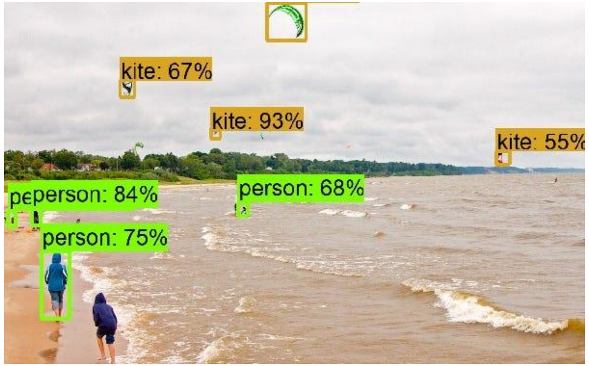
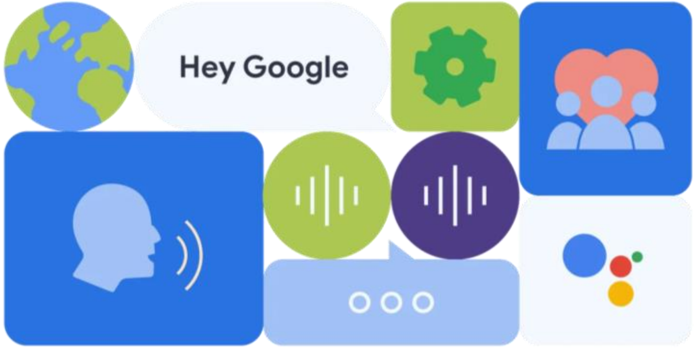
EcoEdgeInfer: Dynamically Optimizing Latency and Sustainability for Inference on Edge Devices

Sri Pramodh Rachuri, Nazeer Shaik, Mehul Choksi, Anshul Gandhi
PACE Lab, Stony Brook University

srachuri@cs.stonybrook.edu
pramodh.rachuri.dev

Motivation





SEC is a forum for top researchers, engineers, students, entrepreneurs, and government officials come together under one roof to discuss the opportunities and challenges that arise from rethinking cloud computing architectures and embracing edge computing.

SEC is a forum for top researchers, engineers, students, entrepreneurs, and government officials come under one roof to discuss the opportunities and challenges that arise from rethinking cloud computing architectures and embracing edge computing.

Motivation

- Deployments are usually inference servers
  - Light weight – resources, energy, processing time, memory etc
  - More frequent executions
    - Scale up in consumption
- User facing
  - Latency sensitive
  - Varying arrival rate

ChatGPT consumes 6-10X more energy than a Google Search - Goldman Sachs

ChatGPT's Energy Consumption
for Responding to Prompts and Its Cost in the U.S.

Time Period	Number of prompts	Energy Consumption	Cost
Day	214,285,714	621,429 kWh	$81,407
Week	1,500,000,000	4,350,000 kWh	$569,850
Month	6,517,860,000	18,901,794 kWh	$2,476,135
Year	78,214,350,000	226,821,615 kWh	$29,713,632

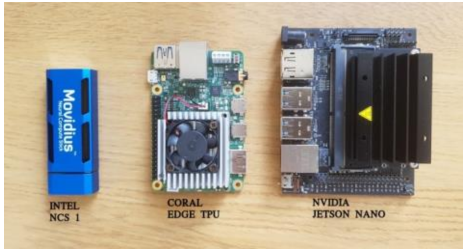
Data sources: U.S. Energy Information Administration, Electric Power Research Institute (EPRI)
Calculations based on: 100 million weekly users, 15 weekly prompts per user, 0.0029 kWh of energy consumption per prompt, average U.S. commercial electricity rate of $0.131/kWh as of June 2024

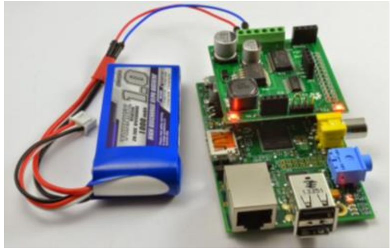
BestBrokers.com

Motivation

Edge for DNN inference

- Commercial products, Prior works
- Sits closer to the user
  - Low communication latency
- Local processing
  - Added privacy
- Possible remote locations
  - Limited energy supply
- Resource contention



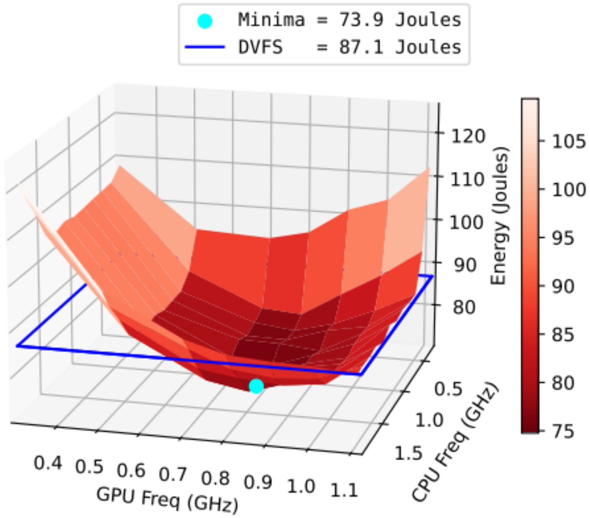


Problem Statement

Jointly optimize energy consumption and latency by tuning both hardware and software parameters under varying request arrival conditions of DNN inference workloads on Edge devices

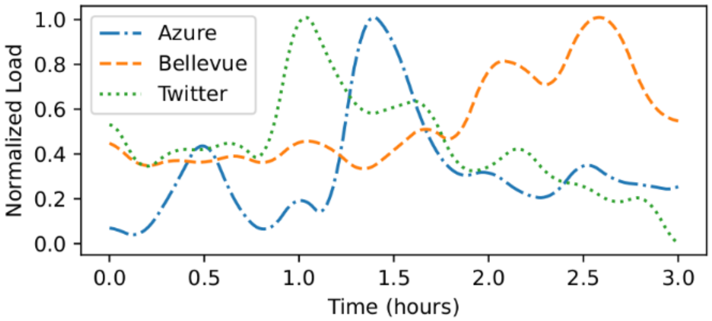
Challenges

- Knobs having non-linear affect on energy
  - CPU Frequency
  - GPU Frequency
- Knobs having complex interactions with latency
  - Batch Size
- Varying request arrival patterns complicates futher



[Source: Dutt et al., IGSC]

Challenges



- Too frequent arrivals ⇒ High load
  - Higher freqs for performance but energy inefficient
- Infrequent arrivals ⇒ Low load
  - Lower freqs for energy saving but higher latencies

- Batch Size
  - High batch size - better energy efficiency and throughput
  - Wait for batching - Increase latency
- Arrivals – Batch Size interaction

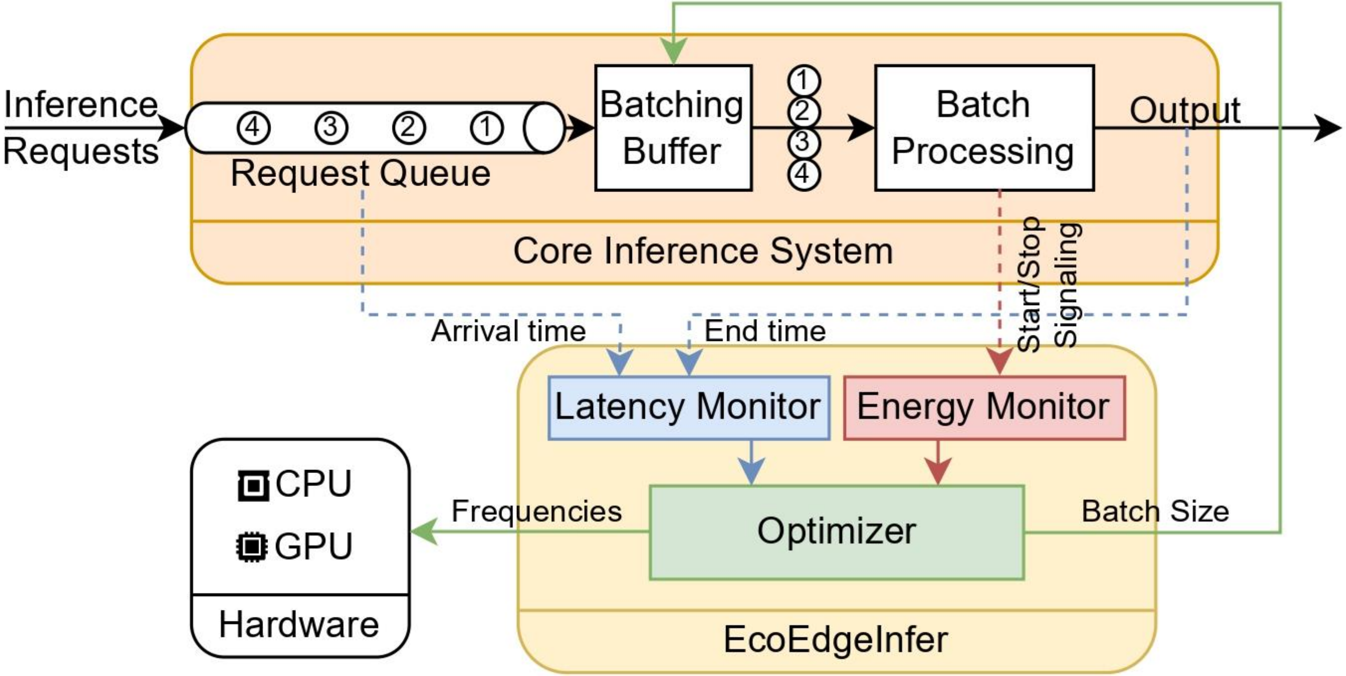
  Example- Consider batch size of 11
  - 100ms arrival interval ⇒ 1.1 s
  - 10ms arrival interval ⇒ 110 ms
  - Bursts of 10 every 100 ms

    ⇒ 200 ms!!!
    - Batch size = 10 ⇒ 100 ms

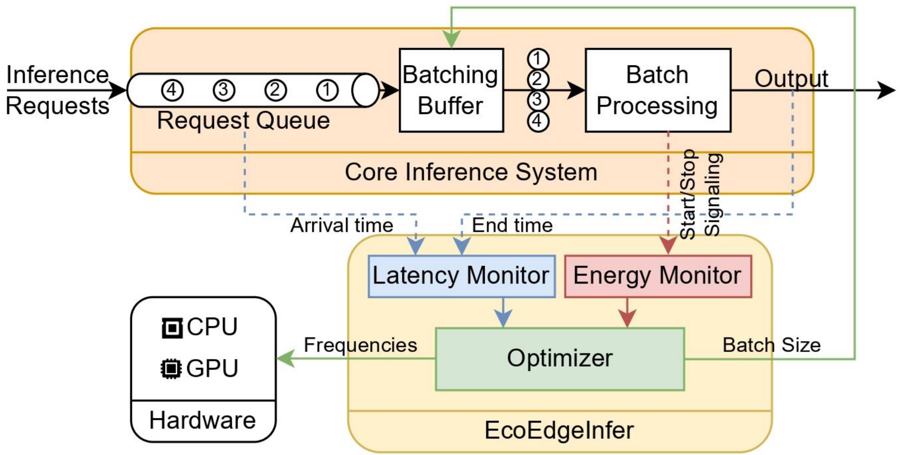
Outline

1. Motivation
2. Problem Statement
3. Challenges
4. **System Design**
5. Optimization Algorithm
6. Experimental Setup
7. Experimental Results
8. Conclusion

