Multitask Scheduling of Computer Vision Workloads on Edge Graphical Processing Units

Arani Bhattacharya, **Paritosh Shukla**, Ansuman Banerjee, Saumya Jaipuria, Nanjangud Narendra, Dhruv Sekhar Garg

Indraprastha Institute of Information Technology Delhi

Indian Statistical Institute

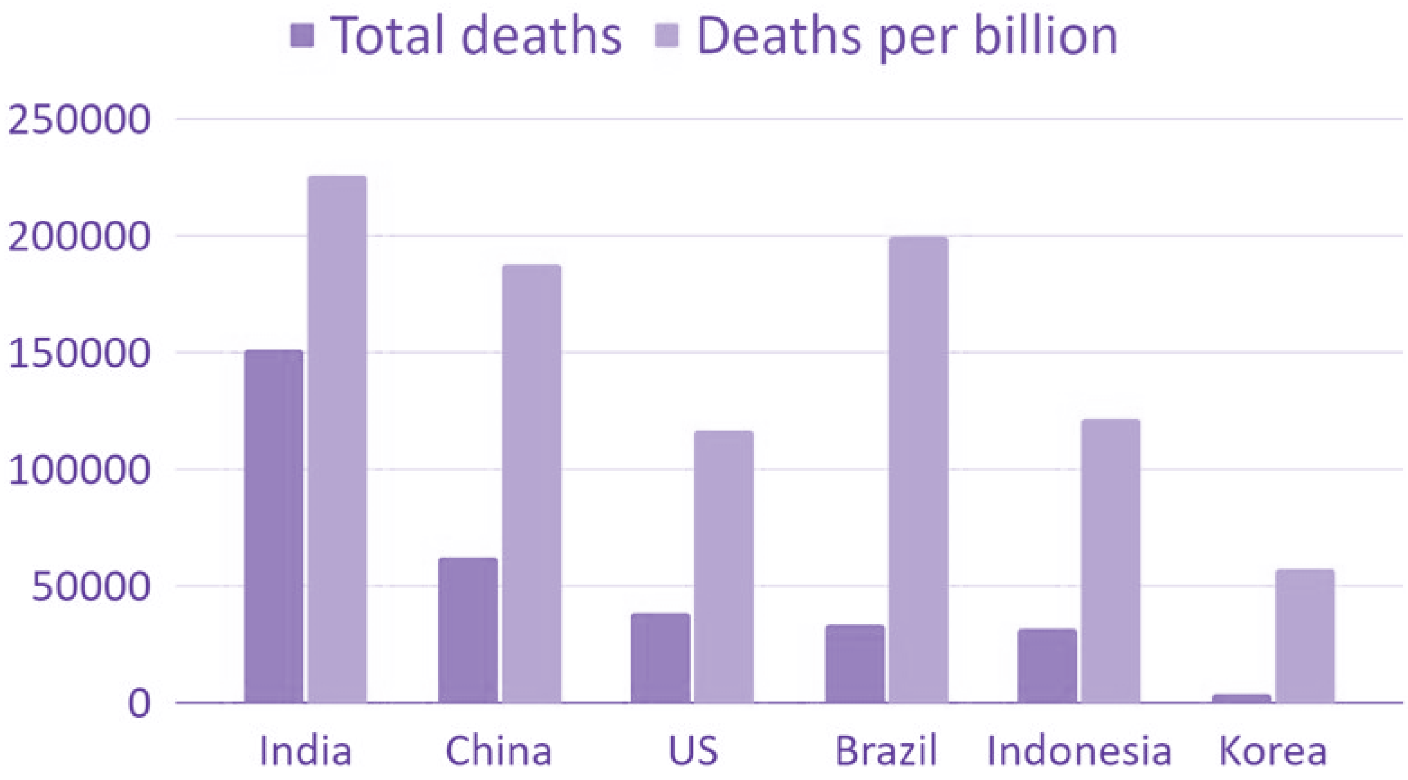
Ericsson Research

1

Motivation and Background

The Need for Traffic Surveillance

