Spatiotemporal Non-Uniformity-Aware Online Task Scheduling in Collaborative Edge Computing for IIoT

Team: Thing Different - 02

Authors: Yang Li, Xing Zhang, Yukun Sun, Wenbo Wang, Bo Lei

IEEE Transactions on Mobile Computing

CB.SC.U4CSE23105 - Amrith
CB.SC.U4CSE23142 - S Ajeth
CB.SC.U4CSE23145 - Sarvesha
CB.SC.U4CSE23157 - Varun

Problem Understanding

Imagine an oil factory with multiple units, each having multiple pipelines and an edge server. Normally, sensor readings are taken every 10 seconds. However, if an anomaly is detected, the frequency of data collection from a specific pipeline is dramatically increased.

- **Temporal Non-Uniformity:** A sudden, high-frequency data burst from one unit (change over time).
- **Spatial Non-Uniformity:** An overloaded edge server in the leaking unit, while others are underutilized (uneven distribution across locations).

Why it is an Edge Problem

Starting with the Cloud

- **The Initial Thought:** The cloud seems like an obvious solution due to its vast, seemingly infinite resources for computation and storage.
- **Benefits:** It's great for tasks that aren't time-sensitive, like long-term data analysis, training large AI models, and acting as a robust backup for the entire system.

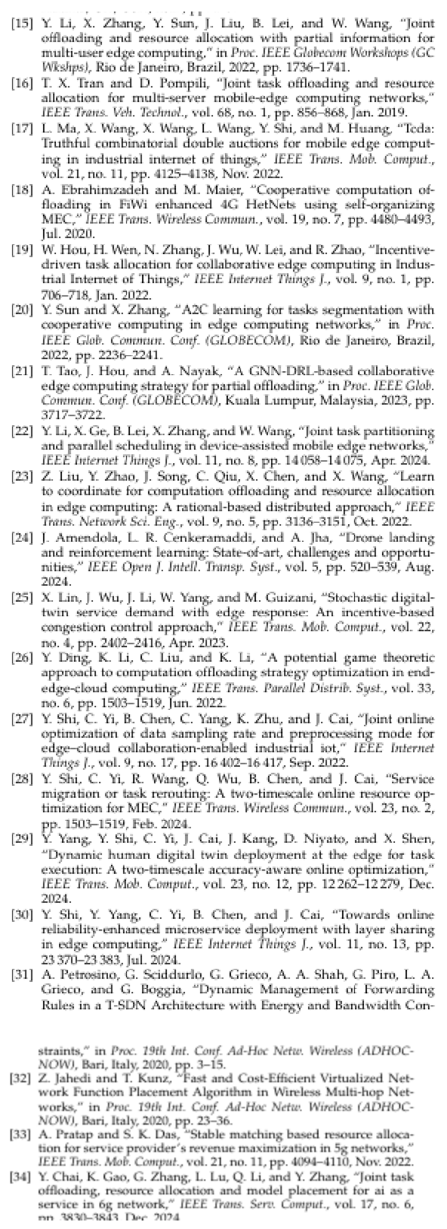
Why Cloud Isn't Feasible for Real-Time Tasks

- **High Latency:** For critical, real-time tasks like anomaly detection, the delay in sending data to a distant cloud server is simply too long and could lead to significant operational failures.
- **Privacy & Security:** Many industrial applications have strict regulations that prevent sensitive data from leaving the local factory network.
- **Cost & Bandwidth:** Continuously sending the massive volume of data generated by IIoT devices to the cloud can be very expensive and strain network bandwidth.

Why Edge Is the Perfect Solution

- **Low Latency:** Edge servers are located on-site, allowing for near-instantaneous data processing. This enables real-time decisions and automated responses critical for safety and efficiency.
- **Data Security:** Data processing remains within the private, local network, satisfying all privacy and security requirements.
- **Cost-Effective:** Edge computing handles the bulk of the data locally, reducing the need for costly and bandwidth-intensive data transfers to the cloud.

State of Art Literature Survey



[15] Y. Li, X. Zhang, Y. Sun, J. Liu, B. Lei, and W. Wang, “Joint offloading and resource allocation with partial information for multi-user edge computing,” in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Rio de Janeiro, Brazil, 2022, pp. 1736–1741.