System Design

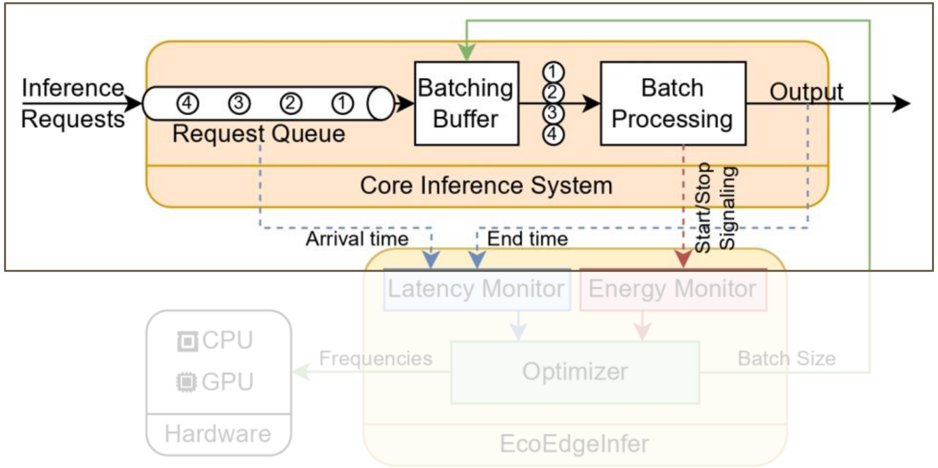


System Design



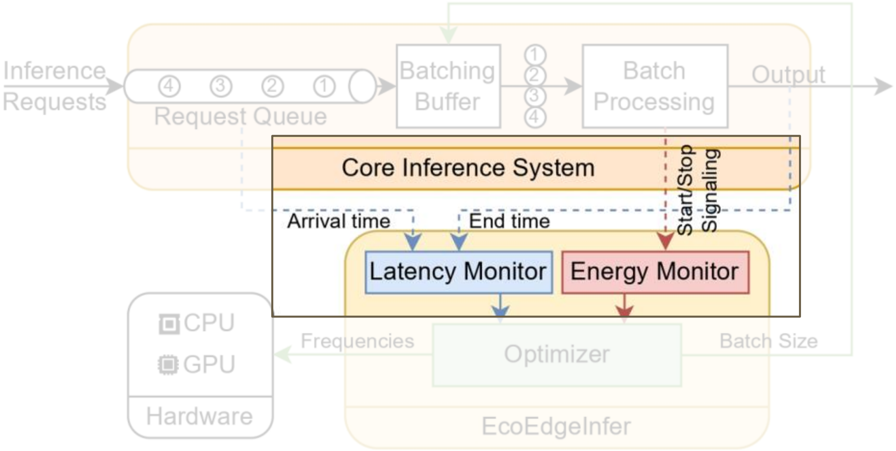
System Design

- Maintains a request queue
  - Store when busy
- Maintains a buffer of size = batch size
- Runs the DNN code provided by the developer
- Other signaling



System Design

- Collects latency using timestamps from core inference system
  - All waiting times are also included
- Records energy consumed per batch
  - Uses Nvidia APIs through sysfs, I2C

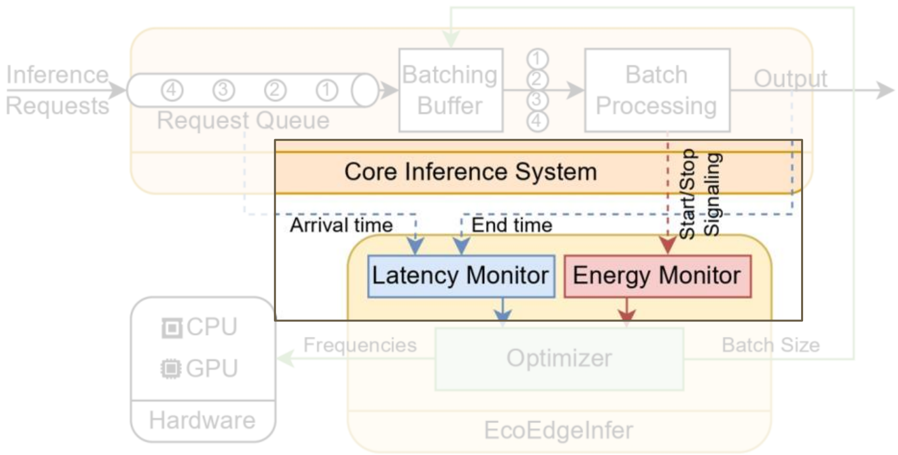


System Design

- Homogenize latency and energy of group of requests into cost

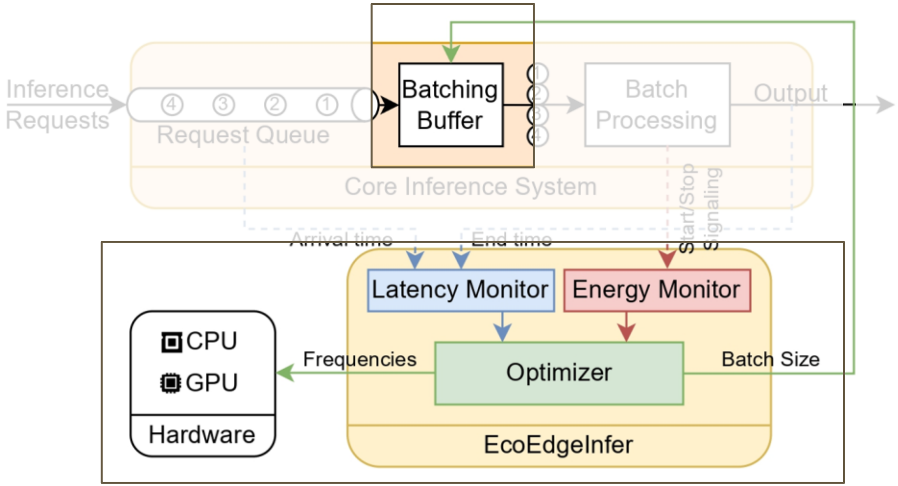
$$\text{Cost} = \frac{1}{2}\left(\frac{E}{E_o} + \frac{L}{L_o}\right)$$

E_0 and L_0 are measured when params are set to max



System Design

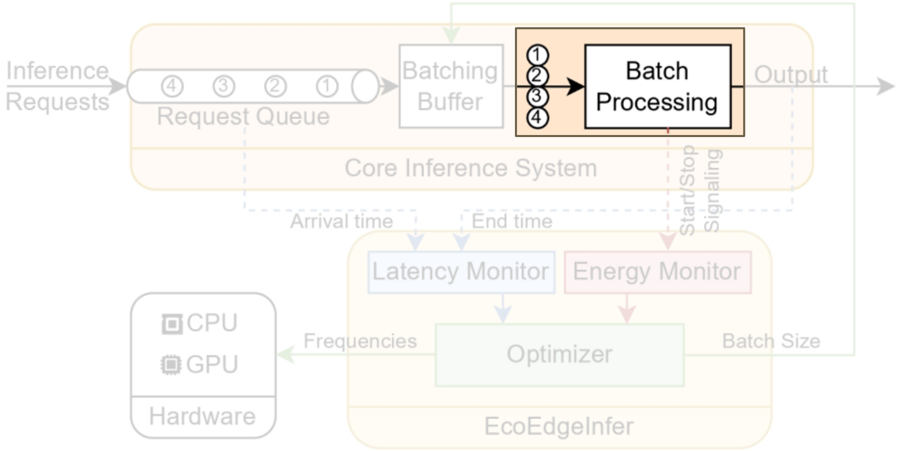
- Optimizer takes cost & predicts a config
- Output of optimizer
  - CPU - cpufrequtil
  - GPU - NVIDIA's devfreq framework
  - Batchsize - Batching buffer



System Design

Made to easily integrate with exiting PyTorch inference scripts using Python decorators

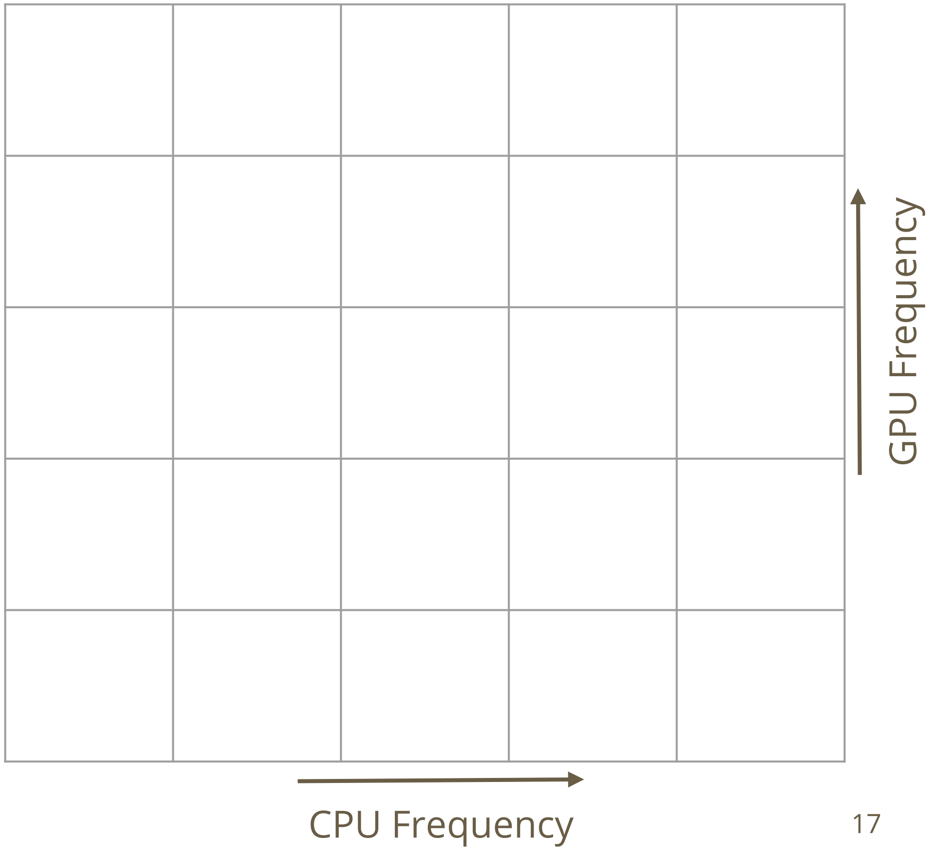
```
@eco_edge_infer.inference_method
def run_inference(input_data):
    output_data = model(input_data)
    output_handler(output_data)
run_inference(inference_request)
```



Outline

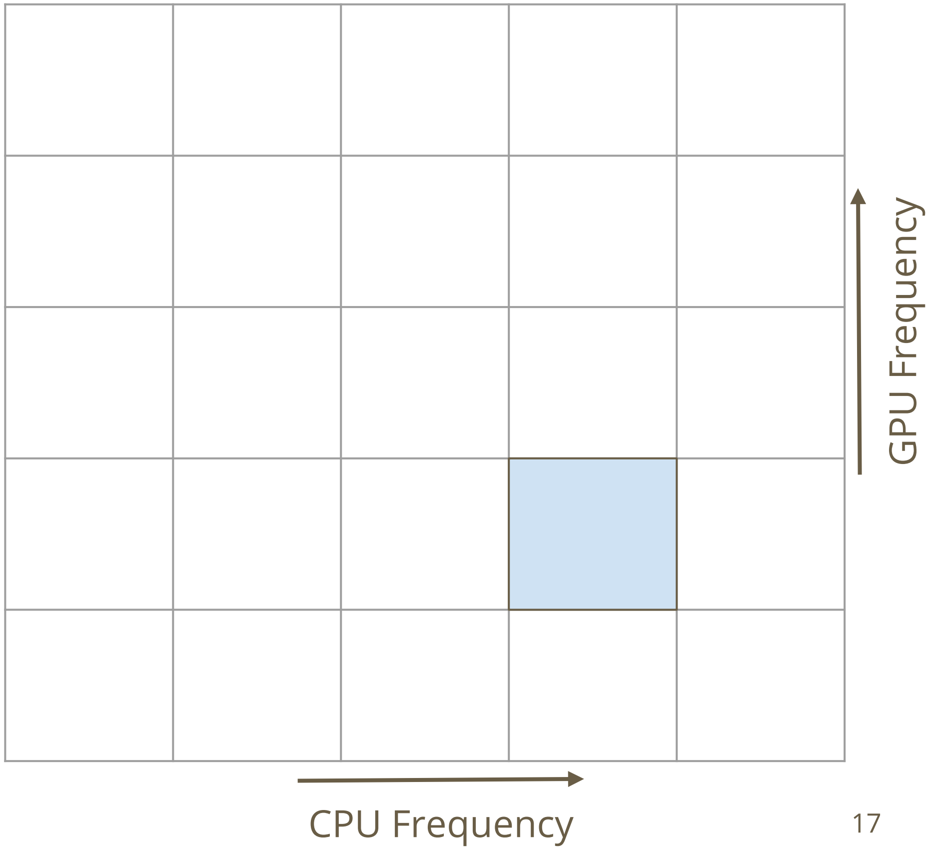
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