There is a widespread need to reduce the prevalence of traffic accidents



The Need for Traffic Surveillance

Lower enforcement of speed limits

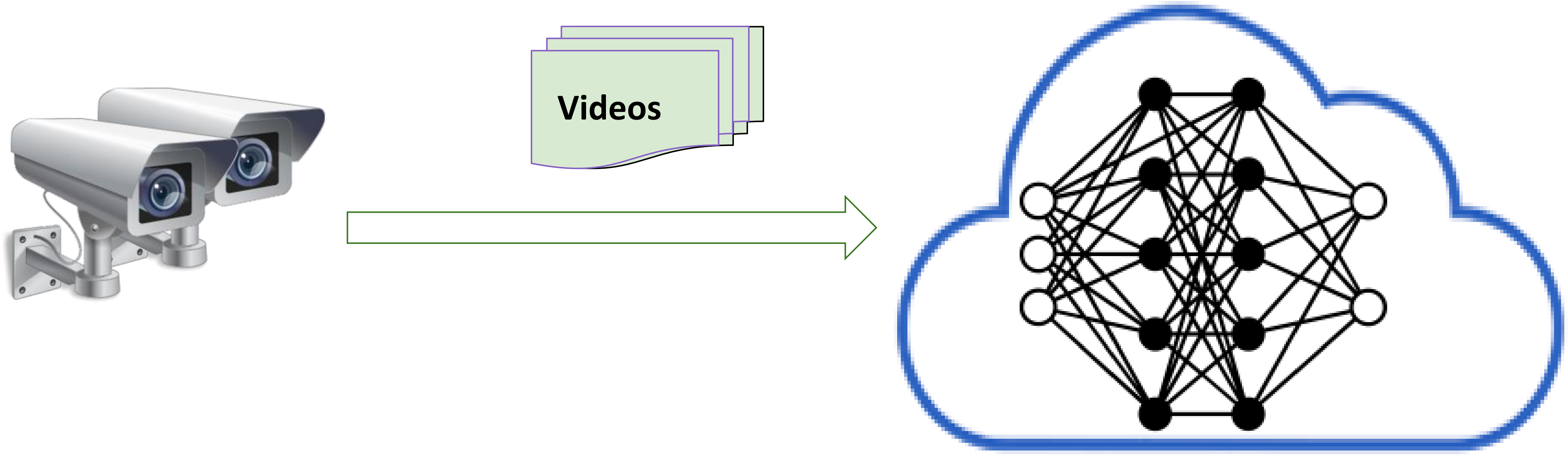
Simultaneous utilization

Unexpected obstructions

Traffic surveillance via cameras is seen as a solution

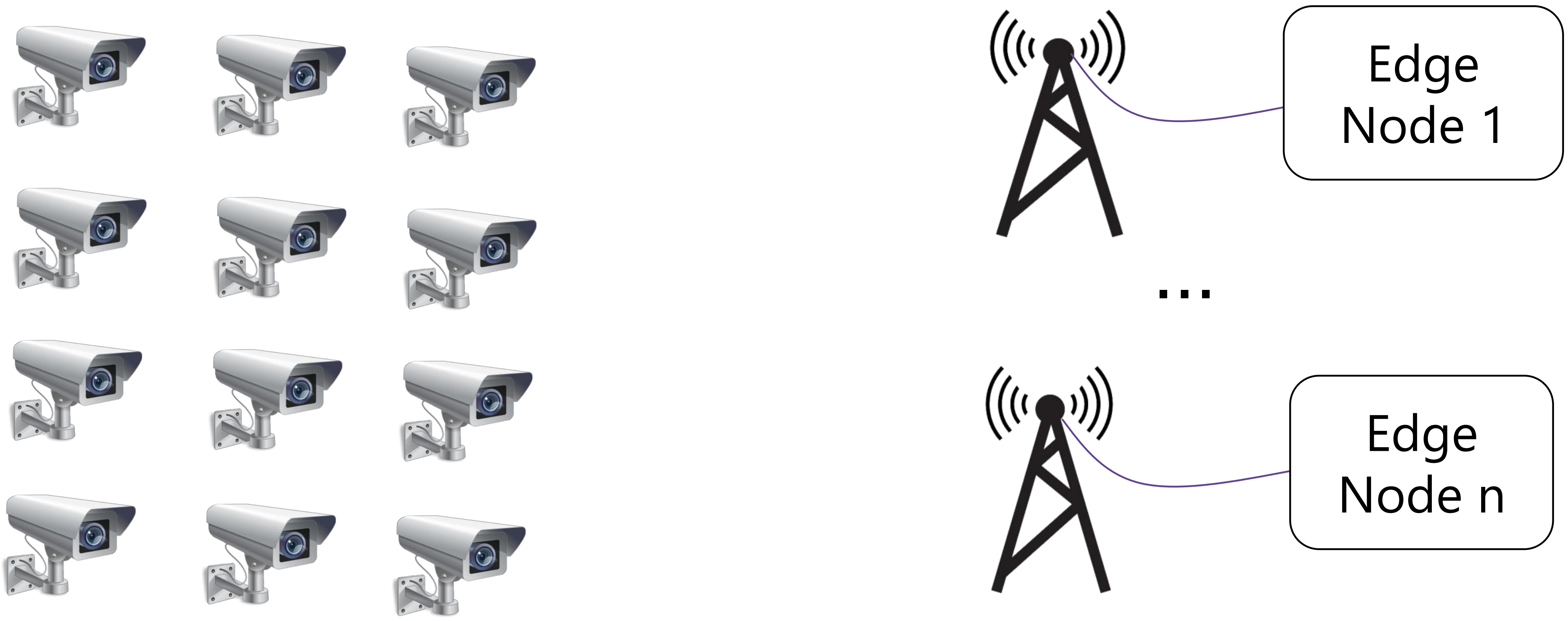
Working of Camera-based Traffic Surveillance

