflows but also introduces scheduling complexities due to heterogeneous resources and variable network latencies [9]. *Allocating interdependent tasks across federated nodes while minimizing latency and meeting deadlines is a significant challenge*, especially for Industry 4.0 workloads that require deterministic execution.

Here, *we propose and compare two algorithms for allocating interdependent tasks in DAG-based workflows on federated fog systems.* The first is a Mixed-Integer Linear Programming (MILP) model that optimizes task placement based on resource capacities, execution costs, and communication delays. The second is a Greedy approach that incrementally assigns tasks to minimize overall completion time. While the MILP formulation guarantees optimality, it becomes computationally intractable for large-scale workflows. In contrast, the Greedy approach offers improved scalability, making it more suitable for resource-constrained fog environments. To the best of our knowledge, this is the first work to formulate and systematically compare MILP and Greedy task allocation strategies for DAG scheduling in federated fog computing systems.

II. Our Contributions

Figure 2 shows a federated fog computing network (FG_1–FG_3), deployed to support real-time fire detection and emergency response applications at offshore oil fields [9]–[11]. The fire detection workflow is a DAG, where each task—i.e., sensor data collection, smoke and heat analysis, fire localization, and alert dissemination—is represented as a node with dependencies. Real-time fire detection systems

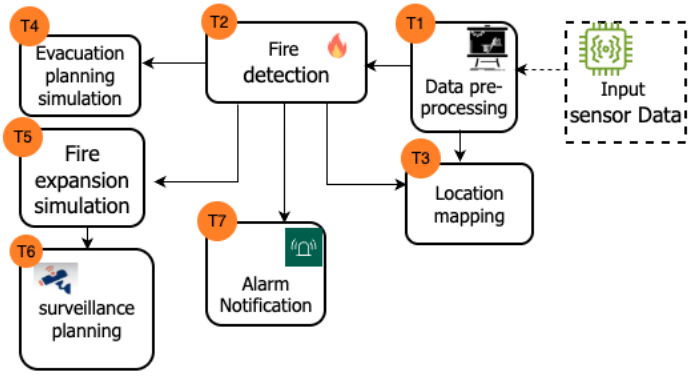

Fig. 1: A DAG-based fire detection workflow illustrating microservices distributed across a federated fog computing system.

must process large volumes of sensor data and trigger response actions with minimal delay. In a federated fog computing environment, where resources are distributed across geographically dispersed and heterogeneous nodes, the challenge is to map the tasks onto the available fog infrastructure that ensures timely execution, reduces communication overhead, and adapts to dynamic resource availability. Achieving this requires intelligent task scheduling and resource allocation strategies that optimize latency, resilience, and responsiveness under uncertainty.

Proposed Solutions: A *Mixed-Integer Linear Programming (MILP)* model enables *optimal task placement* under resource and bandwidth constraints. To address MILP's *scalability limitations*, a lightweight *Greedy approach* provides fast, adaptive scheduling under dynamic conditions. The strategy supports *parallel execution*, reduces response time, and ensures *real-time performance* in mission-critical scenarios such as oil spill mitigation and *DAG-based fire detection*. A synthetic DAG, modeled on a real-world fire detection workflow, is used to evaluate scheduling behavior across varying slack, fog node counts, and DAG sizes (Fig. 3), illustrating *trade-offs* between accuracy and scalability under realistic constraints.

Key contributions: (1) *A DAG-to-Federated Fog mapping strategy* for disaster-response workflows, accounting for resource constraints and heterogeneity; (2) *Two resource allocation methods*: a MILP model for optimal task assignment and a greedy heuristic for scalable, adaptive scheduling; and (3) *A simulation-based evaluation* highlighting *trade-offs* between optimality and scalability, demonstrating improved efficiency and deadline compliance.

