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

Presented by: Gayani Gupta

Ph.D. Research Candidate, University of North Texas

Abstract – Research Summary

- **Problem Context**
  - Industry 4.0 applications in remote, latency-sensitive environments require DAG-based workflows and serverless edge processing
- **Key Challenge**
  - Standalone fog nodes lack elasticity, making them incapable of handling concurrent DAG workloads during critical scenarios
- **Proposed Approach**
  - Mixed-Integer Linear Programming (MILP) for optimal task allocation
  - Greedy heuristic for scalable, approximate scheduling in real-time
- **Research Contributions**
  - First comparative formulation and analysis of MILP vs. Greedy scheduling across federated fog computing infrastructures
- **Experimental Validation**
  - Demonstrates improved elasticity, deadline adherence, and accuracy-efficiency trade-offs on synthetic DAG workloads

Motivation and Use Case – Real-Time Fire Detection

- **Industry Context**
  - Industry 4.0 systems deployed in offshore oil fields, space stations, and disaster-prone zones require real-time, autonomous operation
- **Application Scenario**
  - Real-time fire detection and emergency response is a mission-critical workflow involving tightly coupled tasks
- **Workflow Structure**
  - Tasks include:
    - Sensor data collection
    - Smoke and heat analysis
    - Fire localization
    - Alert dissemination
  - Modeled as a **Directed Acyclic Graph (DAG)**, encoding task dependencies and execution order
- **Fog Deployment Model**
  - Tasks are distributed across a **federated fog network** (e.g., FG1–FG3)
  - Nodes are geographically dispersed, with heterogeneous compute and communication capacities
- **Key Challenge**
  - Ensure deadline-compliant, low-latency task execution despite resource variability and network delays

DAG-Based Fire Detection Workflow

- Real-time emergency response scenario modeled as a **DAG**
- Nodes represent microservices; edges define task dependencies
- Each task depends on sensor data and must execute within latency constraints to ensure timely disaster response.

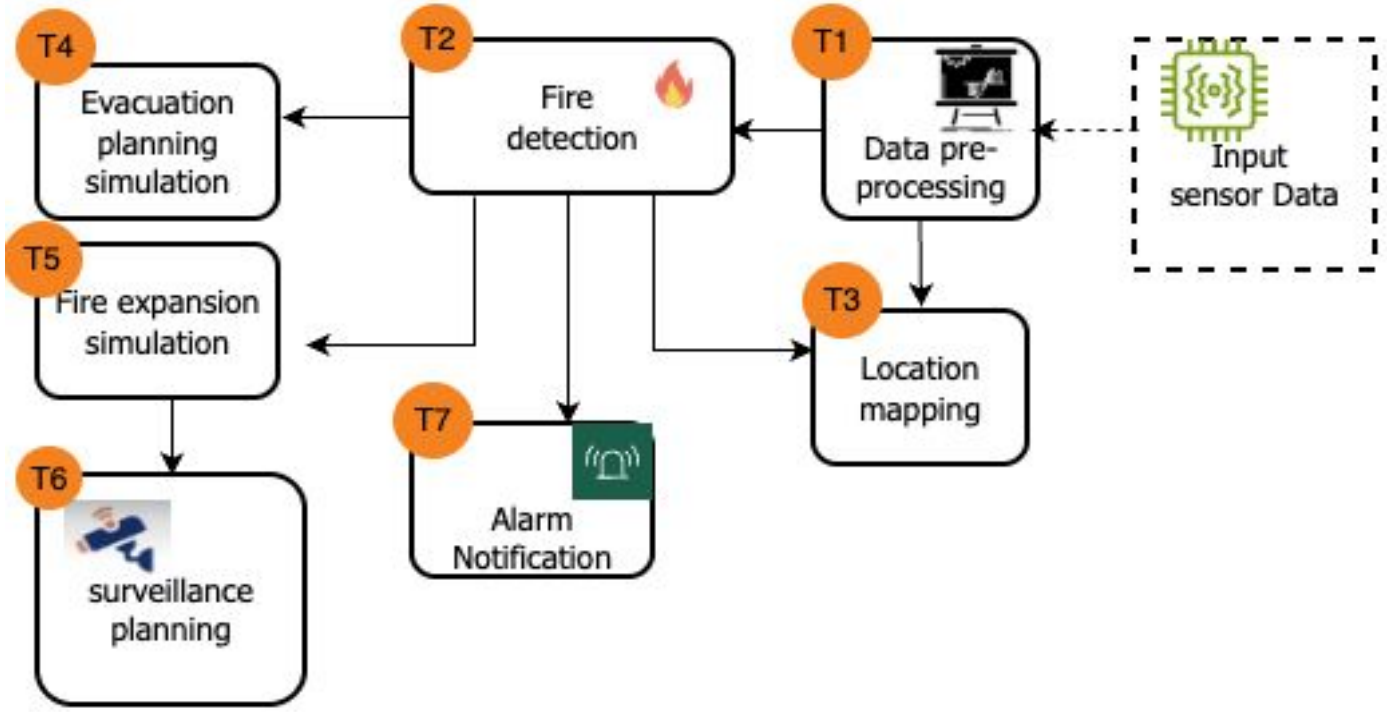


Fig. 1: A DAG-based fire detection workflow deployed across federated fog nodes.