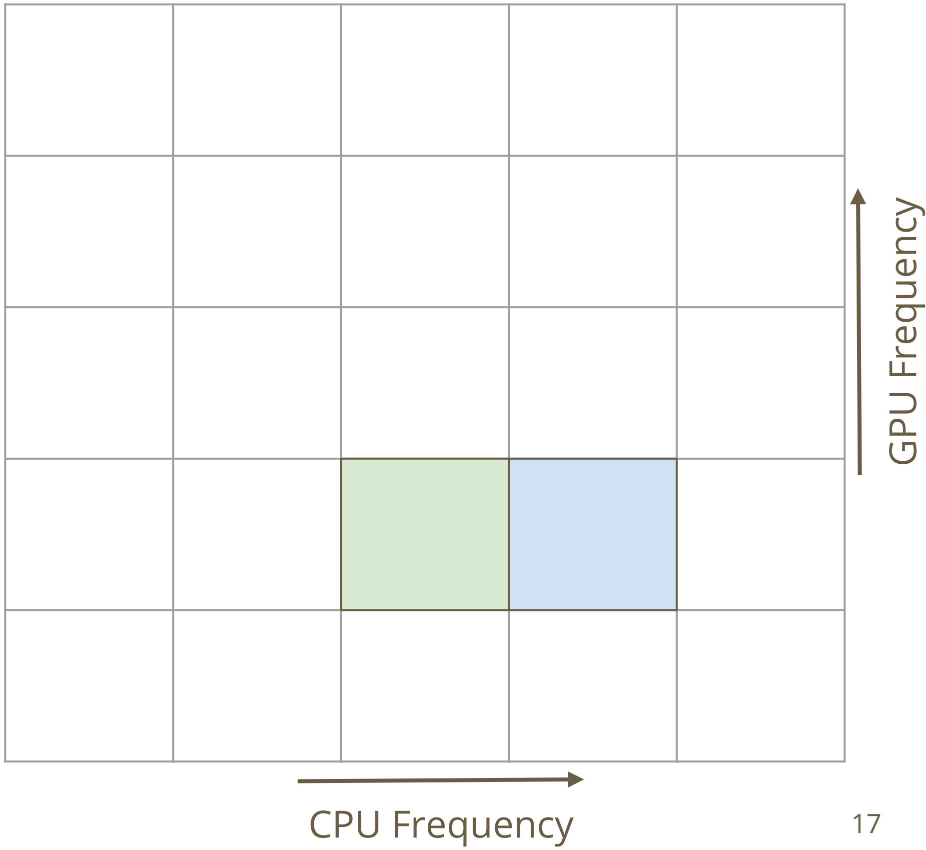
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
    - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
    - Explores 3D neighborhood
        - Only 6-neighborhood
    - Jumps to a better center
        - Loop detection
        - More loops than threshold ⇒ Jump outside 6-nbhrhood



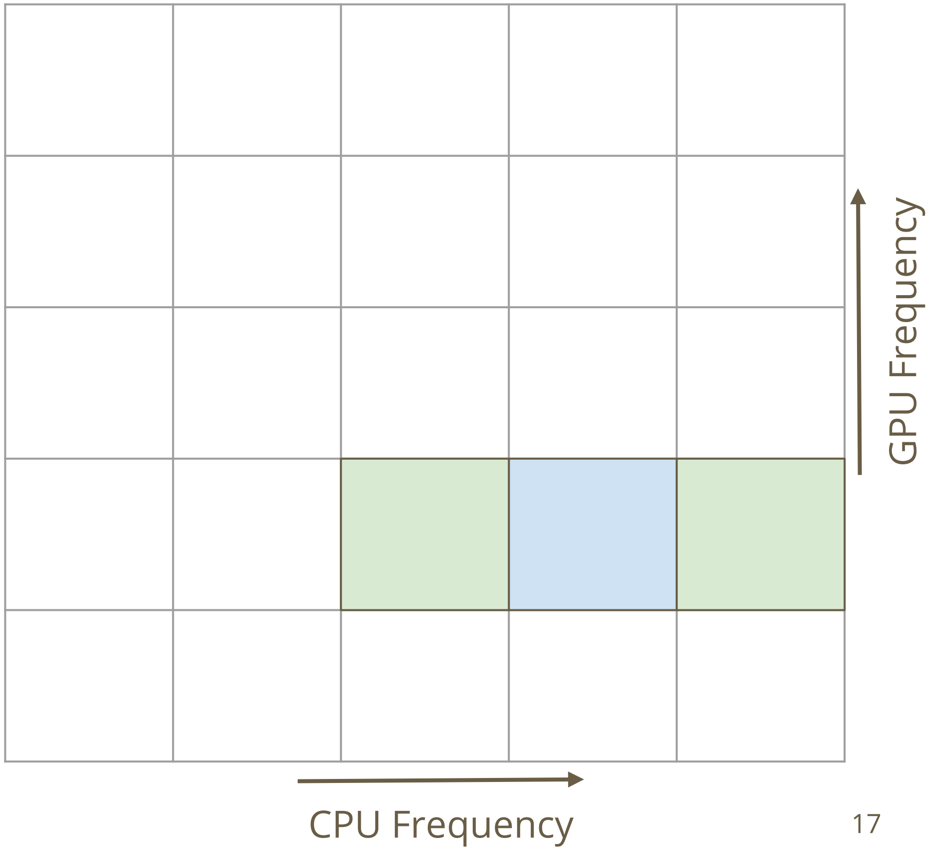
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



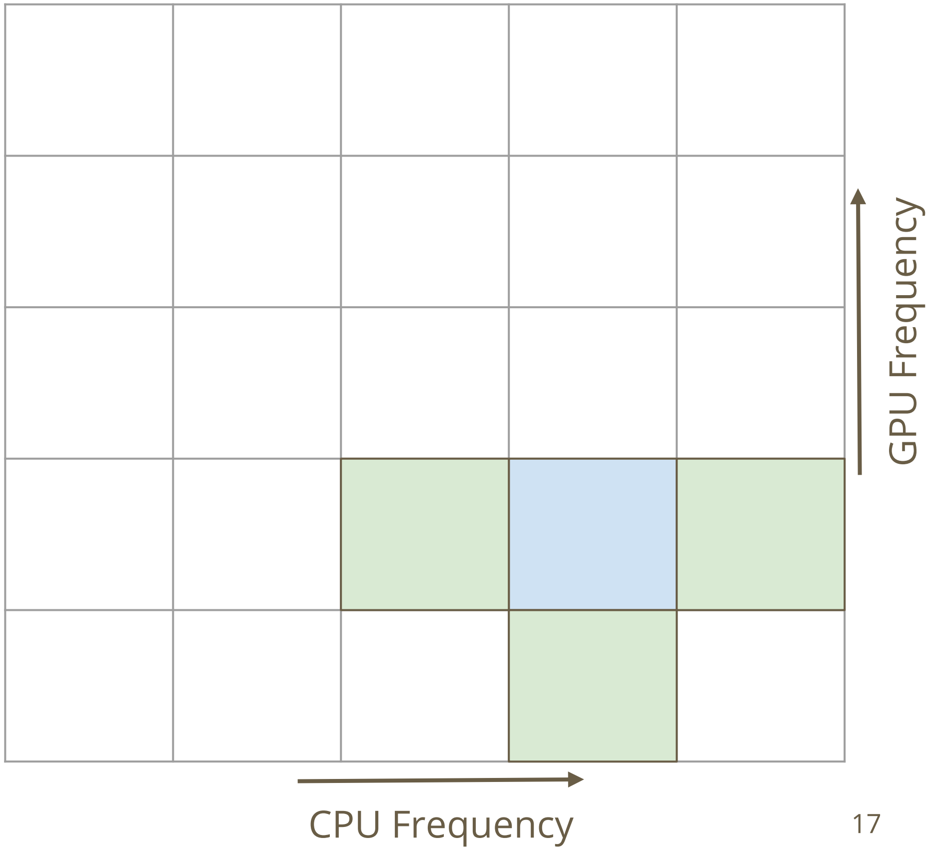
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



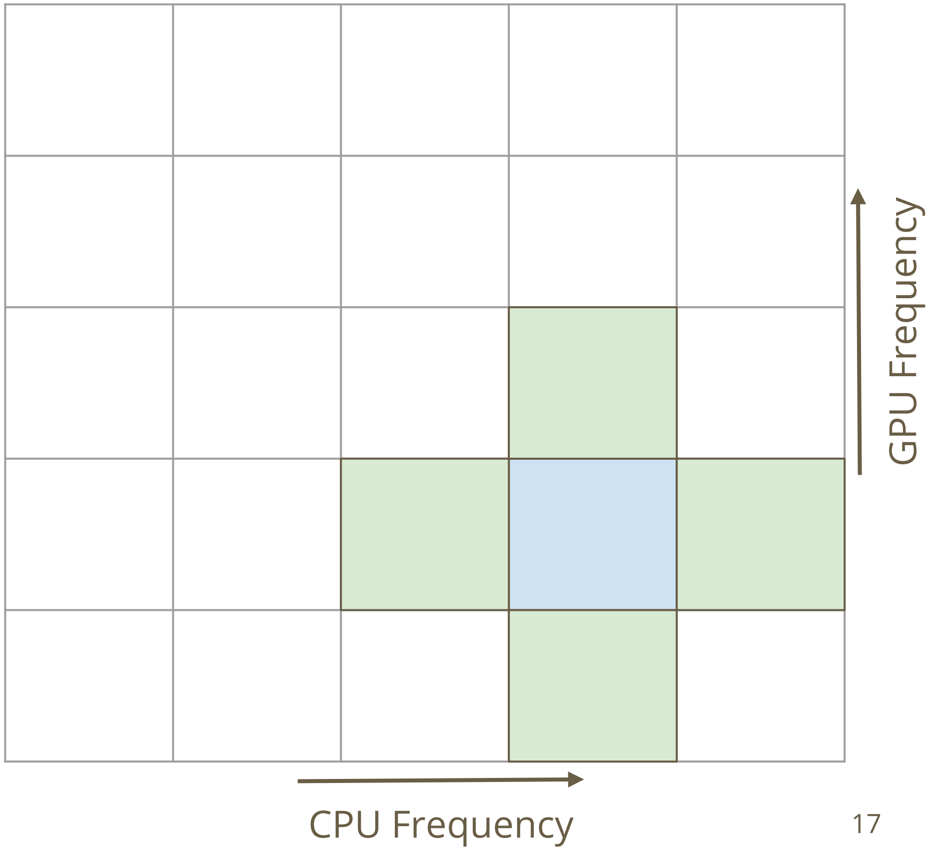
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