- Requires running computer vision techniques
  - In real-time only if specialized hardware (GPU's/NPU's) are available



Requires huge amount of computation, as large number of cameras are deployed in modern cities

Distributed deployment of edge computes nodes can handle such heavy computation

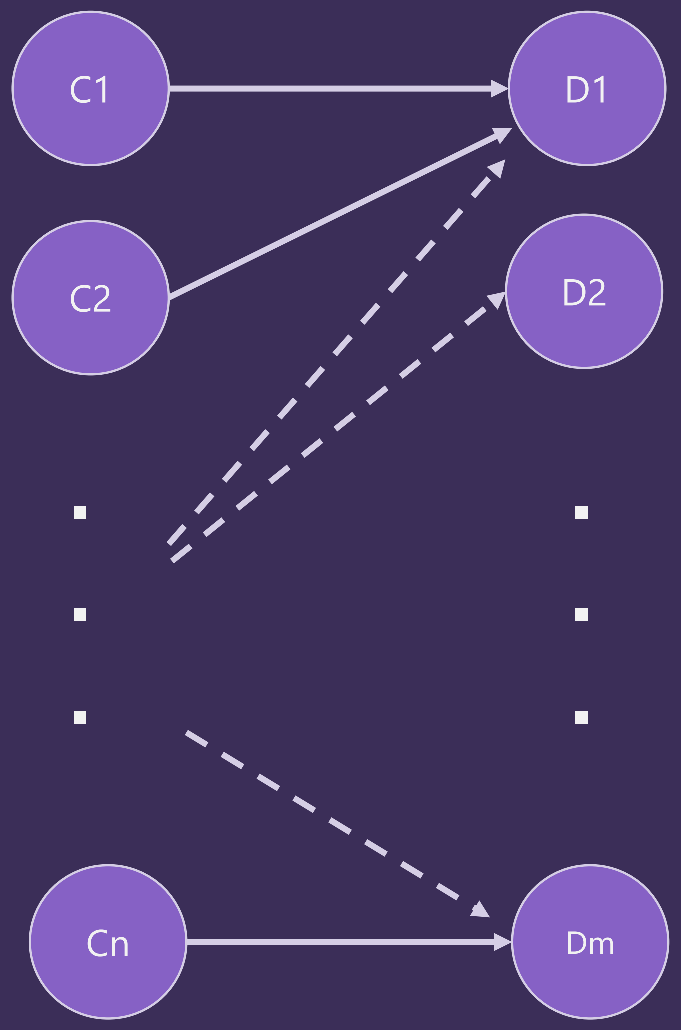


Distributed deployment of edge computes nodes can handle such heavy computation

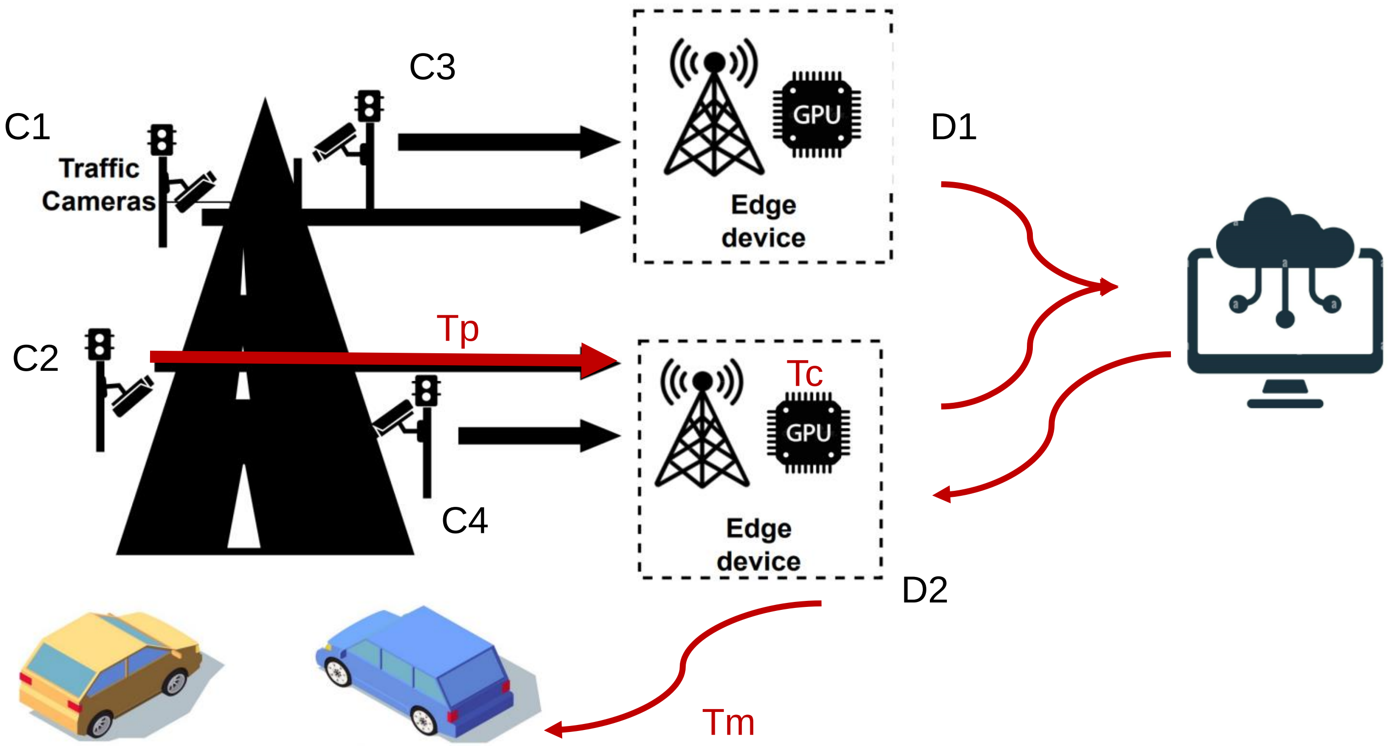
2

Problem Statement and Formal Model

Identifying the Edge Node for each Camera Video can be modelled as an assignment problem, where each camera is “assigned” to an edge device in a way that maximises the throughput while providing stochastic guarantees



Formal Model



01 Tp