Problem Statement and Modeling

❓ Research Question

- How can we enable efficient execution of DAG-based workloads on resource-constrained fog systems?

⚠️ Key Challenge

- Single fog nodes **lack computational elasticity** during critical events (e.g., oil spills, fires)
- Mission-critical DAGs involve **multiple dependent tasks** with tight deadlines
- Execution must satisfy **resource, timing, and dependency constraints**

Federated Fog

🌐 Motivation

- Coordinating across **multiple fog nodes** enables:
  - Resource sharing
  - Load balancing
  - Fault tolerance

👎 Drawbacks

- Introduces complexity in:
  - Task placement
  - Network-aware scheduling
  - Accuracy vs. performance trade-offs

🎯 Goal

- Allocate DAG tasks across fog nodes to:
  - Maximize average execution accuracy
  - Ensure all tasks finish by global deadline ***tmax***

Proposed Architecture

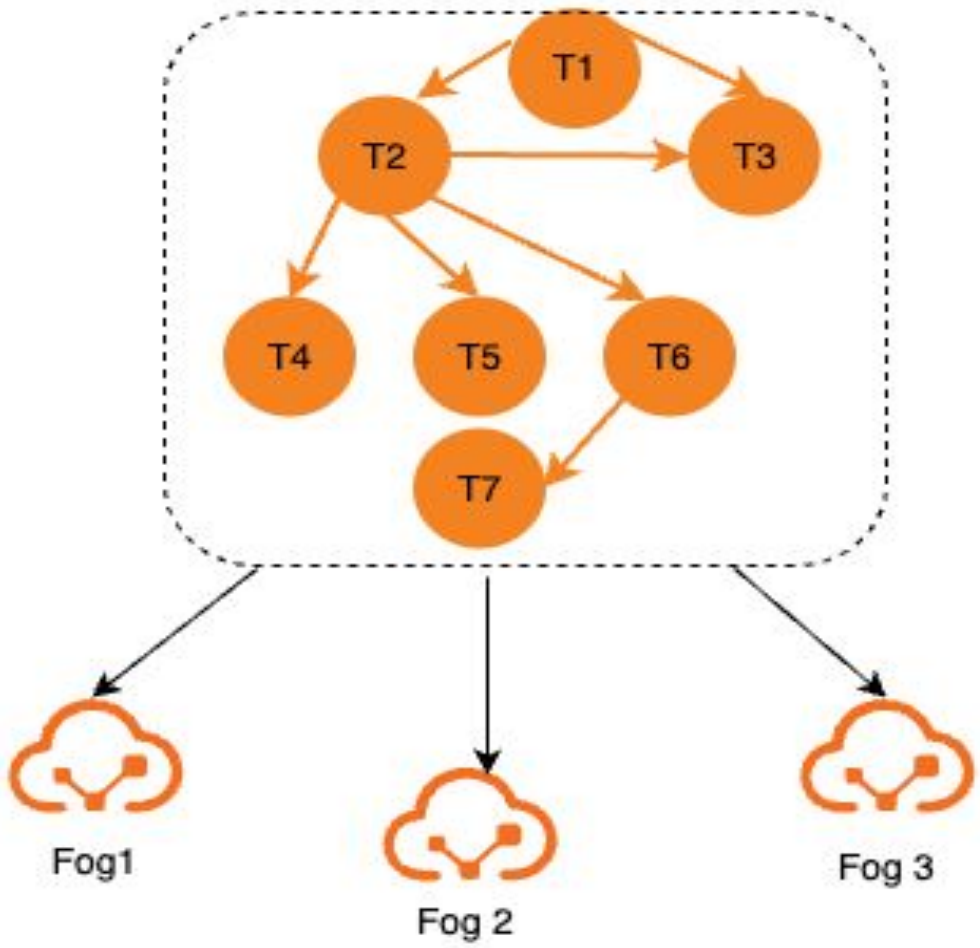


Fig. 2: Mapping of a fire detection workflow DAG onto a federated fog computing infrastructure.