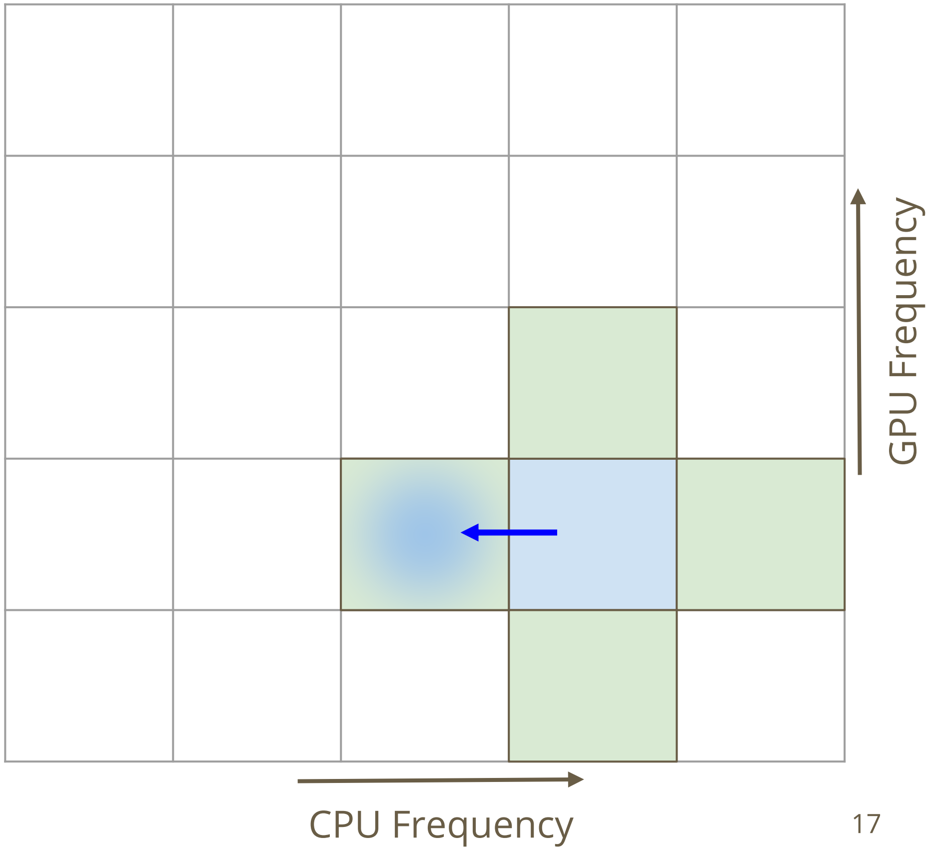
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



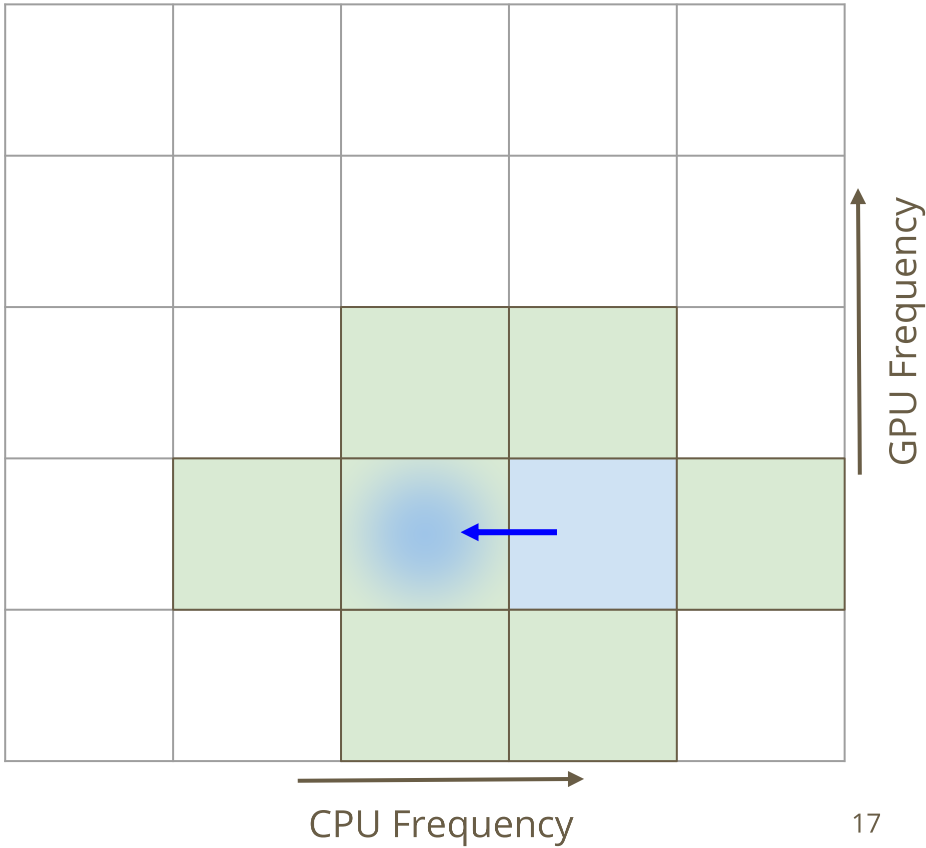
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
    - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
    - Explores 3D neighborhood
        - Only 6-neighborhood
    - Jumps to a better center
        - Loop detection
        - More loops than threshold ⇒ Jump outside 6-nbhrhood



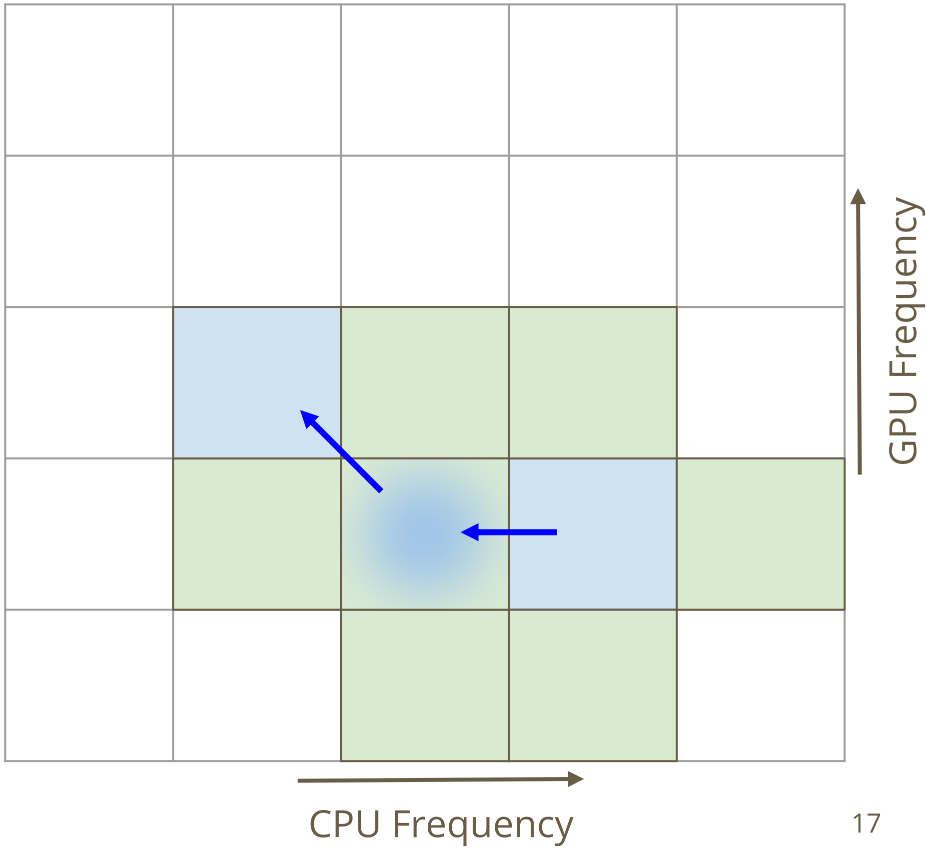
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



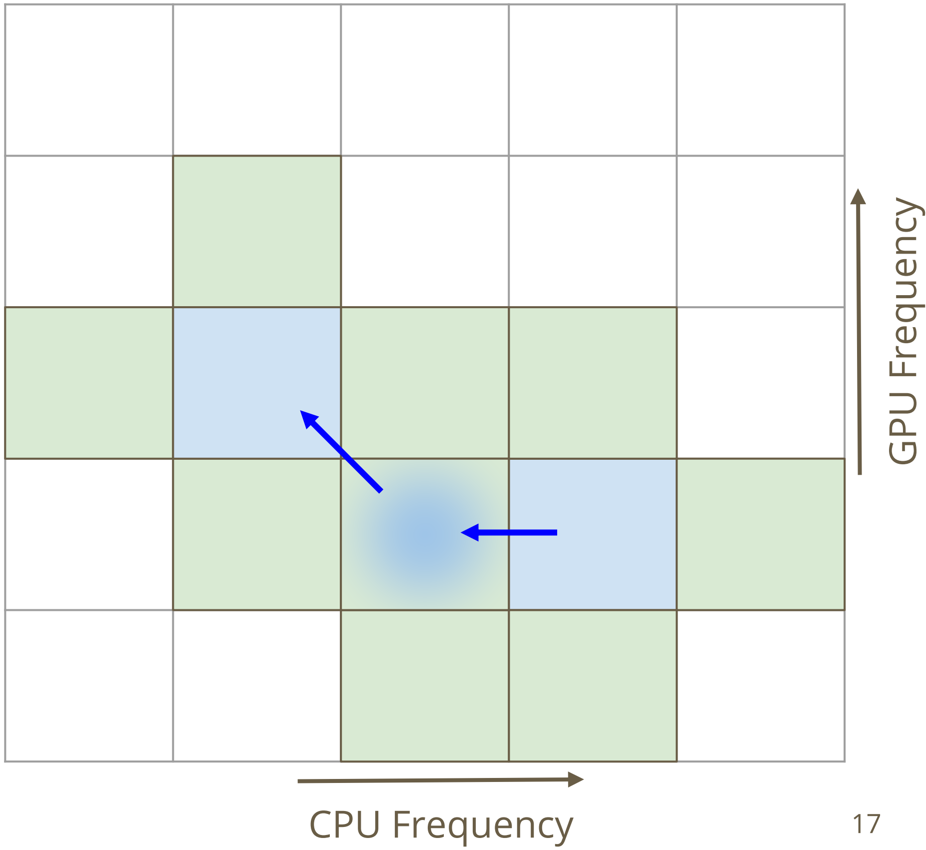
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



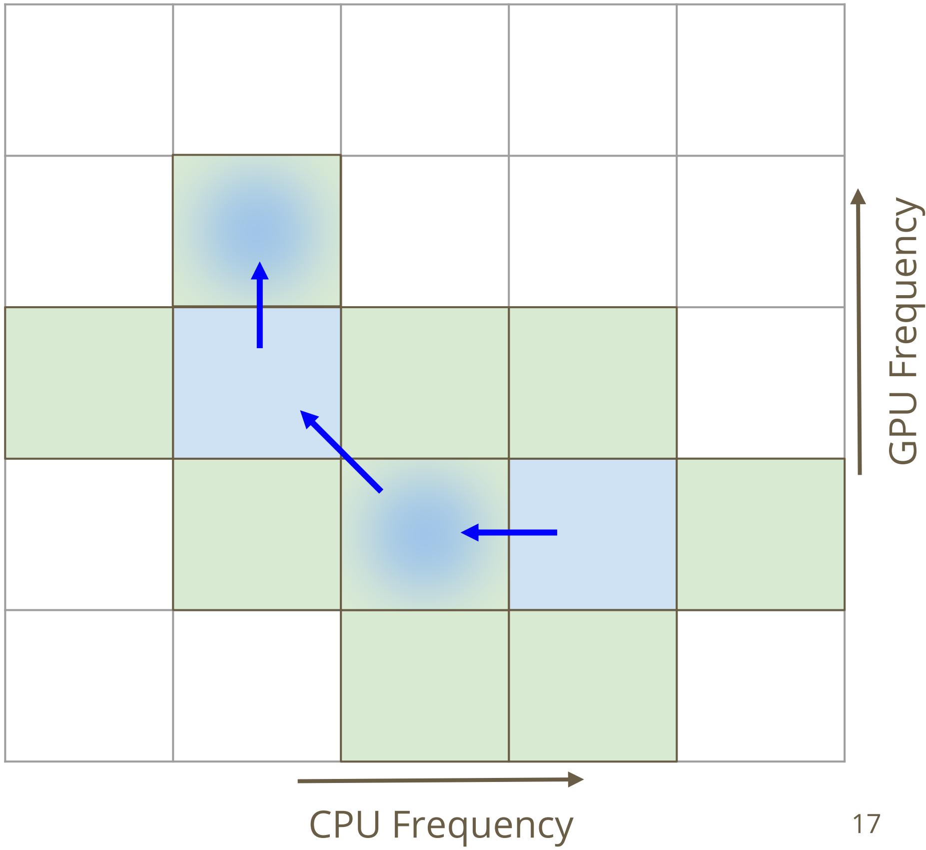
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
  - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
  - Explores 3D neighborhood
    - Only 6-neighborhood
  - Jumps to a better center
    - Loop detection
    - More loops than threshold ⇒ Jump outside 6-nbhrhood