The network latency taken to send the video feeds (Stochastic)

02 Tc

GPU processing time – execution of the ML model.
Additionally, it includes Cloud processing time (if necessary)

03 Tm

The latency incurred when the edge device sends alerts

Challenges

- How to identify the right edge node for each camera?
- How to consider the workload on each edge device while taking such a decision?
  - No proper support for virtualization in edge GPUs
- Can we use this system for a safety-critical application?

3

Observations and Proposed Solution

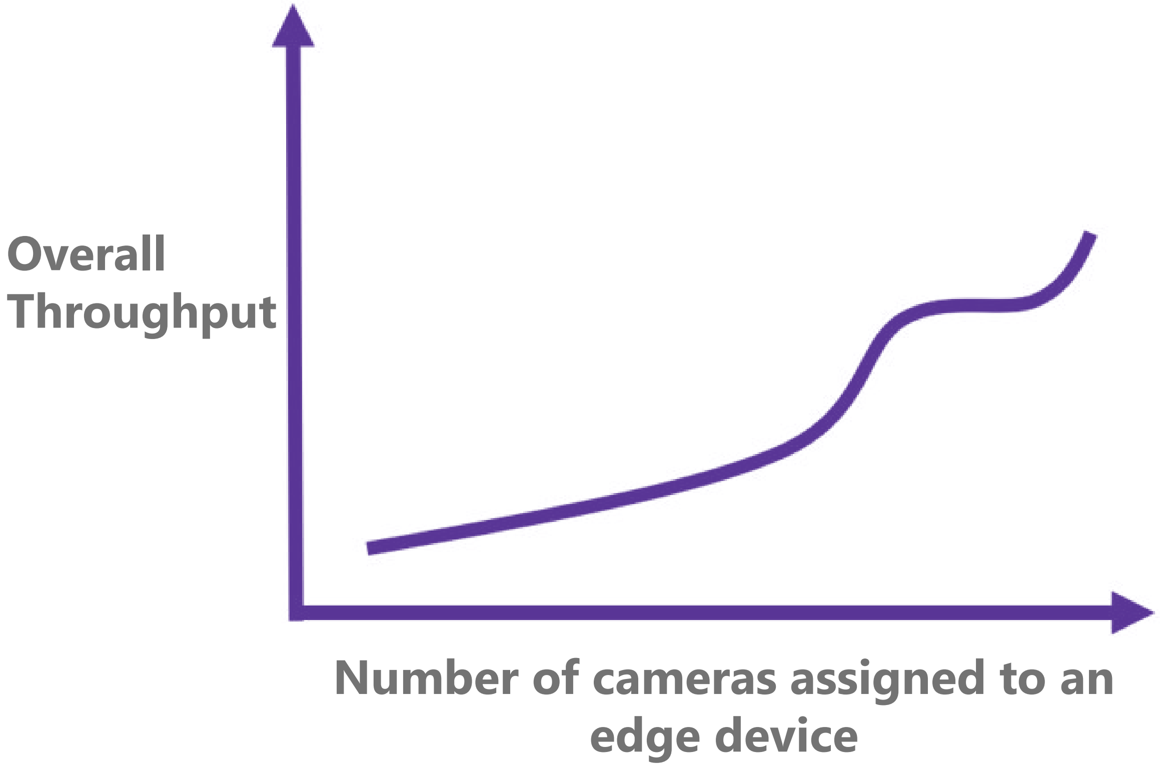
Observations

The number of attempts needed to successfully transmit video packets to the edge device is a random variable that follows negative binomial distribution