Key Features

- **Federated fog nodes** collaborate to share compute resources
- **DAG tasks** are distributed across **heterogeneous fog nodes** under resource and connectivity constraints

Proposed Scheduling Algorithms

🧩 Problem Definition Recap

- **Objective**: Assign DAG tasks to fog nodes to:
  - Maximize **average task accuracy**
  - Ensure all tasks finish by **global deadline *tmax***

1. MILP-Based Optimal Scheduling

- Formulated as a **Mixed-Integer Linear Program**
- Captures dependencies, execution time, communication delays
- Guarantees **maximum achievable accuracy**
- **Drawback**: High computational cost; limited scalability

2. Greedy Scheduling Heuristic

- Fast, **adaptive approximation method**
- Prioritizes high-accuracy tasks and minimizes completion time
- Uses local decisions to iteratively schedule tasks
- **Trade-off**: Reduced optimality, but suitable for large-scale and real-time use

MILP Formulation for Optimal Tasks Assignment

We solve a dependent task allocation problem in a fog federation.

Let $T = \{T_1, T_2, \ldots, T_n\}$ be tasks and $F = \{f_1, f_2, \ldots, f_m\}$ be fog nodes.

Each task T_i must be assigned to one fog f_a such that:

- $P_{ia} \in \{0, 1\}$ indicates if T_i is allowed on f_a
- Full execution gives accuracy A_i, else accuracy decreases linearly
- Execution time is $t_{ia} \geq 0$

A DAG defines dependencies. If $T_j \rightarrow T_i$ and nodes differ:

$$T_i \text{ waits for communication delay } c_{ji}^{ba}$$

Objective: Complete all tasks by deadline $t_{\max}$ while maximizing average accuracy.

Assumptions:

- Tasks are non-preemptive, executed fully/partially on one fog
- No interference due to space-sharing
- Accuracy is averaged across tasks

MILP Formulation: Objective and Constraints

Variables:

- $l_{ia} \in [0, 1]$: Fraction of task T_i on fog f_a
- $e_{ia} \in \{0, 1\}$: Binary task-to-fog assignment
- y_i: Completion time of task T_j
- M_{ia}: Earliest start time of T_i on f_a

Objective (MILP-rewritten):

$$\max \frac{1}{n} \sum_{T_i \in T} A_i \sum_{f_a \in F} l_{ia}$$

Constraints:

$$e_{ia} \geq l_{ia}$$

$$\sum_{f_a \in F} e_{ia} = 1$$

$$M_{ia} \geq y_j + \sum_{f_b \in F} e_{jb} \cdot c_{ji}^{ba} \quad \forall T_j \in \text{in}_i$$

$$y_i = \sum_{f_a \in F} (z_{ia} + t_{ia} \cdot l_{ia})$$

$$y_i \leq t_{\max}$$

$$e_{ia} \leq P_{ia}$$

Auxiliary Variables (to linearize $z_{ia} = e_{ia} M_{ia}$):

$$z_{ia} \leq t_{\max} \cdot e_{ia}$$

$$z_{ia} \leq M_{ia}$$

$$z_{ia} \geq M_{ia} - (1 - e_{ia}) \cdot t_{\max}$$

Greedy Scheduling Heuristic – Motivation

❓ Why Greedy

- MILP gives optimal task assignments but is computationally expensive for large DAGs.
- Real-time scenarios (e.g., fire detection, oil spill mitigation) demand faster decision-making.
- Need a lightweight and adaptive strategy that maintains reasonable accuracy under strict deadlines.

Key Idea

- Assign tasks to fog nodes in priority order based on execution time, communication latency, and estimated impact on accuracy.

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 $maxy \leq t_{max}$

Experimental Setup – Simulation Framework

- **Framework**
  - Python-based simulator for federated fog and DAG workloads
  - Configuration stored in JSON:
    - Task times, deadlines, dependencies
    - Fog node capacity, network latencies
- **Scheduling Objective**
  - Evaluate MILP vs. Greedy task allocation strategies

Workload Generation – DAG Design

❖ Workload Characteristics

- **Task Counts:** Small (10), Medium (100), Large (1000)
- **Task Types:** Mix of compute-intensive and latency-sensitive
- **Execution Times:** Sampled from Gaussian distributions
- **Accuracy:** Each task has a full-execution accuracy score AiA_iAi, reduced when partial

❖ Dependency Structures:

- DAG edges represent data dependencies
- Variability: Sparse DAGs (shallow) and Dense DAGs (deep/complex)