III. Related Work

Prior research in fog computing has examined task scheduling and resource optimization for DAG-based workflows in heterogeneous environments. Heuristic and reinforcement learning methods [12], [13] improve scheduling efficiency but often neglect communication overhead and scalability in federated fog settings. MILP-based models [14] offer optimal

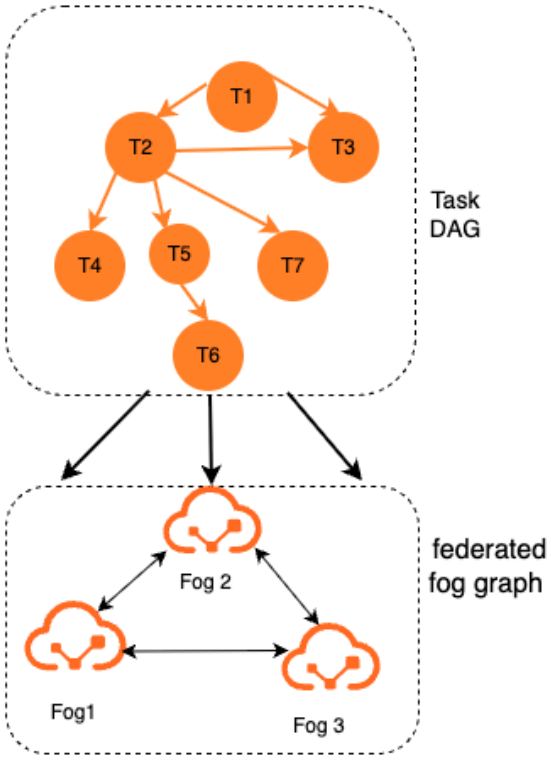

Fig. 2: Mapping of a fire detection workflow DAG onto a federated fog computing infrastructure.

task allocation but lack integrated resource modeling. Predictive and adaptive greedy techniques [15] focus on balancing execution time and energy use, yet often overlook the trade-off between accuracy and real-time responsiveness, which is critical in disaster-response workflows with strict deadlines.

Our work addresses these limitations by *proposing a MILP model for optimal task mapping and a lightweight greedy approach for scalable, low-latency, resource-aware scheduling in federated fog systems*. A key distinction is our explicit evaluation of the accuracy-performance *trade-off* under varying slack, federated fog node counts, and DAG sizes—dimensions often underexplored in prior work.

IV. Problem Definition and Modeling

We aim to solve a dependent task allocation problem in a fog federation environment. Let $T = \{T_1, T_2, \ldots, T_n\}$ be a set of tasks to be executed, and $F = \{f_1, f_2, \ldots, f_m\}$ be a set of available fog nodes in which the tasks can be executed. A solution to the problem assigns each task T_i to exactly one fog f_a from those in which it is allowed (indicated with variables $P_{ia} \in \{0, 1\}$). Task T_i assigned to fog f_a takes time $t_{ia} \geq 0$ to be executed completely. However, if required, it can be executed only partially. If executed completely, task T_i achieves accuracy $A_i > 0$. The achieved accuracy decreases linearly when executed partially.

A DAG represents the dependencies between tasks, where a directed edge from task T_j to task T_i indicates that T_i needs data produced by T_j to start its execution. If T_j and T_i are

TABLE I: Summary of Model Parameters and Variables

Symbol	Description
t_{ia}	Time to execute completely task T_i on fog node f_a
A_i	Accuracy if T_i is fully executed; else decreases linearly
$t_{\max}$	Maximum allowable completion time for tasks
c_{ji}^{ba}	Comm. delay from task T_j on fog f_b to task T_i on fog f_a
P_{ia}	Binary variable indicating if task T_i is allowed on fog f_a
l_{ia}	Fraction of task T_i executed on fog node f_a
e_{ia}	Binary variable indicating if task T_i is assigned to fog f_a
y_i	Completion time for task T_i
M_{ia}	Earliest start time of task T_i on fog node f_a

executed in the same fog node f_a, the execution of T_i can start immediately after the execution of T_j ends. Otherwise, if T_j and T_i are executed in different fogs f_b and f_a, respectively, T_i has to wait $c_{ji}^{ba} \geq 0$ communication time for the data from T_j to be received. For convenience we set $c_{ji}^{ba} = 0$ for $f_b = f_a$.