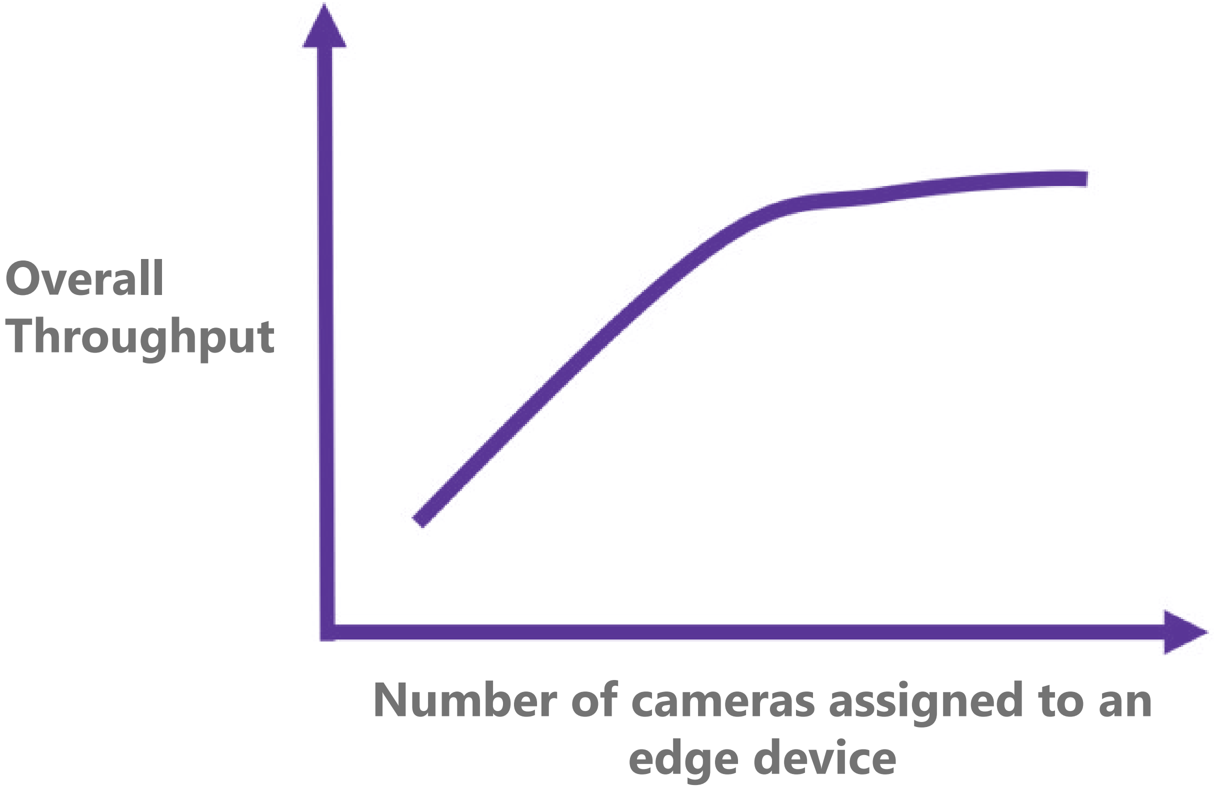
- This provides lower limit for the number of attempts ensuring stochastic guarantees and simplified the stochastic problem to a deterministic form
- Additionally, this makes our objective a form of nonlinear bin packing problem, which is known to be NP Hard.