[16] T. X. Tran and D. Pompili, “Joint task offloading and resource allocation for multi-server mobile-edge computing networks,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[17] L. Ma, X. Wang, X. Wang, L. Wang, Y. Shi, and M. Huang, “Tcda: Truthful combinatorial double auctions for mobile edge computing in industrial internet of things,” *IEEE Trans. Mob. Comput.*, vol. 21, no. 11, pp. 4125–4138, Nov. 2022.

[18] A. Ebrahimzadeh and M. Maier, “Cooperative computation offloading in FiWi enhanced 4G HetNets using self-organizing MEC,” *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4480–4493, Jul. 2020.

[19] W. Hou, H. Wen, N. Zhang, J. Wu, W. Lei, and R. Zhao, “Incentive-driven task allocation for collaborative edge computing in Industrial Internet of Things,” *IEEE Internet Things J.*, vol. 9, no. 1, pp. 706–718, Jan. 2022.

[20] Y. Sun and X. Zhang, “A2C learning for tasks segmentation with cooperative computing in edge computing networks,” in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Rio de Janeiro, Brazil, 2022, pp. 2236–2241.

[21] T. Tao, J. Hou, and A. Nayak, “A GNN-DRL-based collaborative edge computing strategy for partial offloading,” in *Proc. IEEE Glob. Commun. Conf. (GLOBECOM)*, Kuala Lumpur, Malaysia, 2023, pp. 3717–3722.

[22] Y. Li, X. Ge, B. Lei, X. Zhang, and W. Wang, “Joint task partitioning and parallel scheduling in device-assisted mobile edge networks,” *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14 058–14 075, Apr. 2024.

[23] Z. Liu, Y. Zhao, J. Song, C. Qiu, X. Chen, and X. Wang, “Learn to coordinate for computation offloading and resource allocation in edge computing: A rational-based distributed approach,” *IEEE Trans. Network Sci. Eng.*, vol. 9, no. 5, pp. 3136–3151, Oct. 2022.

[24] J. Amendola, L. R. Cenkeramaddi, and A. Jha, “Drone landing and reinforcement learning: State-of-art, challenges and opportunities,” *IEEE Open J. Intell. Transp. Syst.*, vol. 5, pp. 520–539, Aug. 2024.

[25] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani, “Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach,” *IEEE Trans. Mob. Comput.*, vol. 22, no. 4, pp. 2402–2416, Apr. 2023.

[26] Y. Ding, K. Li, C. Liu, and K. Li, “A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[27] Y. Shi, C. Yi, B. Chen, C. Yang, K. Zhu, and J. Cai, “Joint online optimization of data sampling rate and preprocessing mode for edge–cloud collaboration-enabled industrial iot,” *IEEE Internet Things J.*, vol. 9, no. 17, pp. 16 402–16 417, Sep. 2022.

[28] Y. Shi, C. Yi, R. Wang, Q. Wu, B. Chen, and J. Cai, “Service migration or task rerouting: A two-timescale online resource optimization for MEC,” *IEEE Trans. Wireless Commun.*, vol. 23, no. 2, pp. 1503–1519, Feb. 2024.

[29] Y. Yang, Y. Shi, C. Yi, J. Cai, J. Kang, D. Niyato, and X. Shen, “Dynamic human digital twin deployment at the edge for task execution: A two-timescale accuracy-aware online optimization,” *IEEE Trans. Mob. Comput.*, vol. 23, no. 12, pp. 12 262–12 279, Dec. 2024.