Our objective is to assign tasks to fogs so that the whole computation ends by a deadline $t_{\max} > 0$. To do so, some tasks may have to be executed partially, so a second goodness criterion is to maximize the average accuracy of the solution. In safety-critical environments such as *emergency response at offshore oil fields*, tolerable accuracy loss is highly *application-dependent*. For instance, early *anomaly detection systems* may accept a *10–15%* reduction in accuracy if it results in significantly faster response times, whereas *mission planning* or *control decisions* may require near-complete fidelity. Quantifying such thresholds remains an open challenge and depends on both *operational risk tolerance* and *system-level redundancy*. The system operates under the following assumptions:

- Tasks are not preemptive, i.e., a task cannot be started in one fog, stopped, moved to another fog, and continued.
- Each tasks is executed in exactly one fog.
- A task can receive data from all in-neighbors simultaneously.
- Executing multiple tasks concurrently in the same fog does not affect their execution time, we assume the fog servers utilize space-sharing scheduling and isolation mechanisms, minimizing interference between tasks [2].
- Accuracy of a workflow is driven by the average accuracy of the tasks in that workflow.

The task allocation problem is computationally intractable, falling into the class of NP-hard problems [16]. As a result, solving this problem optimally likely requires exponential time in the worst case, making it impractical for large-scale instances. Consequently, exact solutions are typically limited to moderately sized problem instances, and heuristic or approximation approaches are preferred for scalability.

V. Task Assignment using MILP

We present a formulation of the problem as a mixed-integer linear program (MILP). The key parameters and decision variables used in the formulation are given in Table I. The input parameters of the problem are t_{ia}, A_i, $t_{\max}$, c_{ji}^{ba}, and P_{ia}, for tasks T_i and T_j, and fog nodes f_a and f_b. The variables of the MILP formulation are $l_{ia} \in [0,1]$, $e_{ia} \in \{0,1\}$, $y_i \in [0, t_{\max}]$, and $M_{ia} \in [0, t_{\max}]$, for tasks T_i and fog nodes f_a. We denote the set of in-neighbor tasks of T_i in the dependencies DAG by in_i.

A. Basic Formulation

Objective:

$$\text{Maximize } \frac{1}{n} \sum_{T_i \in T} A_i \max_{f_a \in F} \{l_{ia}\} \tag{1}$$

Subject to

$$e_{ia} \geq l_{ia}, \quad \forall T_i \in T, f_a \in F \tag{2}$$

$$\sum_{f_a \in F} e_{ia} = 1, \quad \forall T_i \in T \tag{3}$$

$$M_{ia} = \max_{T_j \in \text{in}_i} \left(y_j + \sum_{f_b \in F} e_{jb} \cdot c_{ji}^{ba} \right), \forall T_i \in T, f_a \in F \tag{4}$$

$$y_i = \sum_{f_a \in F} e_{ia} \left(M_{ia} + t_{ia} l_{ia} \right), \quad \forall T_i \in T \tag{5}$$

$$y_i \leq t_{\max}, \quad \forall T_i \in T \tag{6}$$

$$e_{ia} \leq P_{ia}, \quad \forall T_i \in T, f_a \in F \tag{7}$$

For simplicity, in Eq. 4 we assume that when $\text{in}_i = \emptyset$ the maximum is 0, and hence $M_{ia} = 0$.

B. Correctness of Basic Formulation

Let us consider a solution to the optimization problem defined in Section V-A. Observe that the solution is completely determined by the values of variables l_{ia} and $e_{ia}, \forall T_i \in T, f_a \in F$. Let us show the properties of the solution (l_{ia}, e_{ia}).

Lemma 1. *If $P_{ia} = 0$, task T_i is not assigned to fog f_a.*

Proof. If $P_{ia} = 0$, from Equation (7) $e_{ia} \leq P_{ia} = 0$. Since $e_{ia} \in \{0,1\}$, it must hold that $e_{ia} = 0$. □

Lemma 2. *Every task T_i is assigned to exactly one fog.*

Proof. From Equation (3), and the fact that $e_{ib} \in \{0,1\}$, we have exactly one variable $e_{ia} = 1$ for each task T_i. □

Lemma 3. *For each task T_i at most one variable l_{ia} is larger than 0.*

Proof. Lemma 2, Equation (2), and $e_{ia} \in \{0,1\}$ imply that at most one $l_{ia} > 0$ for each task T_i. □