Observations

Monotone

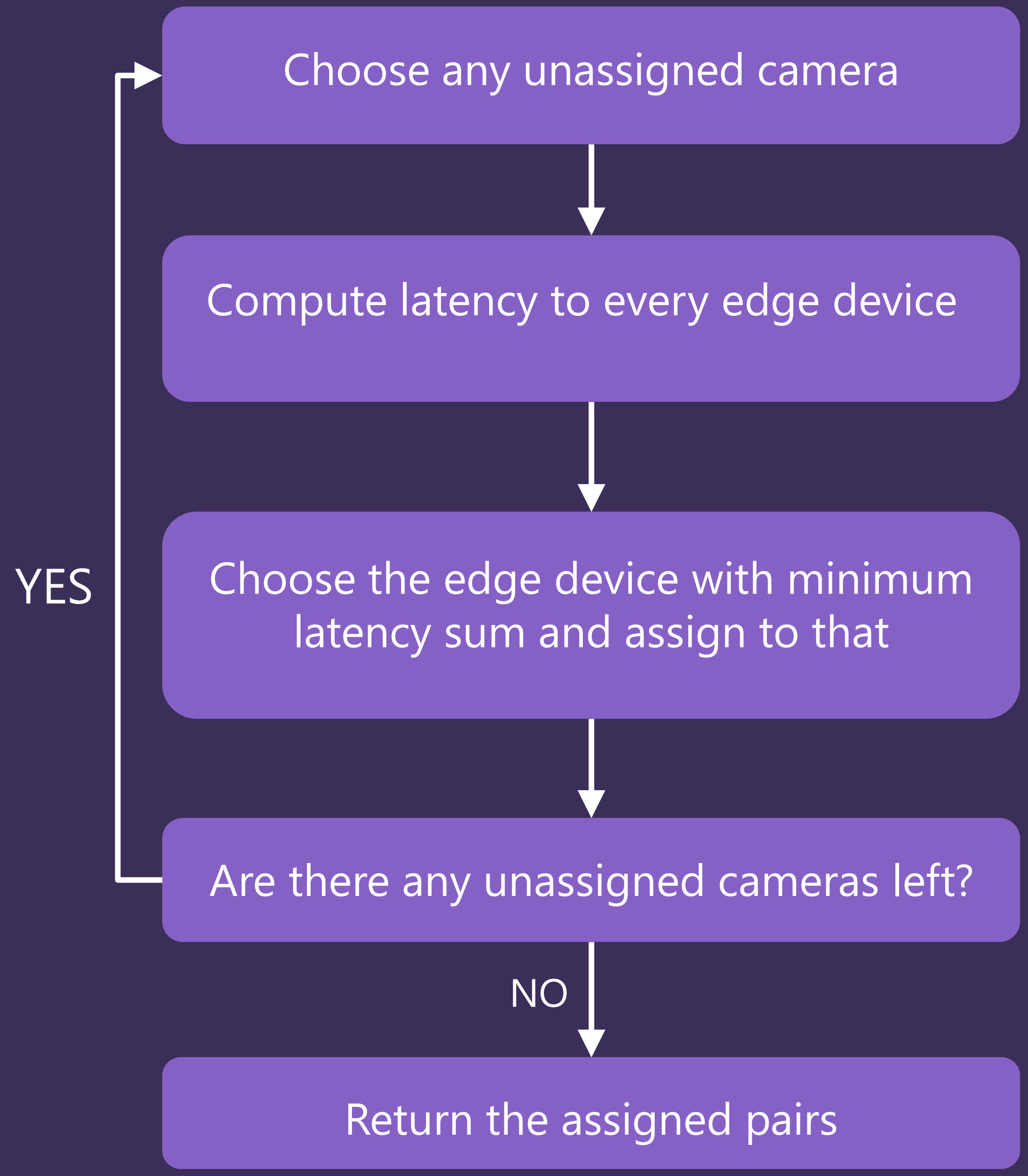


Submodular



Submodular and Monotone Objectives can be Maximized Using a Greedy Algorithm

This algorithm gives an approximation ratio of 0.5



4

Experimental Evaluation

Experimental Evaluation

Trace-driven simulation

We first find the traces by running YOLO-v5 on a Jetson Nano using a traffic surveillance dataset

Multithreading

We use the Python multithreading library available by default to create multiple threads and process the videos in parallel.

Time stamp generation

We use the time command to record the amount of time taken to run the entire program.

Scheduling

We run our greedy assignment algorithm and compare the results with several baseline algorithms