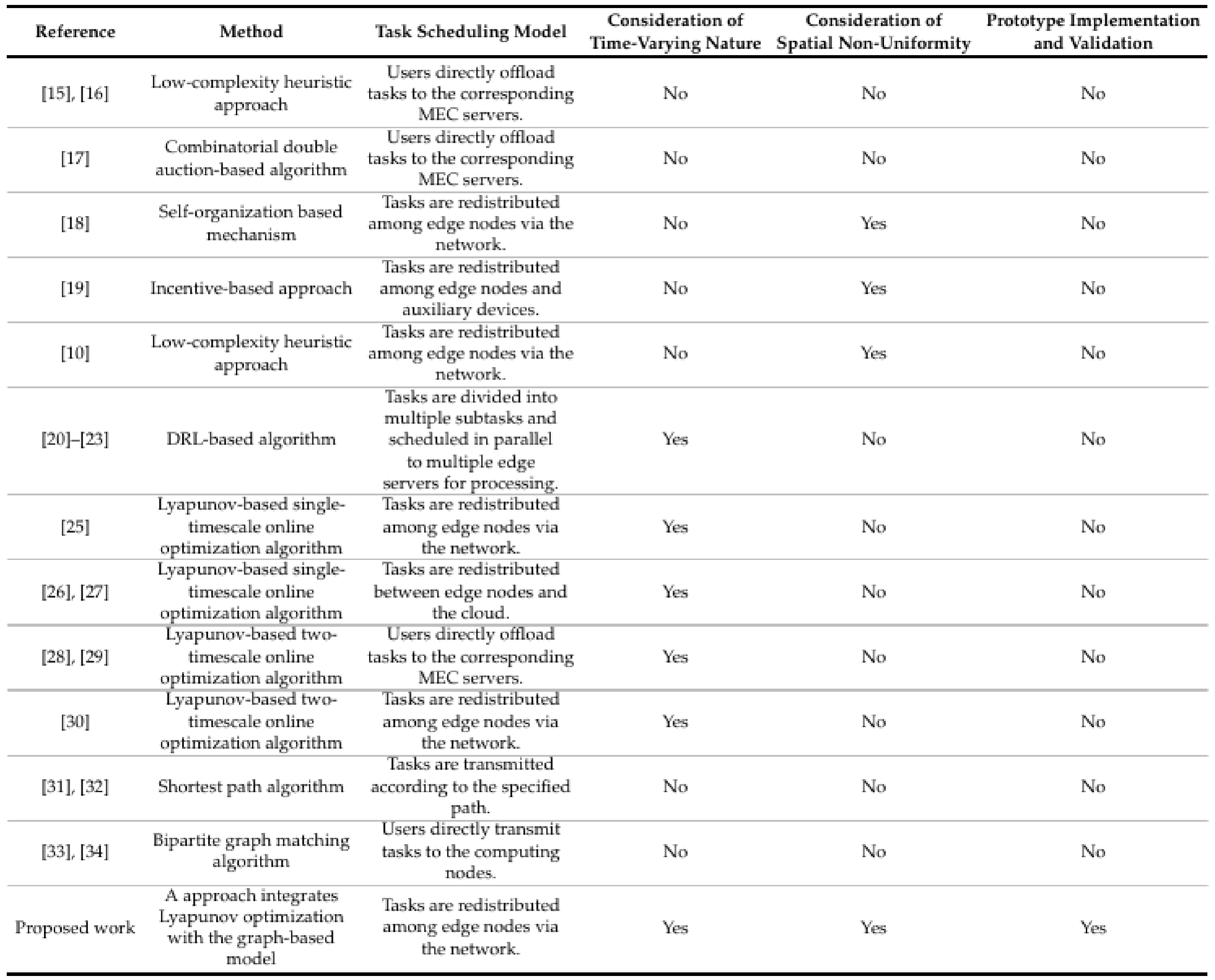
[30] Y. Shi, Y. Yang, C. Yi, B. Chen, and J. Cai, “Towards online reliability-enhanced microservice deployment with layer sharing in edge computing,” *IEEE Internet Things J.*, vol. 11, no. 13, pp. 23 370–23 383, Jul. 2024.

[31] A. Petrosino, G. Sciddurlo, G. Grieco, A. A. Shah, G. Piro, L. A. Grieco, and G. Boggia, “Dynamic Management of Forwarding Rules in a T-SDN Architecture with Energy and Bandwidth Constraints,” in *Proc. 19th Int. Conf. Ad-Hoc Netw. Wireless (ADHOC-NOW)*, Bari, Italy, 2020, pp. 3–15.