Federated Fog Simulation

- **Network Parameters**
  - **Fog Nodes:** Simulated as a set F={f1,f2,...,fN}
  - **Node Heterogeneity:** Processing speed and available cores differ
  - **Network Latency:** Modeled by bandwidth, congestion, and distance
  - **Node Counts Tested:** N={2,4,6,8,10}
- **Offloading Policy**
  - Tasks assigned based on earliest available completion
  - Communication delay included for inter-node data transfer

Experimental Design – Four Key Experiments

Experiment 1: Impact of Deadline tmax

- Vary deadline duration to study how strictness affects scheduling success
- **Analyzed**: task completion rate, system idle time

Experiment 2: Slack Factor α\alphaα

- Defined as: tmax = Critical Path Time + α · Slack Time
- Tested α = {1.0,1.2,1.3,1.5}

Experiment 3: Scaling Fog Nodes

- Large DAGs run on varying number of nodes
- Measures scalability and load balancing

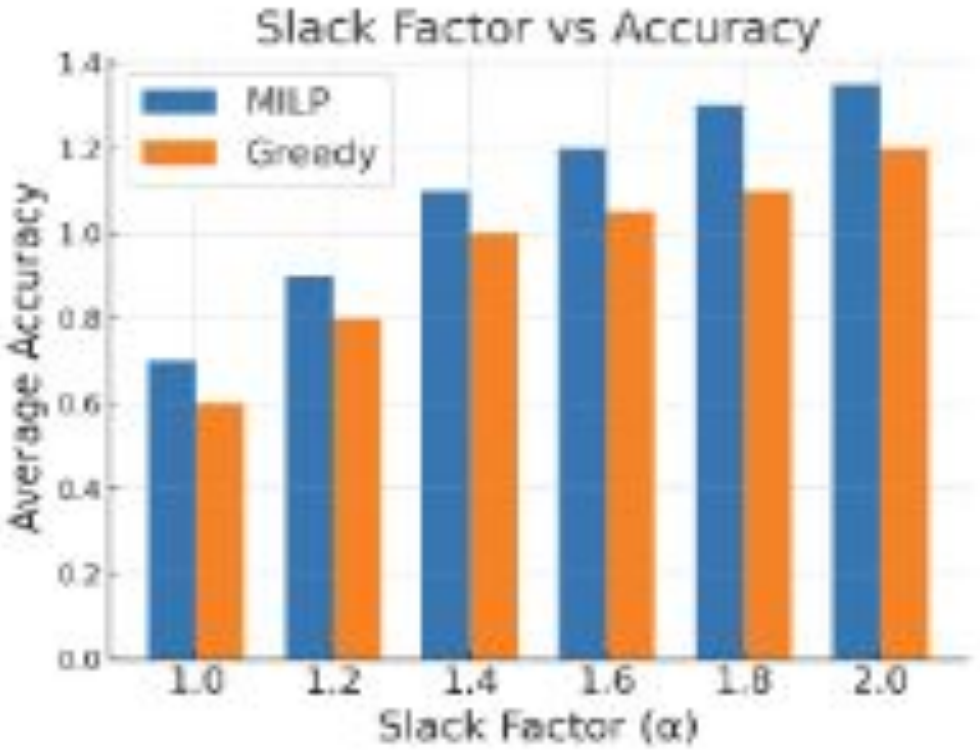
Experiment 4: Baseline Accuracy

- Used infinite deadline to establish upper bounds on accuracy
  Comparison baseline for MILP and Greedy

Performance Evaluation - Slack Factor (α)

Observations

- Accuracy improves with higher slack
- **MILP** outperforms **Greedy** consistently
- **Greedy** saturates at α≥1.5 showing diminishing returns



Interpretation

- Slack helps mitigate network and scheduling bottlenecks, but only to a point.
- MILP can leverage slack better due to its global optimization model.