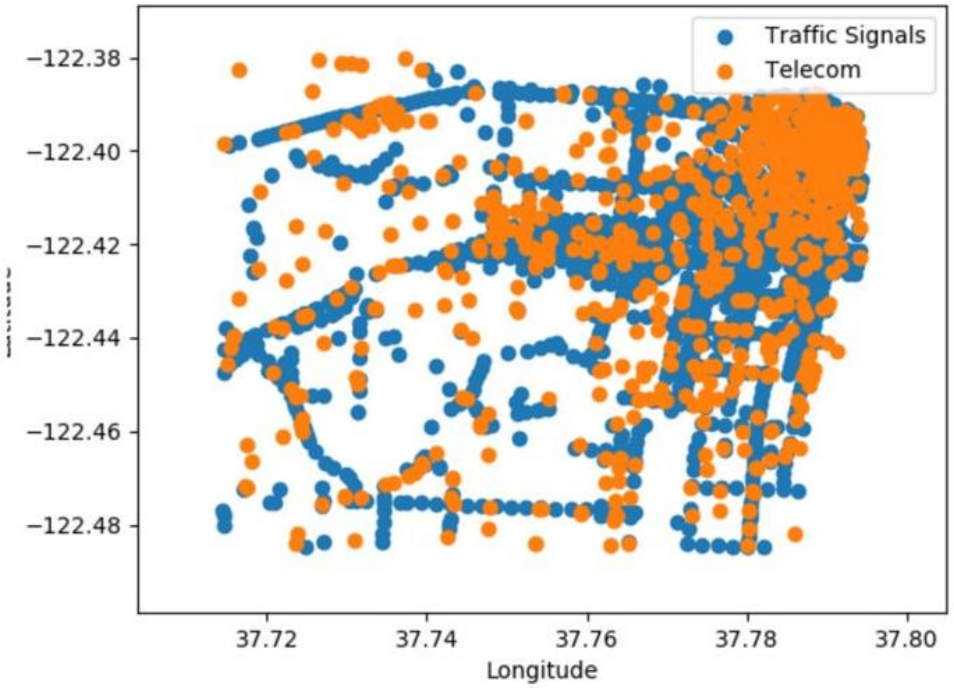
San Francisco Dataset

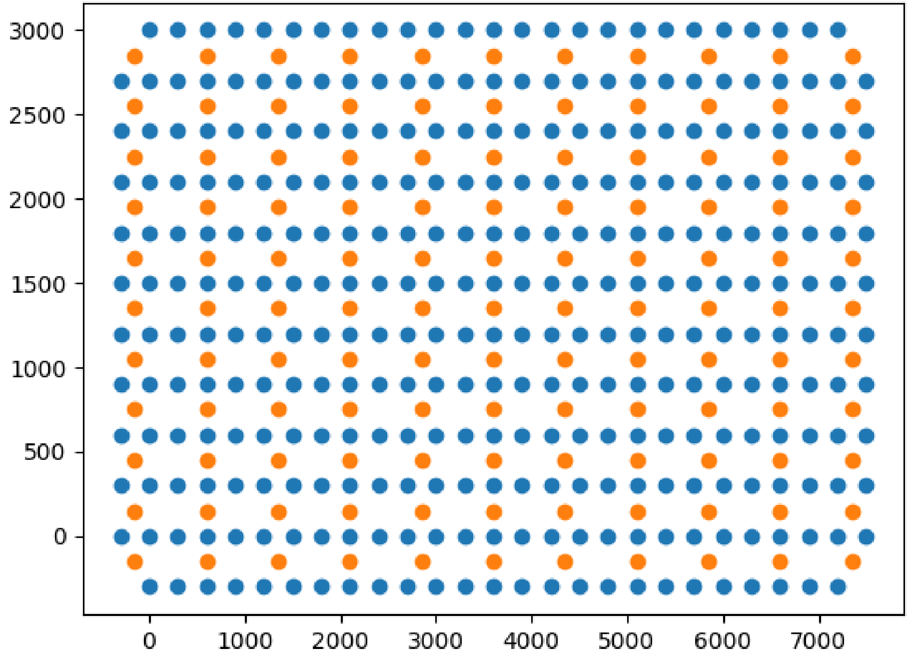
We generate the results for the first dataset, called San Francisco, consists of cell towers in a part of San Francisco city along with the road intersections

City Flow Grid Dataset

We repeat the steps for the second dataset, called City Flow Grid, uses the tool City Flow to synthetically generate a grid of roads.