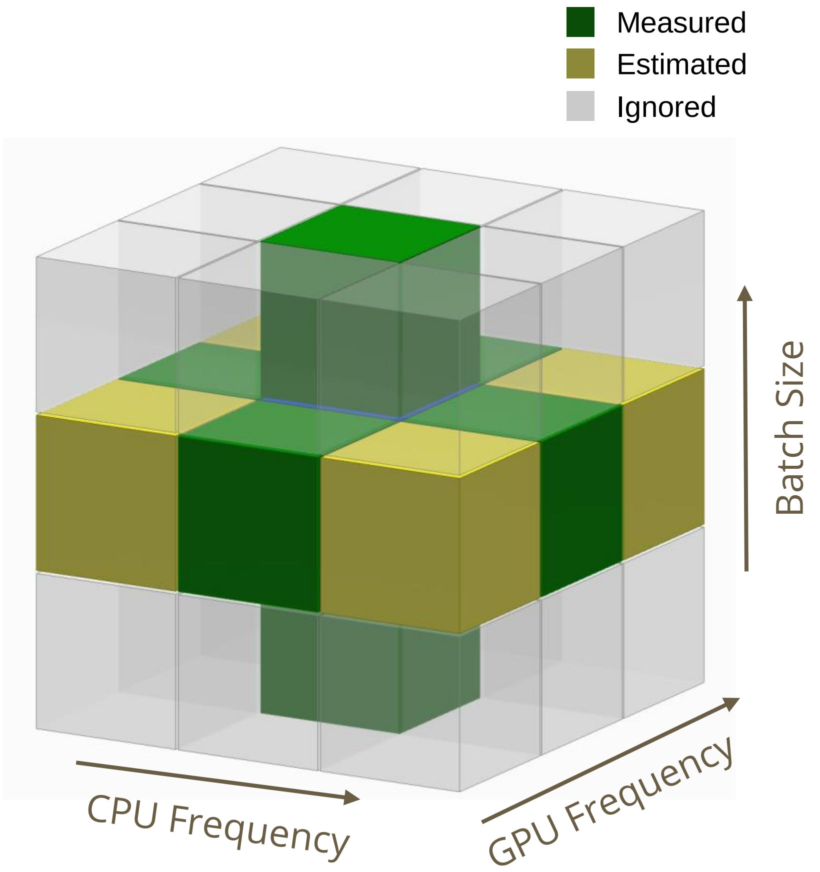
Optimization Algorithm: EcoGD

- Similar to Hill Climbing Local Search algorithm
    - Think of it as - gradient descent but only direction
- Cost is the objective function
- At every optimizer step, it either-
    - Explores 3D neighborhood
        - Only 6-neighborhood
    - Jumps to a better center
        - Loop detection
        - More loops than threshold ⇒ Jump outside 6-nbhrhood



Optimization Algorithm: EcoGD

- Diagonal estimations
  - Data available - 6 Neighborhood
  - Diagonals in CPU – GPU plane can be estimated
  - Diagonals in planes involving Batchsize cannot be estimated
- What is a better center?
  - Lowest cost among 6+4 neighbors
- History trimming
  - Adapt to changing arrival rates
  - Saves memory



Outline

Evaluation Setup

- Nvidia Xavier NX
  - 25 CPU Frequencies
  - 15 GPU Frequencies
- Inference Workloads
  - Resnet 50, BERT-Tiny
  - Batch Sizes: 1-16
- Search Space
  - 25 X 15 X 16 = 6000

- 3 reruns
- Duration
  - Synthetic pattens – Till convergence
  - Traces – 3 hours

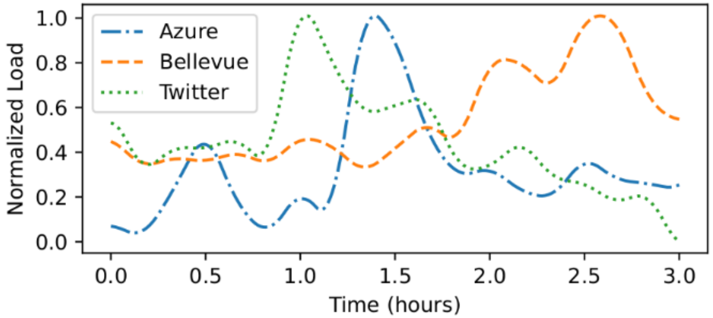
TECHNICAL SPECIFICATIONS OF NVIDIA XAVIER NX

Specification	Value
CPU	6-core Nvidia Carmel
CPU Freq. range	115 MHz – 1.9 GHz; 25 steps of 77 MHz
GPU	NVIDIA Volta
GPU Cores	384 CUDA Cores + 48 Tensor Cores
GPU Freq. range	114 MHz – 1.1 GHz; 15 steps of 90 MHz
Memory	8 GB LPDDR4x
Throughput	21 TOPs
Default Power Modes	10W, 15W, 20W
Jetpack version	5.1.3 [L4T v35.5.0]
Framework	PyTorch 2.1.0
Operating Sys. & Libraries	Ubuntu 20.04.6; CUDA 11.4 + cuDNN 8.6

Evaluation Setup

Request Arrival Pattern

- Fixed inter-arrival times between requests
  - 50 ms/90 ms
- Bursty arrivals
  - 10 requests arriving together every 500 ms/900 ms
- Bellevue Traffic Camera
  - Num of cars passing through
  - Proxy for Edge Traffic Video Analytics
- Twitter Stream Dataset
  - 1% sample of tweets in US East
  - Proxy for Edge content moderation
- Azure Functions Traces
  - Calls to azure serverless functions
  - Proxy to Serverless Edge Computing



Evaluation Setup

Comparison Baselines

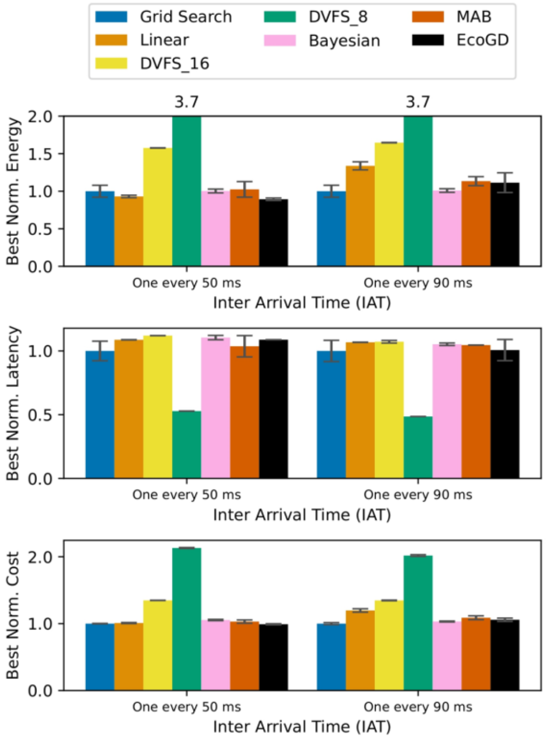
- Grid Search
  - Brute force method. Tries all configurations
- Linear Search
  - Sweeps each dimension one by one
  - Keeps sweeping forever
- Dynamic Voltage Frequency Scaling (DVFS)
  - CPU - schedutil
  - GPU - nvhost_podgov
  - Batch Size - 8, 16
- Bayesian Optimization
- Multi-Armed Bandit (MAB)

In detail explanations and exact hyperparameters can be found in the paper.