Lemma 4. *For each task T_i and fog f_a, if $l_{ia} > 0$ then $e_{ia} = 1$.*

Proof. Follows from Equation (2) and $e_{ia} \in \{0,1\}$. □

Lemma 5. *M_{ia} is the earliest time task T_i can start if assigned to fog f_a, and y_i is the time to approximate task T_i to level l_{ia} in the fog f_a to which it was assigned ($e_{ia} = 1$).*

Proof. We use induction on the maximal distance to a source. A task T_i is a source if its set of in-neighbors, in_i, is empty.

The base case is when a task is at a distance 0 from a source (i.e., is a source itself). For a source task T_i, $\text{in}_i = \emptyset$ and hence $M_{ia} = 0, \forall f_a$, from Equation (4). Let us assume

T_i is assigned to fog f_a, i.e, $e_{ia} = 1$. From Lemma 2 it holds that $e_{ib} = 0, \forall f_b \neq f_a$. Hence, Equation (5) becomes

$$y_i = M_{ia} + t_{ia} l_{ia} \tag{8}$$
$$= t_{ia} l_{ia}, \tag{9}$$

– the time to execute task T_i to a fraction l_{ia} in fog f_a.

We assume by induction that for all tasks T_j whose maximal distance to a source is less than $k > 0$, M_{jb} is the time T_j can start if assigned to fog f_b, and y_j is the time to execute task T_j to fraction l_{jb^*} in the fog f_{b^*} to which it was assigned.

Let us consider a task T_i whose maximal distance to a source is $k > 0$. Then, all the tasks in the set in_i have maximal distance from a source smaller than k. Let us consider any $T_j \in \text{in}_i$ and assume T_j was assigned to fog f_c (i.e., $e_{jc} = 1$). Then, if T_i is assigned to fog f_a, it must wait until the output data from T_j at f_c is transferred to fog f_a. By induction hypothesis, y_j is the time to approximate task T_j to level l_{jc} in fog f_c. Hence, T_i cannot be started before time $y_j + c^{ca}_{ji}$. From Lemma 2 it holds that $e_{jb} = 0, \forall f_b \neq f_c$. Hence, $c^{ca}_{ji} = \sum_{f_b \in F} e_{jb} \cdot c^{ba}_{ji}$.

Since this holds for all $T_j \in \text{in}_i$, the time T_i can start if assigned to fog f_a is

$$M_{ia} = \max_{T_j \in \text{in}_i} \left(y_j + \sum_{f_b \in F} e_{jb} \cdot c^{ba}_{ji} \right)$$

Moreover, if T_i is assigned to fog f_a, then $e_{ia} = 1$ and $e_{ib} = 0, \forall f_b \neq f_a$, and the time to execute task T_i to fraction l_{ia} in the fog f_a is

$$y_i = \sum_{f_b \in F} e_{ib} \left(M_{ib} + t_{ib} l_{ib} \right)$$
$$= M_{ia} + t_{ia} l_{ia}$$

□

Lemma 6. *All tasks complete their execution by time* $t_{\max}$.

Proof. The claim follows from the previous lemma and Equation (6). □

Definition 1. *We say that a solution* (l^*_{ia}, e^*_{ia}) *is feasible if it satisfies Constraints 2-7, i.e.,*

- *If* $P_{ia} = 0$ *then* $e^*_{ia} = 0$.
- *For each task* T_i, $l^*_{ia} > 0$ *for at most one fog.*
- *For each task* T_i, $e^*_{ia} = 1$ *for exactly one fog.*
- *For each task* T_i *and fog* f_a, *if* $l^*_{if} > 0$ *then* $e^*_{if} = 1$.
- *All tasks complete by time* $t_{\max}$.