[32] Z. Jahedi and T. Kunz, “Fast and Cost-Efficient Virtualized Network Function Placement Algorithm in Wireless Multi-hop Networks,” in *Proc. 19th Int. Conf. Ad-Hoc Netw. Wireless (ADHOC-NOW)*, Bari, Italy, 2020, pp. 23–36.

[33] A. Pratap and S. K. Das, “Stable matching based resource allocation for service provider’s revenue maximization in 5g networks,” *IEEE Trans. Mob. Comput.*, vol. 21, no. 11, pp. 4094–4110, Nov. 2022.

[34] Y. Chai, K. Gao, G. Zhang, L. Lu, Q. Li, and Y. Zhang, “Joint task offloading, resource allocation and model placement for ai as a service in 6g network,” *IEEE Trans. Serv. Comput.*, vol. 17, no. 6, pp. 3830–3843, Dec. 2024.



Reference	Method	Task Scheduling Model	Consideration of Time-Varying Nature	Consideration of Spatial Non-Uniformity	Prototype Implementation and Validation
[15], [16]	Low-complexity heuristic approach	Users directly offload tasks to the corresponding MEC servers.	No	No	No
[17]	Combinatorial double auction-based algorithm	Users directly offload tasks to the corresponding MEC servers.	No	No	No
[18]	Self-organization based mechanism	Tasks are redistributed among edge nodes via the network.	No	Yes	No
[19]	Incentive-based approach	Tasks are redistributed among edge nodes and auxiliary devices.	No	Yes	No
[10]	Low-complexity heuristic approach	Tasks are redistributed among edge nodes via the network.	No	Yes	No
[20]–[23]	DRL-based algorithm	Tasks are divided into multiple subtasks and scheduled in parallel to multiple edge servers for processing.	Yes	No	No
[25]	Lyapunov-based single-timescale online optimization algorithm	Tasks are redistributed among edge nodes via the network.	Yes	No	No
[26], [27]	Lyapunov-based single-timescale online optimization algorithm	Tasks are redistributed between edge nodes and the cloud.	Yes	No	No
[28], [29]	Lyapunov-based two-timescale online optimization algorithm	Users directly offload tasks to the corresponding MEC servers.	Yes	No	No
[30]	Lyapunov-based two-timescale online optimization algorithm	Tasks are redistributed among edge nodes via the network.	Yes	No	No
[31], [32]	Shortest path algorithm	Tasks are transmitted according to the specified path.	No	No	No
[33], [34]	Bipartite graph matching algorithm	Users directly transmit tasks to the computing nodes.	No	No	No
Proposed work	A approach integrates Lyapunov optimization with the graph-based model	Tasks are redistributed among edge nodes via the network.	Yes	Yes	Yes

State of the Art Literature Survey Contd.

Single-Node Focus: Many existing studies assume end devices can offload tasks directly to all edge nodes, neglecting the reality that tasks must be redistributed via the network between collaborating edge nodes. - [15][16][17]

Short-Term Performance: Research on collaborative offloading often focuses on short-term gains, ignoring the long-term effects of time-varying request distributions on overall costs and performance. - [18][19]

DRL Drawbacks: While Deep Reinforcement Learning (DRL) is a popular approach, it suffers from significant issues for critical IIoT applications. Its training process is prone to instability and oscillations, and as a "black box" model, it lacks strict performance guarantees. -[20][21][23]

Lyapunov Limitations: Existing Lyapunov-based methods primarily focus on either spatial or temporal respectively and cannot be directly applied to problems with the coupled spatial and temporal dimensions present in IIoT.- [25][26][27]

Computational Cost: Other approaches, particularly those using graph models, often suffer from high computational costs that make them impractical for large-scale, real-time IIoT environments. - [31][33]

Lack of Real-World Validation: A critical gap is the absence of demonstrated superior performance in real engineering applications, as many solutions are difficult to implement in practice.

Architecture Overview

1. User Plane: IIoT Devices

What it is: Data sources like sensors, robots, and smart machines in factories.

What it does: Generate real-time data and send service requests to the nearest Access Point (AP).