Outline

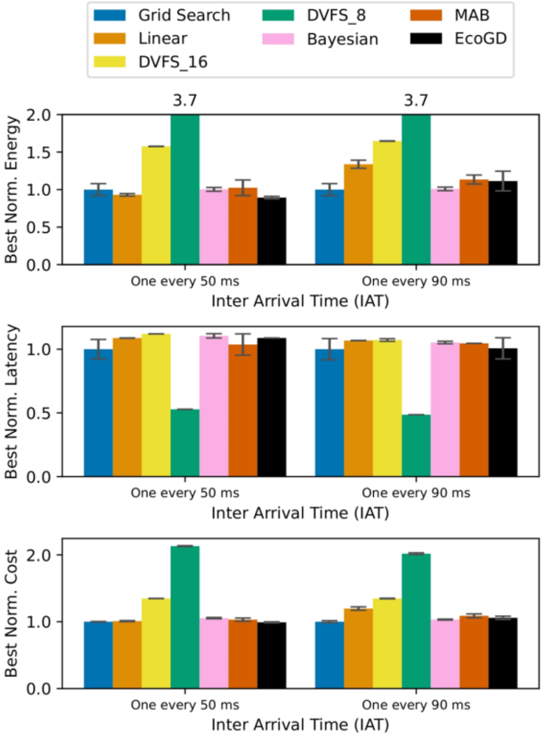
Experimental Results: Fixed Loads

Experimental Results: Fixed Loads

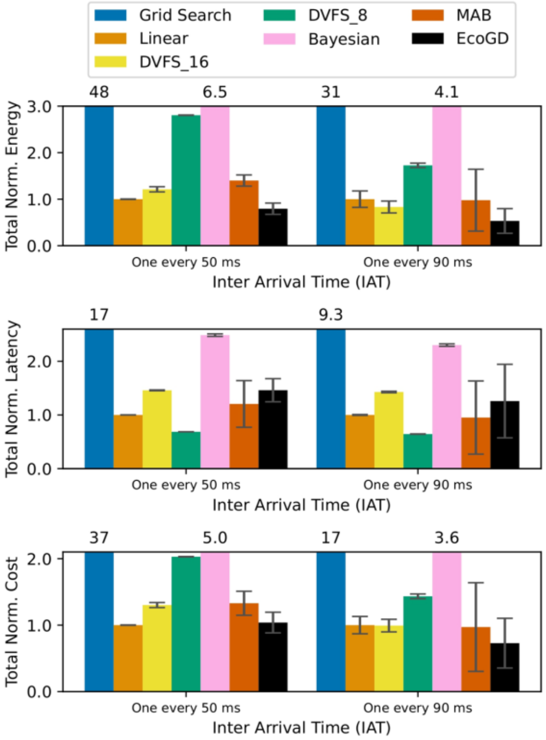


Metrics **after** convergence

Experimental Results: Fixed Loads

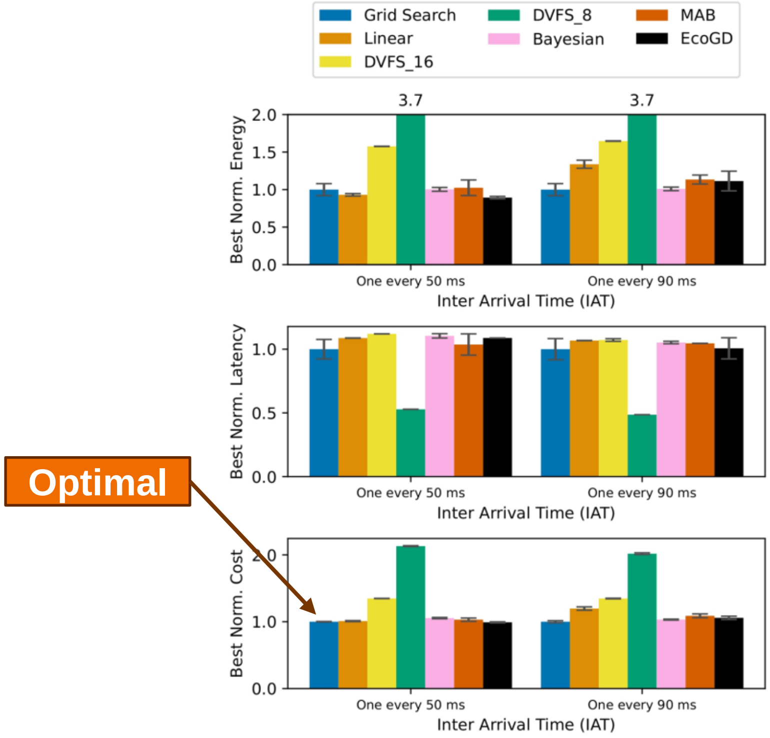


Metrics **after** convergence



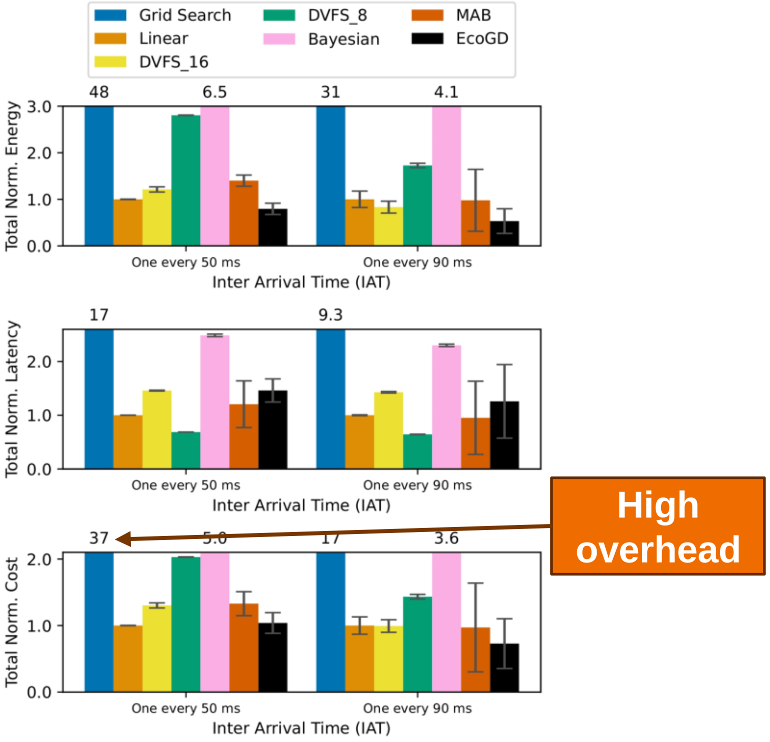
Metrics **during** convergence (overhead)

Experimental Results: Fixed Loads



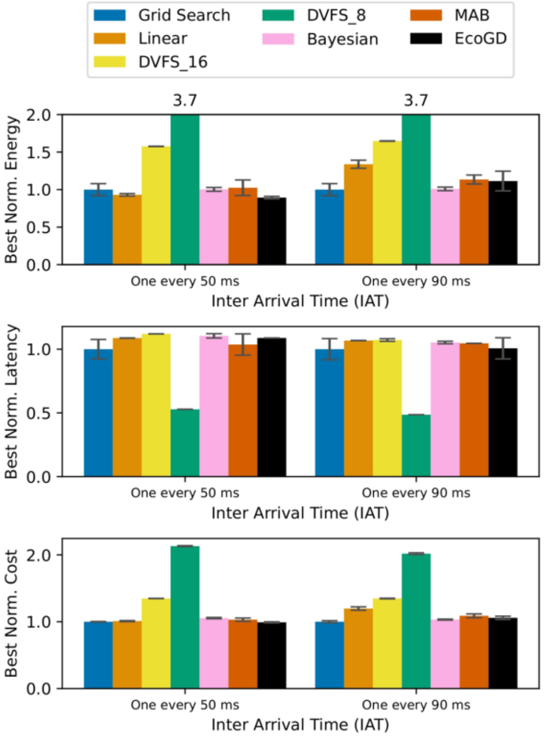
Metrics **after** convergence

Grid Search



Metrics **during** convergence (overhead)

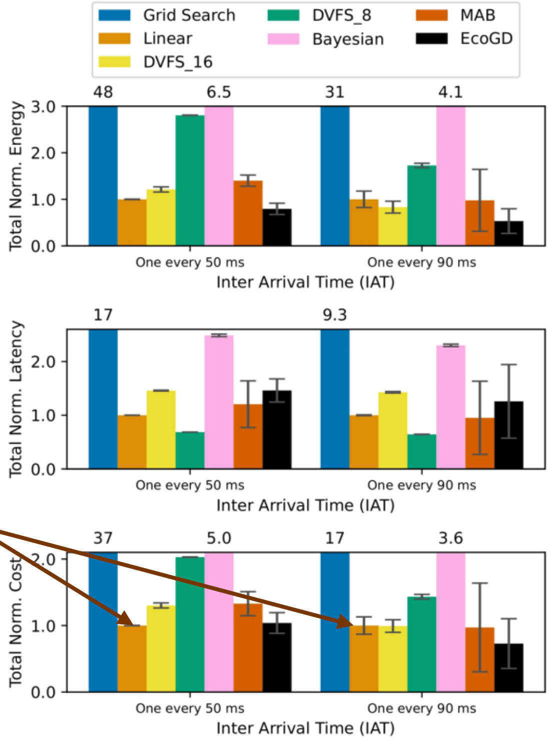
Experimental Results: Fixed Loads



Metrics **after** convergence

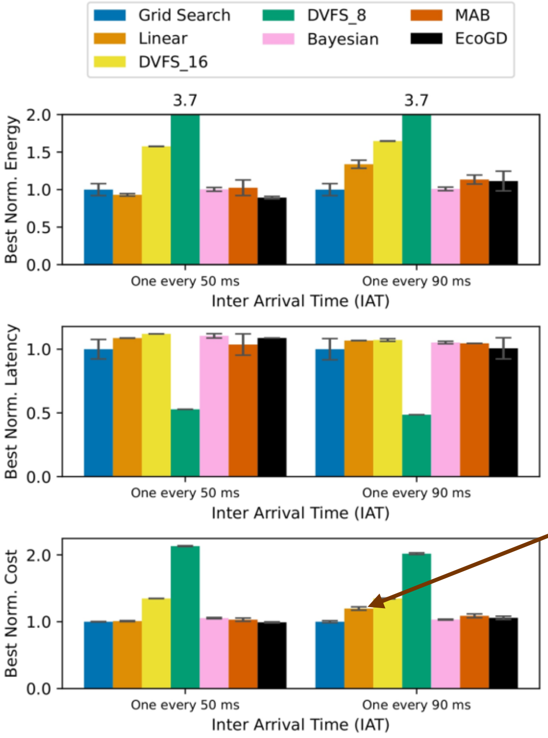
Linear Search

Low overhead but suboptimal



Metrics **during** convergence (overhead)

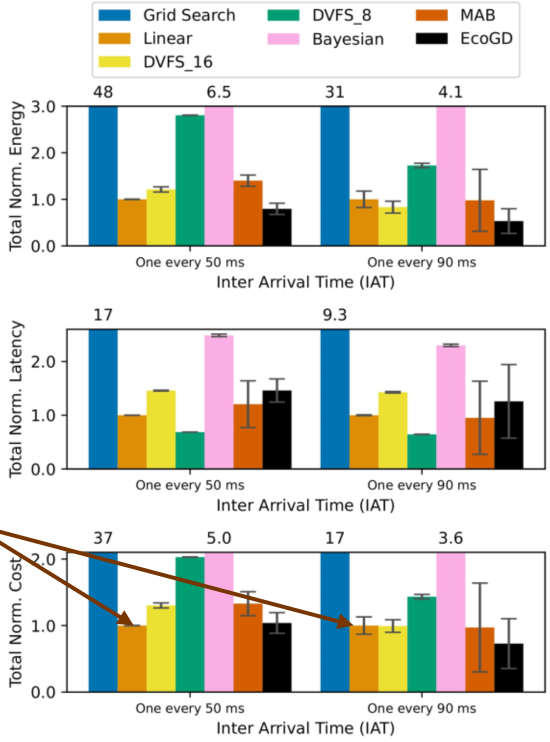
Experimental Results: Fixed Loads



Metrics **after** convergence

Linear Search

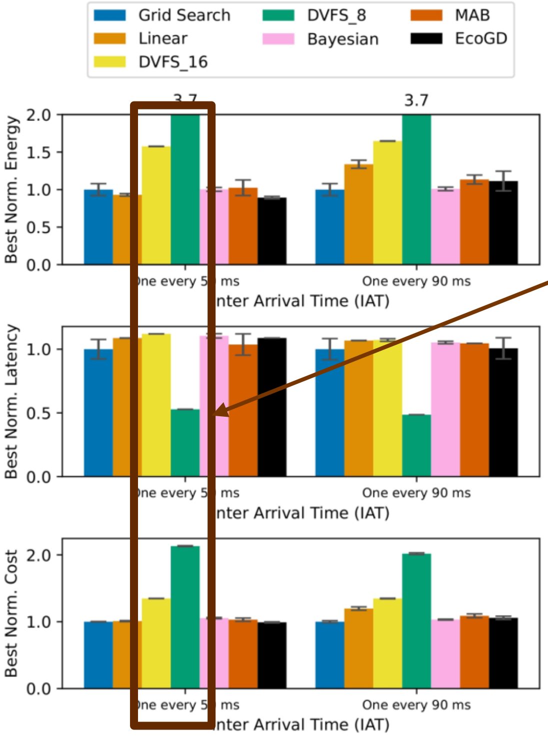
Low overhead but suboptimal



Metrics **during** convergence (overhead)

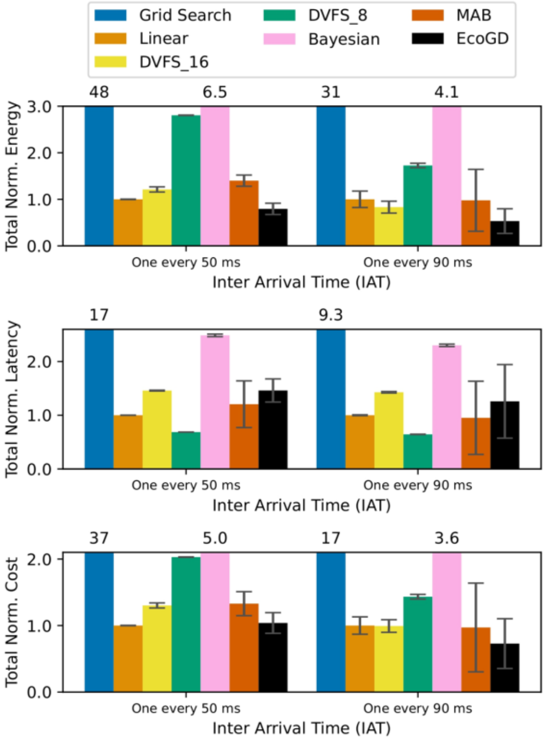
Experimental Results: Fixed Loads

DVFS



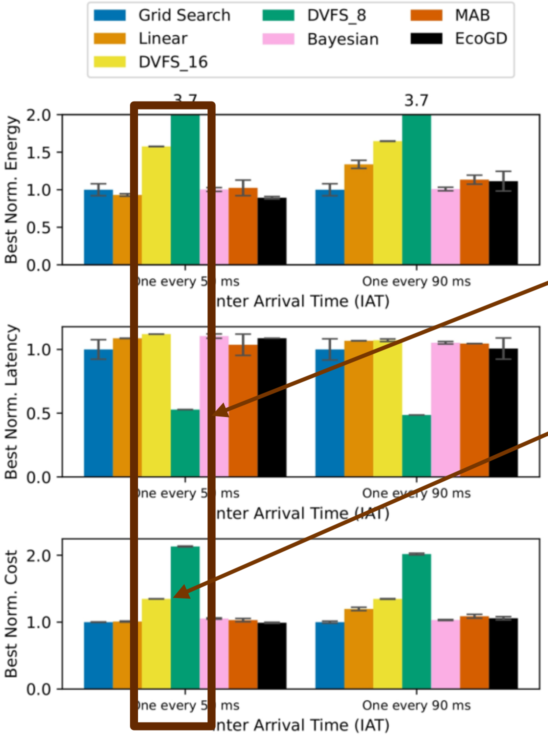
BS 16 w.r.t BS 8
Lower energy
Higher latency, cost