Theorem 1. *The solution* (l_{ij}, e_{ij}) *of the basic formulation is feasible and maximizes the average accuracy.*

Proof. From Lemma 1 to Lemma 6 the solution (l_{ij}, e_{ij}) is feasible. Now consider any feasible solution (l^*_{ij}, e^*_{ij}) with average accuracy $\frac{1}{n} \sum_{T_i \in T} A_i \max_{f_a \in F} \{l^*_{ia}\}$. Then, since (l_{ij}, e_{ij}) is the feasible solution that maximizes Objective (1) it must hold that $\frac{1}{n} \sum_{T_i \in T} A_i \max_{f_a \in F} \{l_{ia}\} \geq \frac{1}{n} \sum_{T_i \in T} A_i \max_{f_a \in F} \{l^*_{ia}\}$. □

C. MILP Formulation

Observe that the basic formulation is not an MILP, because it has constraints with products of variables. We use standard transformations to obtain an MILP with the same solutions.

First, we change the objective to

$$\text{Maximize } \frac{1}{n} \sum_{T_i \in T} A_i \sum_{f_a \in F} l_{ia}. \tag{10}$$

Second, Constraint 4 can be reformulated as

$$M_{ia} \geq y_j + \sum_{f_b \in F} e_{jb} \cdot c^{ba}_{ji}, \forall T_i \in T, f_a \in F, T_j \in in_i \tag{11}$$

Finally, we add a variable $z_{ia} \in [0, t_{\max}]$ to replace $e_{ia} M_{ia}$ in Constraint 5, which can be reformulated as

$$y_i = \sum_{f_a \in F} e_{ia} M_{ia} + \sum_{f_a \in F} t_{ia} l_{ia} \tag{12}$$
$$= \sum_{f_a \in F} z_{ia} + \sum_{f_a \in F} t_{ia} l_{ia}, \quad \forall T_i \in T \tag{13}$$

To avoid a quadratic restriction $z_{ia} = e_{ia} M_{ia}$, we use instead

$$z_{ia} \leq t_{\max} \cdot e_{ia}, \quad \forall T_i \in T, f_a \in F \tag{14}$$
$$z_{ia} \leq M_{ia}, \quad \forall T_i \in T, f_a \in F \tag{15}$$
$$z_{ia} \geq M_{ia} - (1 - e_{ia}) t_{\max}, \quad \forall T_i \in T, f_a \in F \tag{16}$$

These changes produce the desired MILP formulation.

Theorem 2. *The MILP formulation has the same solution as the basic formulation.*

Proof. First, the change in the objective does not have an impact, since from Lemma 3 they are equivalent.

Second, the new version of Constraint 4 only affects Lemma 5 partially. It can be restated as "M_{ia} is *lower bounded by* the earliest time task T_i can start if assigned to fog f_a, and y_i is *lower bounded by* the time to approximate task T_i to level l_{ia} in the fog f_a to which it was assigned ($e_{ia} = 1$). The bounds can be made as tight as required." This does not affect the correctness of Lemma 6, since the bounds can be fully tight if required.

Finally, the changes in Constraint 5 do not change it as long as $z_{ia} = e_{ia} M_{ia}$. Since $e_{ia} \in \{0, 1\}$, we have to show that (a) $z_{ia} = 0$ if $e_{ia} = 0$ and (b) $z_{ia} = M_{ia}$ if $e_{ia} = 1$.

For Claim (a), when $e_{ia} = 0$ Constraint 14 becomes $z_{ia} \leq 0$, which combined with $z_{ia} \in [0, t_{\max}]$ yields $z_{ia} = 0$. Regarding Claim (b), when $e_{ia} = 1$ Constraint 16 becomes $z_{ia} \geq M_{ia}$, which combined with Constraint 15 yields $z_{ia} = M_{ia}$. □

VI. Task Assignment using Greedy Algorithm

We develop a Greedy DAG scheduler that allocates tasks in federated fog environments by prioritizing tasks based on their accuracy levels while managing task dependencies. It makes locally optimal decisions at each step to approximate an efficient overall schedule for resource allocation. The scheduling algorithm operates as follows, also given in Algorithm 1:

Algorithm 1 Greedy Task Scheduler Algorithm

$approx \leftarrow 1$
Sort the tasks in descending order of accuracy A_i
repeat
  Initialize $y_i \leftarrow \infty, \forall T_i \in T$
  while $\exists T_i \in T : y_i = \infty$ **do**
    Select first task T_i such that $y_i = \infty$ and $\forall T_j \in \text{in}_i, y_j \neq \infty$
    for each fog node $f_a \in F : P_{i,a} = 1$ **do**
      – Compute earliest start time
      $M_{ia} \leftarrow \max_{T_j \in \text{in}_i} \left(y_j + \sum_{f_b \in F} e_{jb} \cdot c_{ji}^{ba} \right)$
      – Compute completion time
      $y_i^* \leftarrow M_{ia} + t_{ia} \cdot approx$
      if $y_i^* < y_i$ **then**
        $y_i \leftarrow y_i^*$
        $e_{ia} \leftarrow 1$; $e_{ib} \leftarrow 0, \forall f_b \neq f_a$
      end if
    end for
  end while
  $maxy \leftarrow \max_{T_i \in T}(y_i)$
  if $maxy > t_{max}$ **then**
    – Unfeasible; reduce the average accuracy
    $approx \leftarrow approx \cdot \sigma - \rho$
  end if
until $maxy \leq t_{max}$

The greedy scheduling algorithm proceeds as follows: **Input Initialization** parses the DAG to extract tasks, dependencies, and associated accuracy levels. A **Topological Sort** initializes the ready queue with zero-indegree tasks to respect precedence constraints. In **Task Prioritization**, ready tasks are sorted in descending order of accuracy. During **Greedy Assignment**, each task is evaluated for earliest completion time across available fog nodes, and assigned to the node minimizing this time. Node availability is updated, and the task is marked complete. A **Dependency Update** decreases the indegree of successors, adding them to the queue when eligible. These steps **repeat** until all tasks are scheduled.

The Greedy Task Scheduler assigns tasks to fog nodes by minimizing individual completion times while considering dependencies, execution durations, and inter-node communication delays. Tasks are sorted by accuracy and scheduled iteratively on the fog node that offers the earliest finish time. If the total execution time exceeds the deadline $t_{\max}$, a scaling factor $\sigma = \frac{t_{\max}}{\max y}$ is applied, where $\max y$ denotes the current total completion time. All task durations are reduced proportionally by multiplying with σ, ensuring the workflow completes within the deadline. Additionally, a further reduction factor of, say, $\rho = 0.005$ can be applied, if necessary, to aggressively adjust task durations under strict time constraints.

The greedy approach focuses on local optimization, which may lead to suboptimal global schedules. In contrast, the MILP model jointly optimizes task assignments and resource utilization, providing better performance for complex dependencies and heterogeneous fog resources. The greedy method serves as a lightweight baseline suitable for dynamic scenarios.

VII. Experimental Setup

We developed a Python-based simulation framework to generate synthetic datasets representing the federated fog network and DAG workloads. Configurations, including task execution times, dependencies, deadlines, fog node capacities, and network latencies, were stored in JSON format.

Experiments evaluated the impact of deadline constraints (t_{max}), slack factors (α, see below), and fog node counts (N) on task accuracy, deadline adherence, and resource utilization. The simulation dynamically scheduled tasks across fog nodes, considering network delays and workload variations.

Workload Generation: DAG-based workloads simulated disaster response scenarios, with tasks assigned execution times from a normal distribution. Workloads varied by:
Evaluation settings: (1) DAG Size: Small (10 tasks), Medium (100), Large (1000); **(2) Dependency Levels**: Ranging from sparse to dense; **(3) Deadline Constraints**: Controlled using slack factor α; **(4) Task Heterogeneity**: Mix of compute-intensive and latency-sensitive tasks.

Federated Fog Simulation: A discrete-event simulator modeled variable number, $N = \{2, 4, 6, 8, 10\}$, of fog nodes with varied processing capacities, factors for network latencies such as bandwidth, distances, and congestion, and task offloading based on resource availability and scheduling policies.

A. Experimental Design

Experiments were conducted to evaluate the impact the following factors;

Experiment 1 – Effect of $t_{\max}$: We analyzed the impact of $t_{\max}$ on scheduling accuracy using a fixed fog network of 10 nodes. By varying $t_{\max}$, we examined its effect on task completion rates and evaluated whether increasing $t_{\max}$ improves deadline adherence and resource allocation efficiency.