2. Edge Computing Plane: Local Network

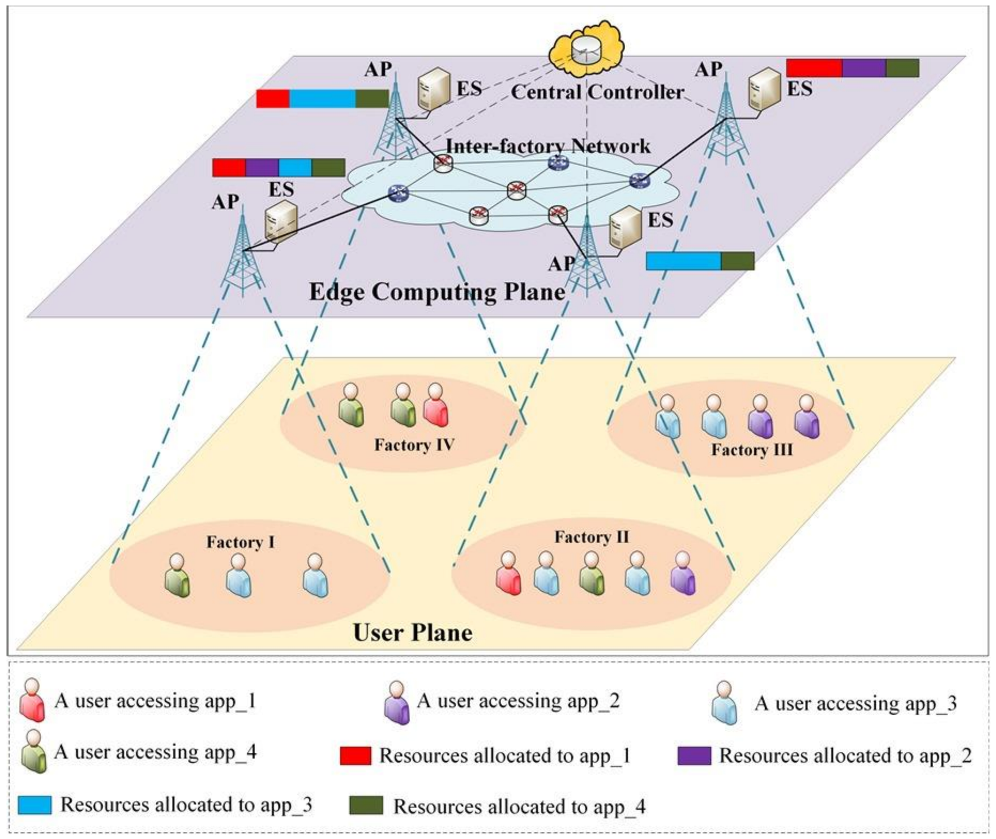
What it is: Each factory's AP and Edge Server (ES), connected to other factories.

What it does: AP handles requests; With the help of Central Controller, decision is made to process the request in the same ES or offload it elsewhere

3. Central Controller: The Brain

What it is: The system's decision-making core.

What it does: Monitors system status and adjusts task scheduling in real time, instructing APs where to route tasks – separating resource allocation from scheduling.



Contribution of this paper

1. Problem Formulation & Online Optimization

- **Dynamic Task Scheduling:** The paper formulates the dynamic task scheduling in collaborative edge computing as a **stochastic optimization problem**.
- **Long-Term Goals:** It aims to optimize **long-term average task processing delay** under a constraint of **long-term operational cost**.
- **Online Decision-Making:** It employs **Lyapunov optimization** to make real-time scheduling decisions without needing prior knowledge of future system states.

2. Novel Solution Approach

- **Graph Model Integration:** A graph model is introduced to guide optimal task scheduling decisions, especially for the NP-hard subproblems derived from Lyapunov optimization.
- **Two-Stage Heuristic Algorithm:** A two-stage heuristic algorithm is developed to find near-optimal solutions efficiently.
- **Imitation Learning for Acceleration:** An imitation learning-based scheme is developed to further reduce the algorithm's execution time, enhancing scalability.