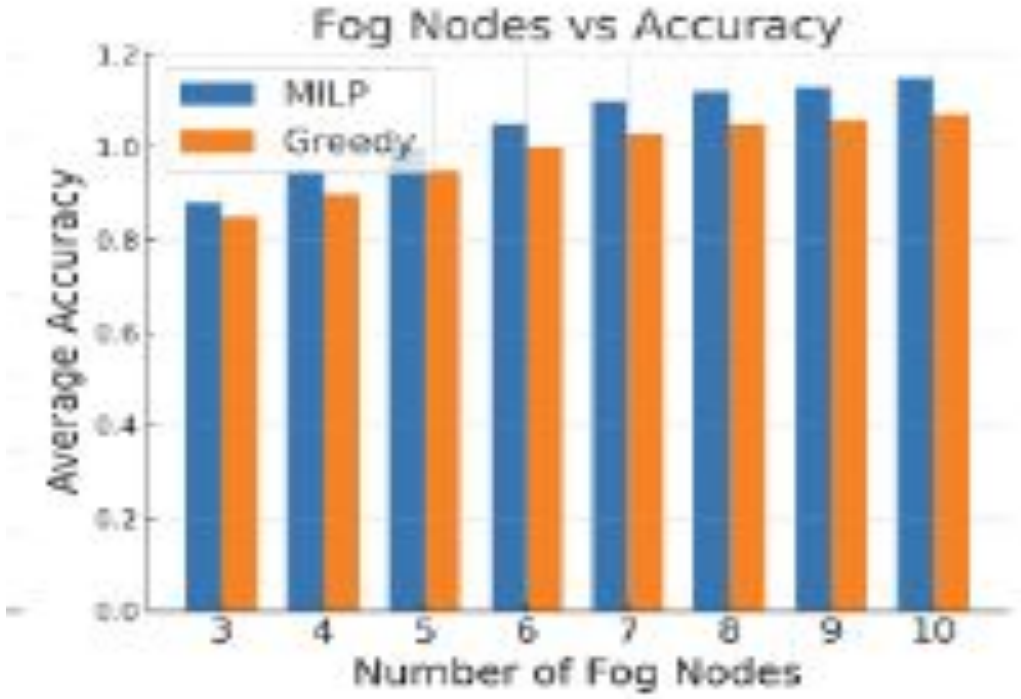
Performance Evaluation - Fog Node Scaling

Observations

- More fog nodes → higher parallelism
- **MILP** continues to improve
- **Greedy** plateaus after 8 nodes (limited ability to exploit extra nodes)

Interpretation

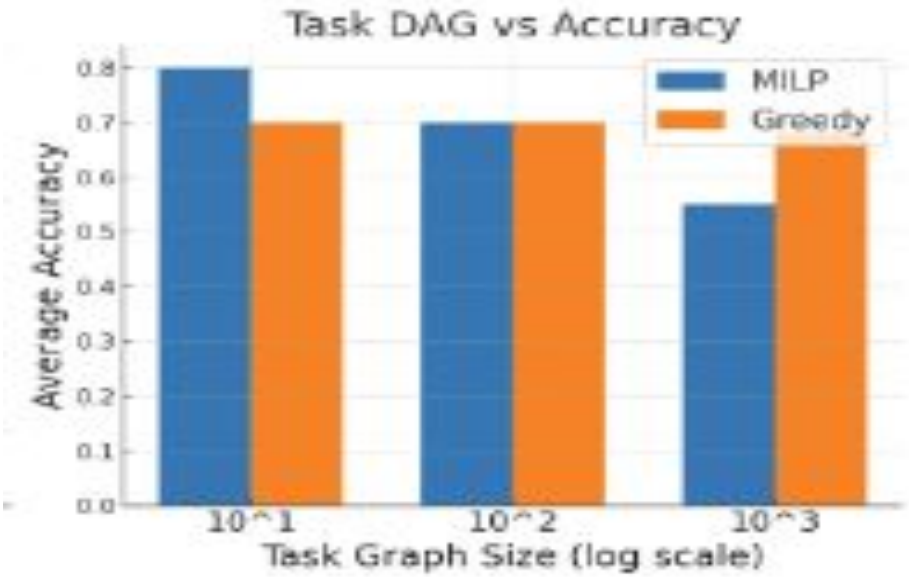
- Greedy operates locally per task
- MILP accounts for the entire workflow structure, adapting more flexibly to additional resources.



Performance Evaluation - DAG Size Impact

Observations

- **Small DAGs (10 tasks):** MILP is highly accurate
- **Medium DAGs (100 tasks):** MILP still outperforms but slower
- **Large DAGs (1000 tasks):** Greedy becomes competitive due to MILP's exponential complexity



Interpretation

- MILP is suitable for static, predictable workloads.
- Greedy excels when task graphs are large or generated on-the-fly.

Key Findings Summary

- **MILP**
  - **Pros:** High accuracy, optimality
  - **Cons:** Intractable for large DAGs
- **Greedy**
  - **Pros:** Fast, scalable
  - **Cons:** Suboptimal in accuracy
- **Overall Insight**
  - There is a clear **accuracy vs. scalability trade-off**, highlighting the need for hybrid or adaptive strategies.

Future Work

- **Scalability Improvements:** MILP relaxation, decomposition
- **Metaheuristics:** Genetic algorithms, simulated annealing
- **Real Deployments:** Offshore fog systems, industrial IoT pipelines
- **Resilience Considerations:** Task failure, node drop, load spikes