Experiment 2 – Slack Factor (α) in $t_{\max}$: This experiment assessed how introducing slack into $t_{\max}$ affects execution performance. The threshold was adjusted as:

$$t_{\max} = \sum \text{Execution time of critical path} + \alpha \cdot \text{Slack time}$$

We tested strict deadlines ($\alpha = 1.0$), moderate slack ($\alpha = 1.2, 1.3$), and loose deadlines ($\alpha = 1.5$) to determine the optimal range balancing deadline adherence and resource efficiency.

Experiment 3 – Scaling Fog Network Size: A large DAG with 1000 tasks was scheduled across fog networks with $N = \{2, 4, 6, 8, 10\}$ nodes. This helped evaluate how increasing the number of fog nodes influences execution efficiency and deadline compliance for large workloads.

Experiment 4 – Baseline Performance: A baseline was established using an infinite $t_{\max}$, allowing all tasks to complete without deadline constraints. This run provided an upper bound on achievable scheduling accuracy. The same DAG dataset was used across all experiments to ensure consistency and comparability.

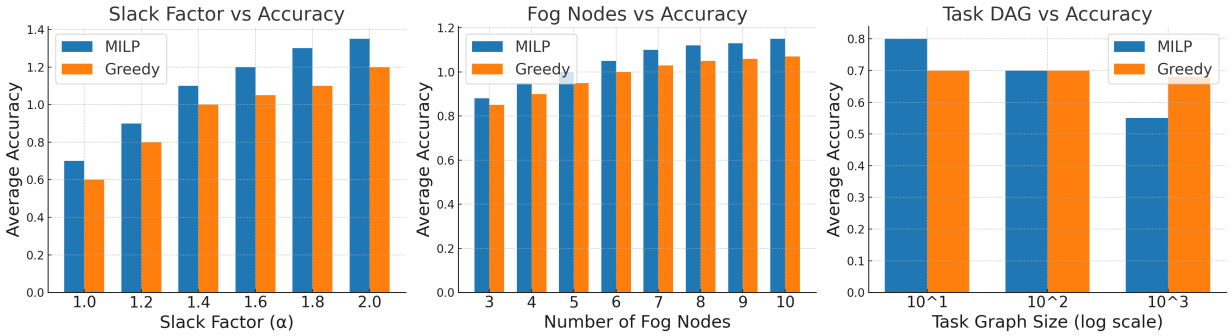

Fig. 3: Performance comparison of MILP and Greedy approaches across three configurations in terms of average accuracy: (a) MILP achieves higher accuracy than Greedy as slack factor increases; (b) MILP continues to outperform Greedy with increasing fog node count; (c) As task DAG size grows, MILP's accuracy degrades significantly, while Greedy also exhibits some level of decline.

B. Experimental Results

MILP and Greedy were evaluated in terms of: *average accuracy* metric, with respect to varying *slack factor* (deadline tightness), size of federated fog (*number of fogs in the federation*), and DAG size (*task graph size*), as shown in Fig. 3.

Slack Factor: we can see that increasing slack ($\alpha \in \{1.0, 1.2, 1.5, 2.0\}$) improves the average accuracy in both MILP and Greedy. However, MILP maintains a higher accuracy; Greedy improvement slows down beyond$\alpha = 1.5$.

Fog Nodes: As node count increases, MILP benefits consistently; However, Greedy plateaus beyond 8 nodes, highlighting its limited adaptability.

DAG Size: MILP performs well on small DAGs (10 tasks) but loses momentum for larger graphs. Greedy remains viable at scale (e.g., for 1000 tasks), despite offering lower accuracy. This prompt us that although **MILP** achieves optimal accuracy, but suffers from *exponential complexity*, limiting its scalability. **Greedy** *scales efficiently*, making it suitable for large or dynamic workloads, which shows a *trade-off* between accuracy and scalability.