Experimental Evaluation

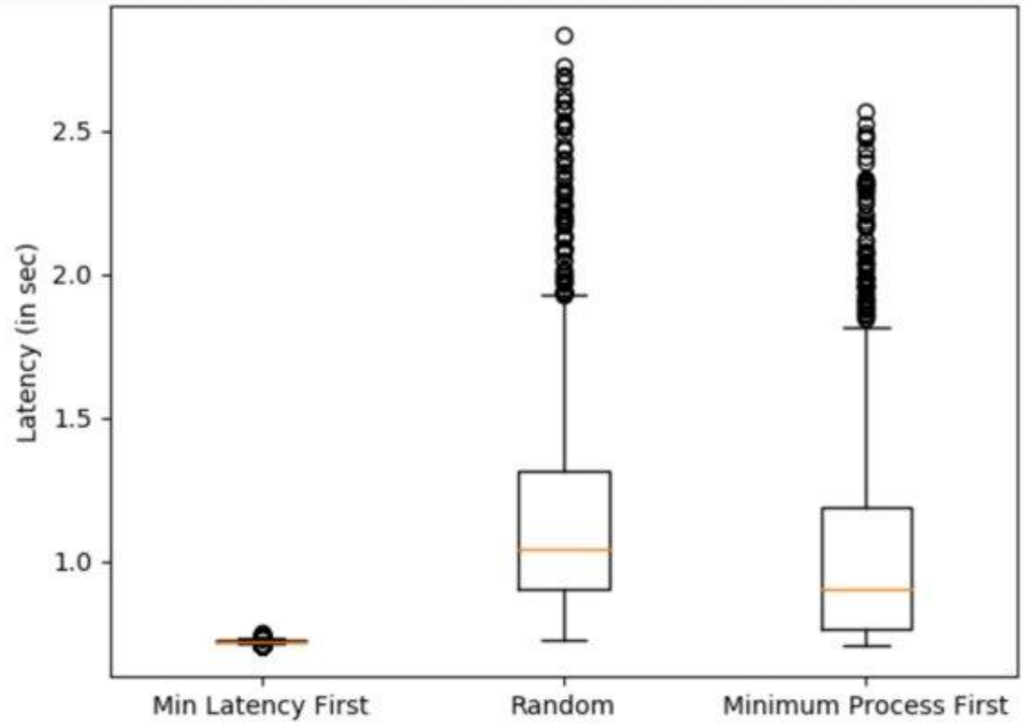




5

Results and Conclusion

Results – San Francisco Dataset



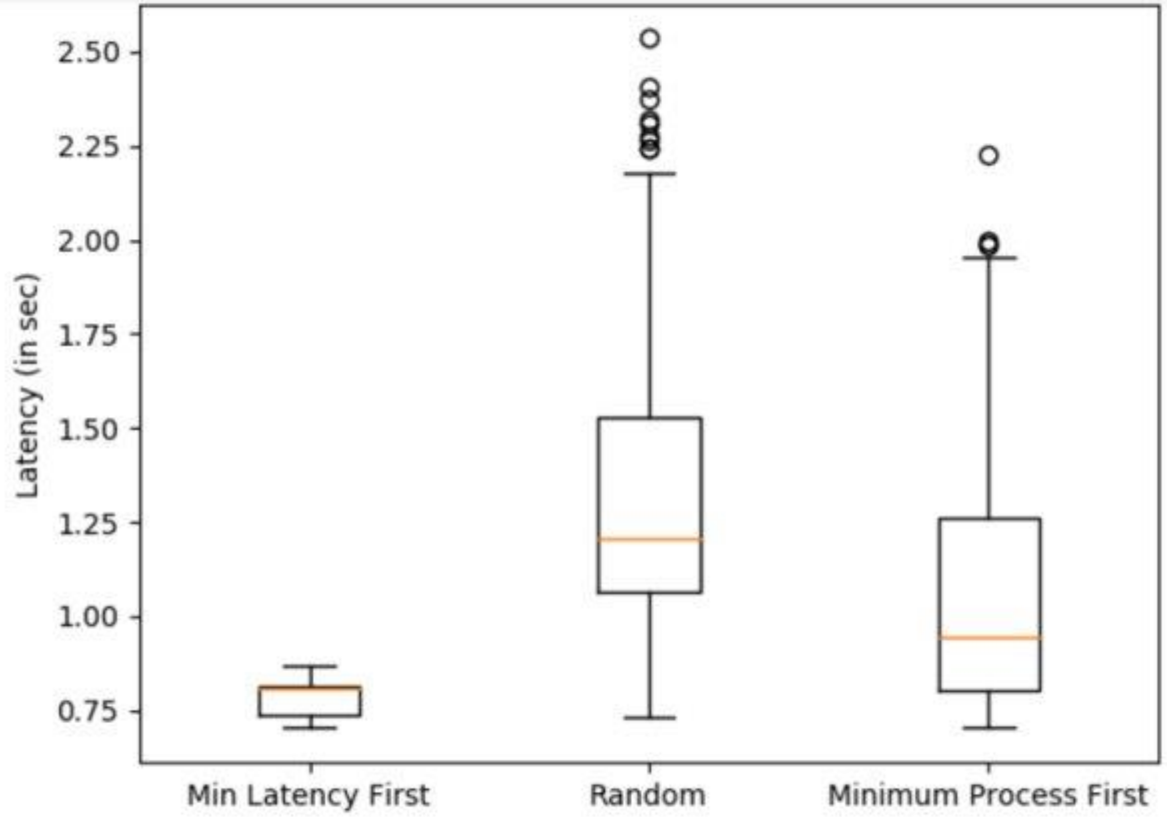
01 Min Latency First
Our greedy algorithm

02 Random
Sends the videos to any random edge device that is reachable from the camera

03 Min Process First
Sends the videos to the closest edge device, i.e. the one that has the minimum network latency

Our approach has a median latency of 0.35s, whereas the "Minimum Process" has a latency of 0.61s (an improvement of 42.7%) on the San Francisco dataset.

Results – City Flow Grid



01 Min Latency First

Our greedy algorithm

02 Random

Sends the videos to any random edge device that is reachable from the camera

03 Min Process First

Sends the videos to the closest edge device, i.e. the one that has the minimum network latency

On the city flow grid dataset, the latency values seen are equal to 0.78s and 0.97s (an improvement of 19.6%) for our approach and “Minimum Process” respectively.

Conclusion

1

We solve the problem of scheduling computer vision-based traffic surveillance on edge devices connected to cell towers

2

We observe that the execution time follows a monotone and submodular pattern and utilize this observation to design a greedy approximation algorithm.

3

We then evaluate this algorithm with a few reasonable baselines via both synthetic and real data and show that our algorithm performs much better in practice.