Metrics **after** convergence



Metrics **during** convergence (overhead)

Experimental Results: Fixed Loads

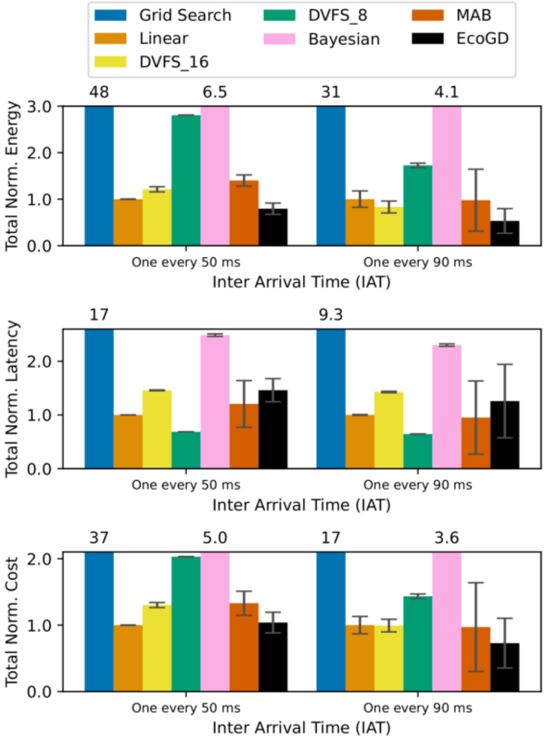


Metrics **after** convergence

DVFS

BS 16 w.r.t BS 8
Lower energy
Higher latency, cost

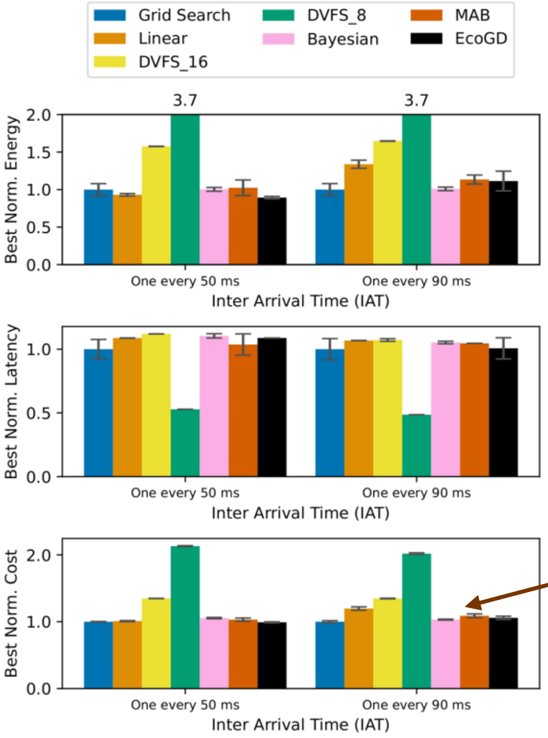
Both BS 8, 16 suboptimal



Metrics **during** convergence (overhead)

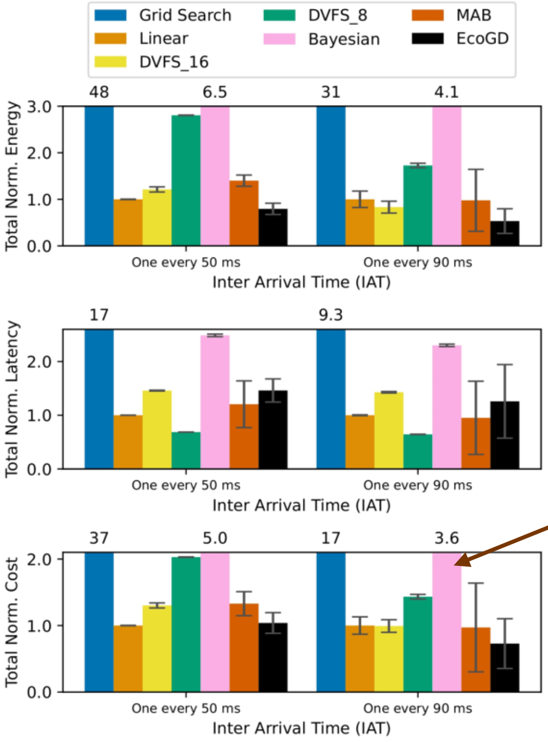
Experimental Results: Fixed Loads

Bayesian



~Optimal

Metrics **<u>after</u>** convergence

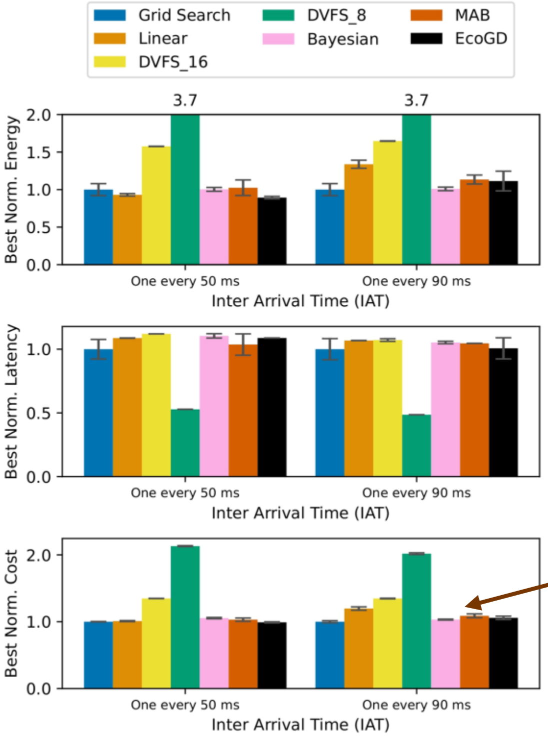


High overhead

Metrics **<u>during</u>** convergence (overhead)

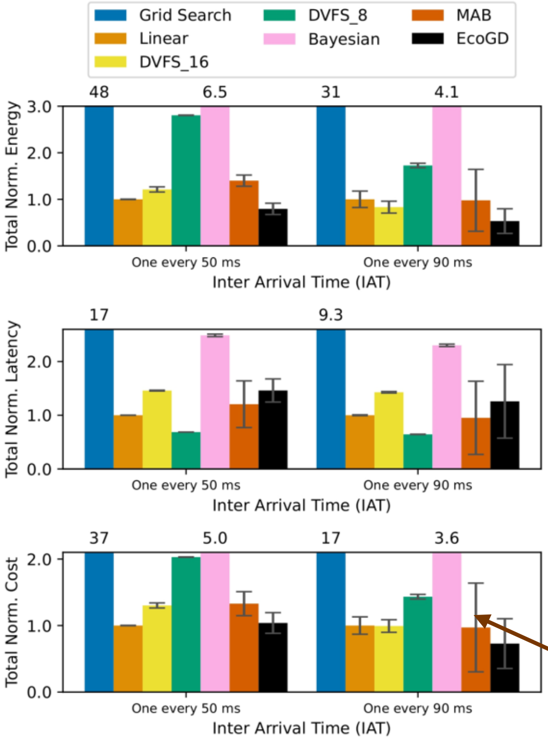
Experimental Results: Fixed Loads

MAB



~Optimal

Metrics **after** convergence

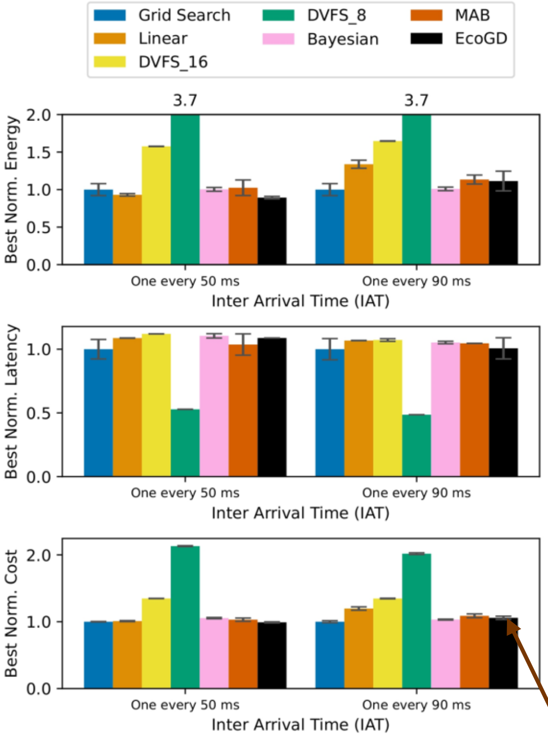


High variance

Metrics **during** convergence (overhead)

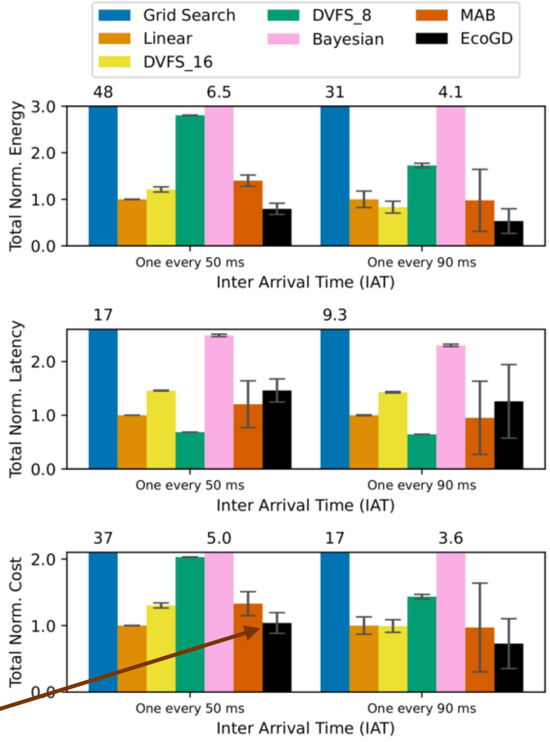
Experimental Results: Fixed Loads

EcoGD



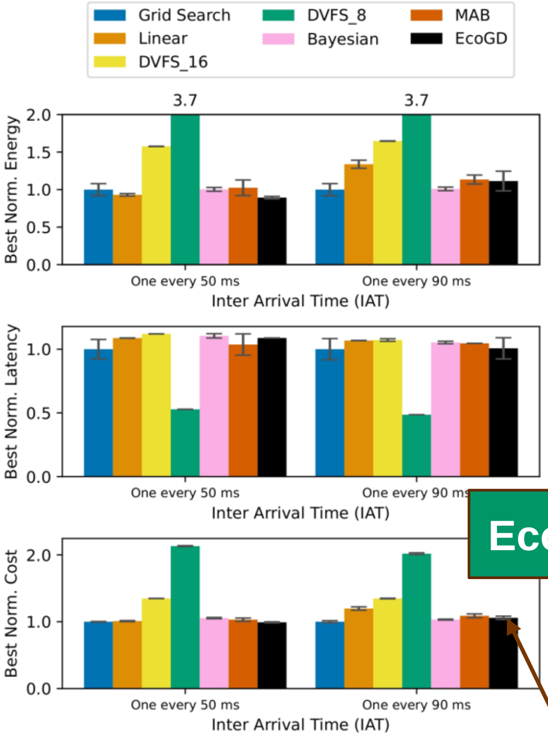
Metrics **after** convergence

~ Optimal and low overhead



Metrics **during** convergence (overhead)