The Role of Lyapunov Optimization:

Purpose: A mathematical tool for solving stochastic optimization problems in unpredictable systems[1]. It helps achieve long-term goals, like adhering to a budget, by guiding real-time, short-term decisions, such as minimizing delay.
Mechanism: It transforms a long-term problem into a series of real-time, single-slot problems. It does this by creating a **"virtual queue"** that tracks the long-term cost constraint. The algorithm's job is to make decisions in each time slot to keep this queue stable.

Why it fits IIoT: The IIoT environment is highly dynamic and stochastic, making traditional methods that require forecasting future events impractical. Lyapunov optimization is proactive and adaptive, making it a crucial capability for real-time industrial applications.

Workflow (per time slot):
•**Observe:** It senses the current system state, including new requests and the virtual queue backlog.
•**Solve:** It finds the best scheduling decision for that specific time slot by solving the real-time sub-problem.
•**Update:** It updates the virtual queue based on the cost of the decision.
•**Advance:** The process repeats for the next time slot.

Heuristic-Based Hierarchical Optimization

General Use

•**What it is:** A strategy that breaks down a complex, difficult problem into smaller, more manageable sub-problems to find a quick, effective solution.

•**Why it's used:** It's a pragmatic way to solve **NP-hard problems**, which are too slow to solve perfectly.

Specific Use in this Research

•**How it's used here:** A two-step process to solve the NP-hard task scheduling problem:

- **Step 1:** A **discrete particle swarm algorithm (PSO)** classifies factories as a **source**, **sink**, or **isolated node**. This dramatically simplifies the problem.
- **Step 2:** A **harmony search (HS)** algorithm then determines the specific number of tasks to be forwarded between the classified nodes.

•**The result:** This approach finds an efficient, near-optimal scheduling decision in real time, making the problem solvable for practical IIoT deployments.

Low-Complexity Acceleration with Imitation Learning

General Use
What it is: A machine learning strategy where an agent learns to perform a task by **mimicking an expert's behavior**, rather than learning through trial and error.
Why it's used: It is highly effective for reducing **computational complexity** and improving **scalability**, especially in systems where real-time decisions are critical and a slower, more complex "expert" algorithm already exists.

Specific Use in this Research
How it's used here: The project's more complex **Heuristic-Based Hierarchical (HH)** algorithm acts as the "expert." An **Imitation Learning (IL)** model, built with a **Graph Neural Network (GNN)**, is trained to mimic the HH algorithm's high-level category assignment decisions.
The result: This allows the system to make decisions with a significant **reduction in runtime**, making the solution practical and scalable for large-scale IIoT deployments where low-latency responsiveness is a priority.

Implementation strategy

1. Environment Setup:
Setup the necessary python environment with PyTorch

2. Scenario Formulation:
Simulations to be formulated to test the system's behavior, including:

- **Trade-off Analysis:** We will vary the **V** parameter(prioritization between delay and cost) to demonstrate the balance between delay and cost.
- **Scalability Test:** Increase the number of factories to validate the scalability and runtime efficiency of both the **HH** and **IL** models, and to show where the IL model becomes more practical.
- **Comparative Analysis:** Run various benchmark algorithms to compare their performance against both the **HH** and **IL** models on metrics like delay, cost, and runtime.