VIII. Conclusion and Future Work

This work introduces MILP and Greedy-based DAG scheduling for federated fog systems. MILP offers high accuracy, but lacks scalability; Greedy enables fast, deadline-aware scheduling with lower accuracy. Runtime comparisons and real-world DAG evaluation (e.g. fire detection) demonstrate the performance *trade-offs*. Future work includes real-world deployment in offshore and industrial IoT environments and addressing MILP scalability using relaxation techniques and metaheuristic algorithms for efficient, near-optimal scheduling.

Acknowledgments

This work was supported by NSF IRES Track 1 grants 2417064 and 2246391, NSF award CCF-1956373, and project PID2022-140560OB-I00 (DRONAC), funded by MICIU/AEI/ 10.13039/501100011033 and the European Regional Development Fund (ERDF, EU). We thank the reviewers for their valuable feedback.

References

[1] A. M. Nemati and N. Mansouri, "Resource allocation in fog computing: A survey on current state and research challenges," *Knowledge and Information Systems*, vol. 67, pp. 2091–2170, 2025.

[2] X. Yang and N. Rahmani, "Task scheduling mechanisms in fog computing: Review, trends, and perspectives," *Kybernetes*, vol. 50, no. 1, pp. 22–38, 2021.

[3] A. Pakmehr, M. Gholipour, and E. Zeinali, "Energy–efficient and deadline–aware task scheduling in fog computing," *Sustainable Computing: Informatics and Systems*, vol. 43, p. 100988, 2024.

[4] R. Thakur, G. Sikka, U. Bansal, J. Giri, and S. Mallik, "Deadline-aware and energy efficient iot task scheduling using fuzzy logic in fog computing," *Multimedia Tools and Applications*, vol. 84, no. 14359, pp. 14359–14386, 2025.

[5] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 8, pp. 13849–13875, Sept. 2021.

[6] J. Kaur, A. Agrawal, and R. A. Khan, "Security issues in fog environment: A systematic literature review," *International Journal of Wireless Information Networks*, vol. 27, pp. 467–483, 2020.

[7] A. Aljumah, A. Kaur, M. Bhatia, and T. A. Ahanger, "Internet of things–fog computing–based framework for smart disaster management," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 8, p. e4078, 2021.

[8] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Resource provisioning in fog computing through deep reinforcement learning," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, (Bordeaux, France), pp. 431–437, IEEE, 2021.

[9] R. F. Hussain, A. Pakravan, and M. Amini Salehi, "Analyzing the performance of smart industry 4.0 applications on cloud computing systems," *HPCC*, pp. 11–18, 2020.

[10] R. F. Hussain and M. A. Salehi, "Resource allocation of industry 4.0 micro-service applications across serverless fog federation," *Future Generation Computer Systems*, vol. 154, pp. 479–490, 2024.

[11] R. F. Hussain, A. Mokhtari, A. Ghalambor, and M. Amini Salehi, *IoT for Smart Operations in the Oil and Gas Industry: From Upstream to Downstream*. Elsevier, 1st ed., 2022.

[12] L. Cai, X. Wei, C. Xing, X. Zou, G. Zhang, and X. Wang, "Failure-resilient dag task scheduling in edge computing," *Computer Networks*, vol. 198, p. 108361, 2021.

[13] J. Shi, J. Du, J. Wang, and J. Yuan, "Federated deep reinforcement learning-based task allocation in vehicular fog computing," in *2022 IEEE 95th Vehicular Technology Conference (VTC2022-Spring)*, (Helsinki, Finland), pp. 1–6, IEEE, 2022.

[14] M. Amini Salehi, J. Smith, A. A. Maciejewski, H. J. Siegel, E. K. P. Chong, J. Apodaca, L. D. Briceno, T. Renner, V. Shestak, and J. Ladd, "Stochastic-based robust dynamic resource allocation for independent tasks in a heterogeneous computing system," *Journal of Parallel and Distributed Computing*, vol. 97, pp. 96–111, 2016.

[15] M. Goudarzi, M. Palaniswami, and R. Buyya, "A distributed deep reinforcement learning technique for application placement in edge and fog computing environments," *IEEE Transactions on Mobile Computing*, vol. 22, no. 5, pp. 2491–2505, 2023.

[16] J. K. Lenstra, D. B. Shmoys, and É. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Mathematical Programming*, vol. 46, pp. 259–271, 1990.