3. Simulation & Validation :
The simulation will use the pre-trained models to execute the formulated scenarios.

•**Codebase:** GitHub repository provided in the paper

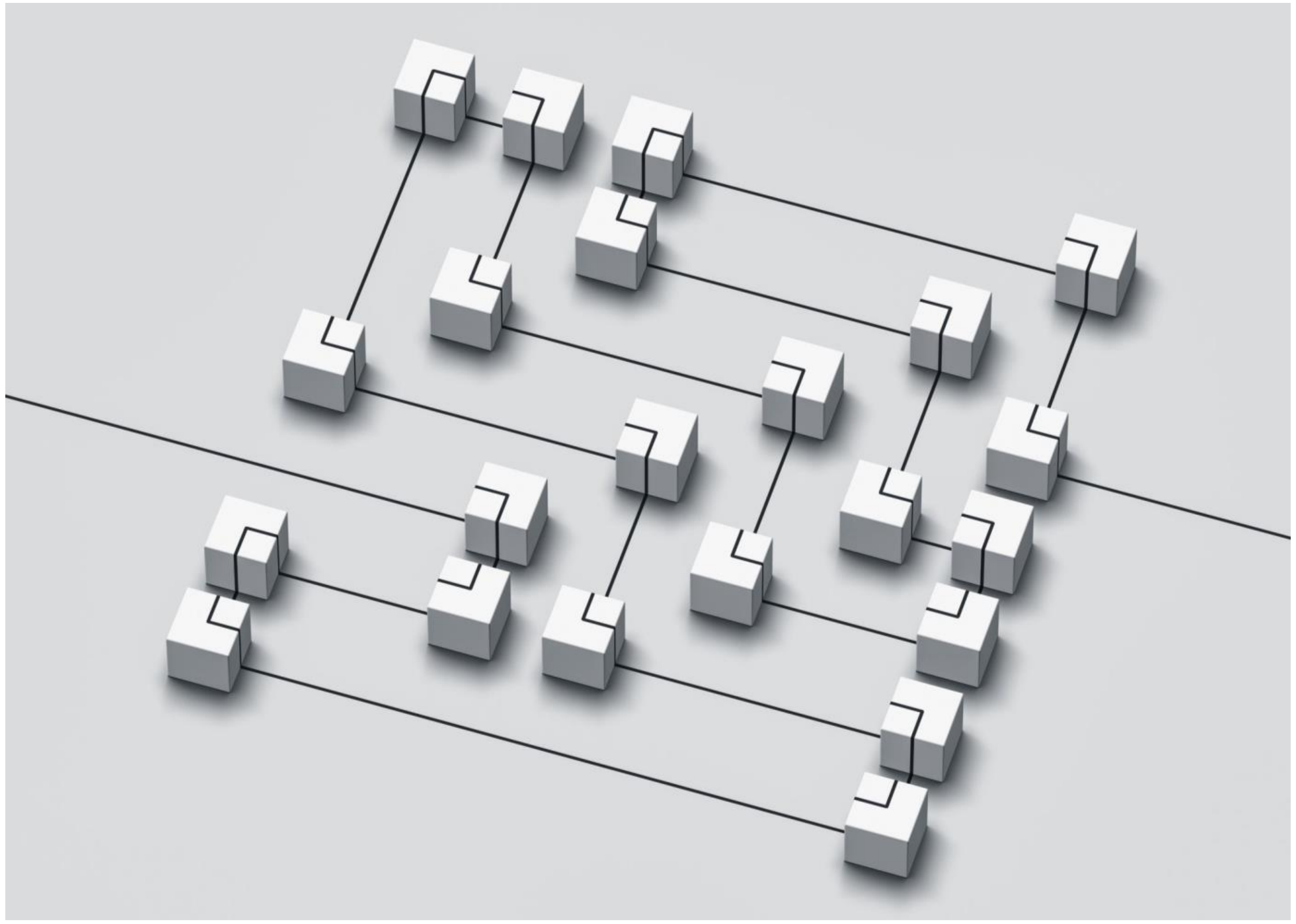
•**Programming Language:** Python

•**Core Libraries:** PyTorch

Simulation Environment: PyTorch

Performance Metrics:
•**Average Task Processing Delay(ms):** Measures the quality of service by calculating the average time from task arrival to completion.
•**Long-Term Average Task Processing Cost(J):** Measures the total operational cost over time to ensure adherence to the budget.
•**Computational Running Time(ms):** Evaluates the scalability and real-time feasibility of the algorithms as the network size changes.
•**Virtual Queue Backlog(J):** Used to demonstrate the algorithm's stability and its ability to effectively control the long-term cost constraint.
•**Weighted Sum of Normalized Delay and Overrun Cost:** A combined metric that shows the tunable balance between performance and cost.

Phase two Plan



- **Amrith**: Exploration of different practical use case of this paper, & study Heuristic based Search Algorithm
- **Varun**: study of Lyapunov Optimization Algorithm
- **Ajeth**: Simulator Setup
- **Sarvesha**: Experiment Design and Analysis