Experimental Results: Fixed Loads

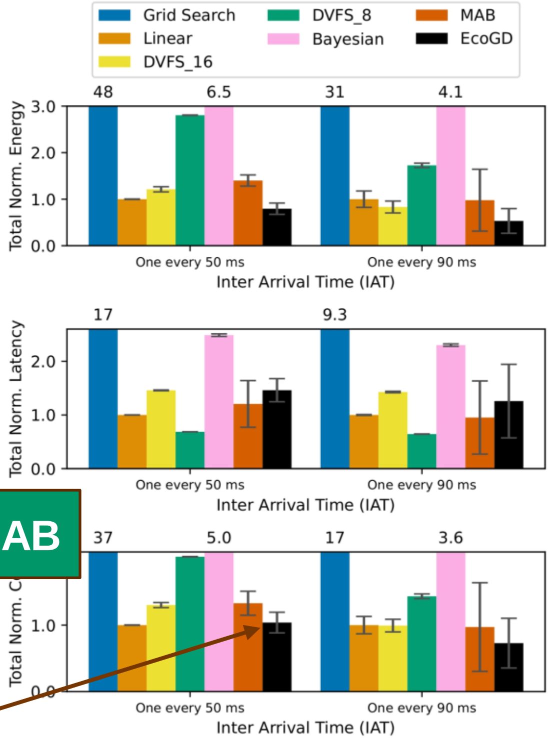


Metrics **after** convergence

EcoGD

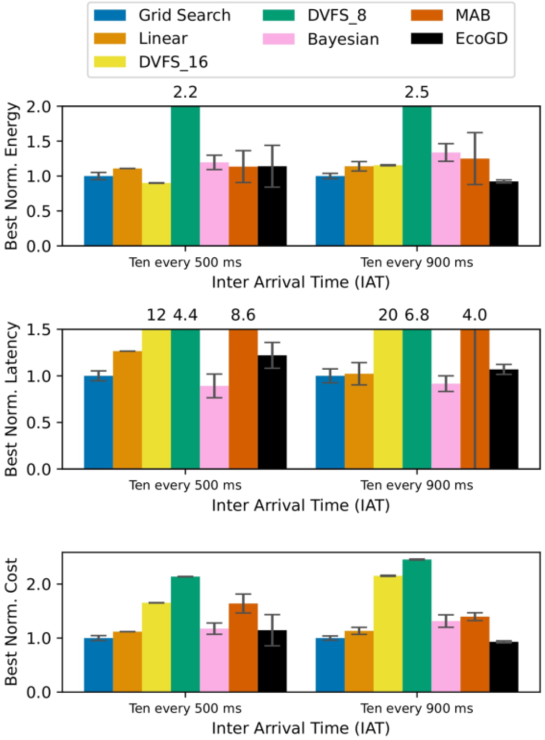
EcoGD better than MAB

~ Optimal and low overhead

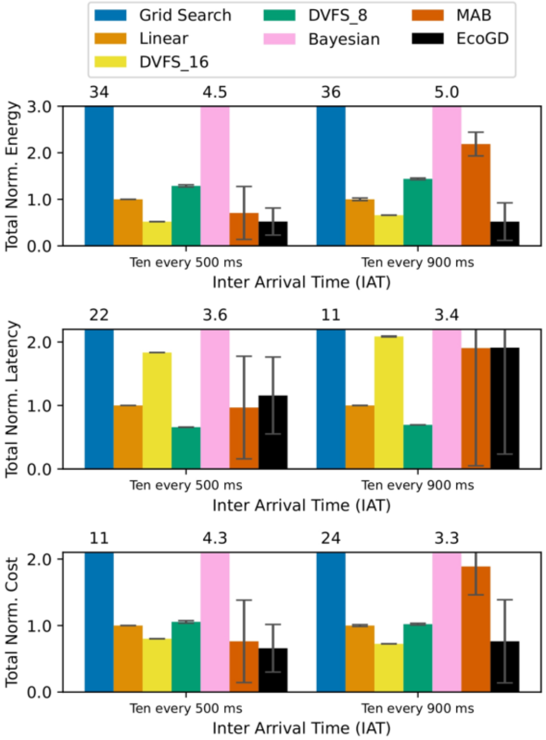


Metrics **during** convergence (overhead)

Experimental Results: Bursty Loads

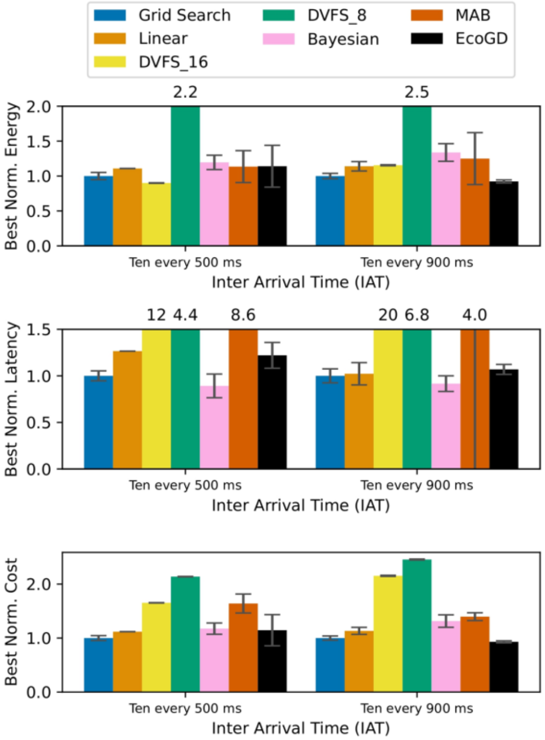


Metrics after convergence



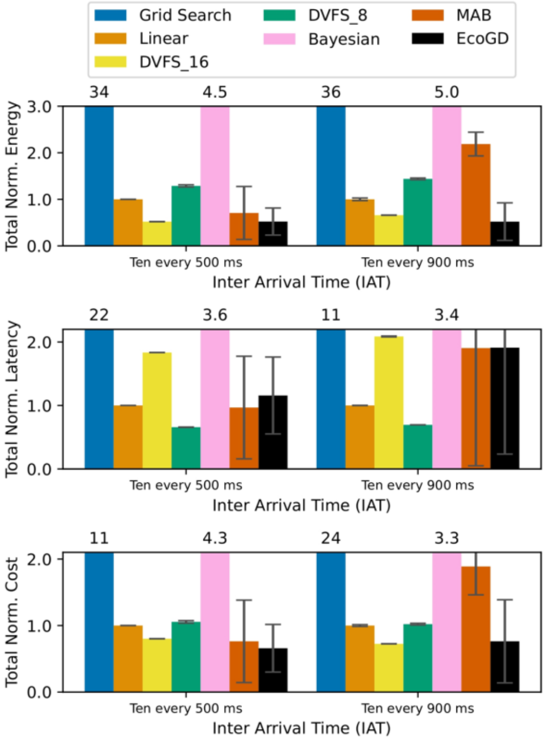
Metrics during convergence (overhead)

Experimental Results: Bursty Loads



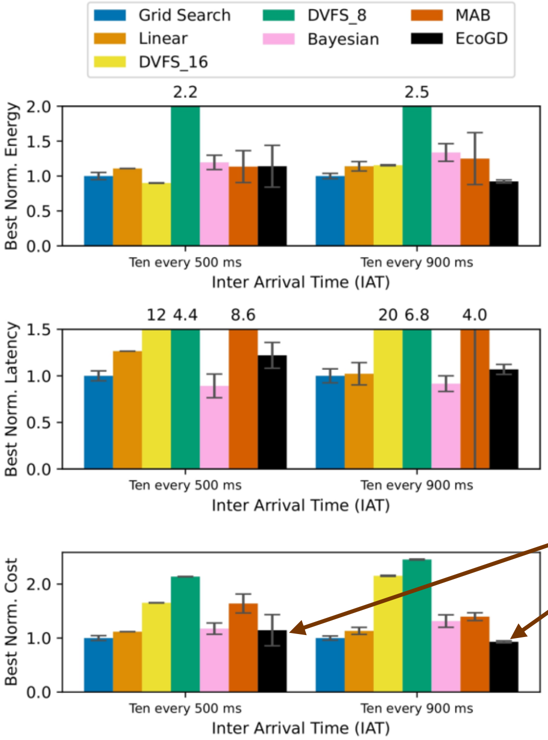
Metrics after convergence

Similar Observations



Metrics during convergence (overhead)

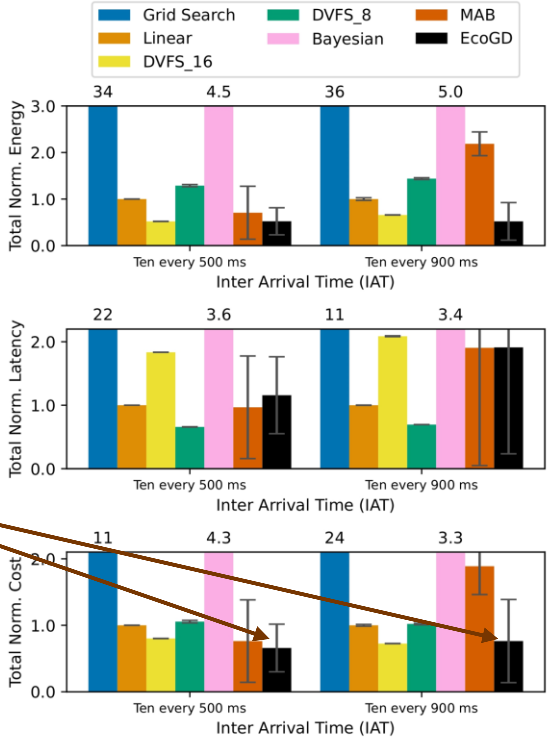
Experimental Results: Bursty Loads



Metrics after convergence

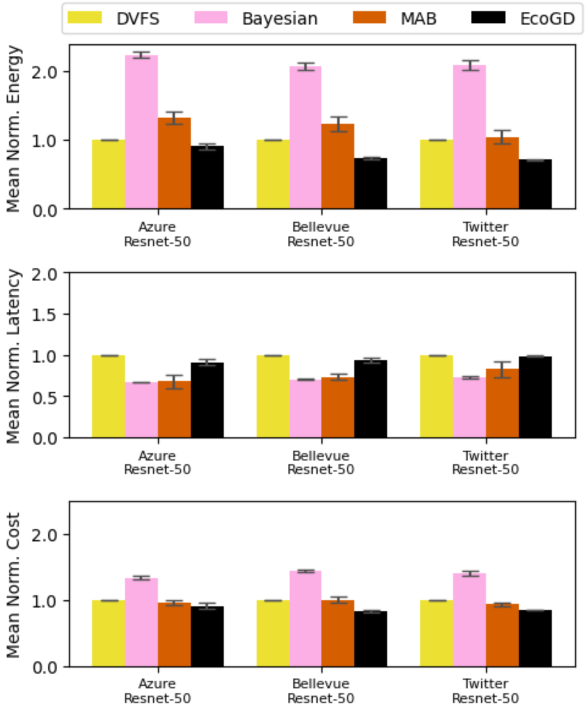
Similar Observations

EcoGD good in both convergence and overhead

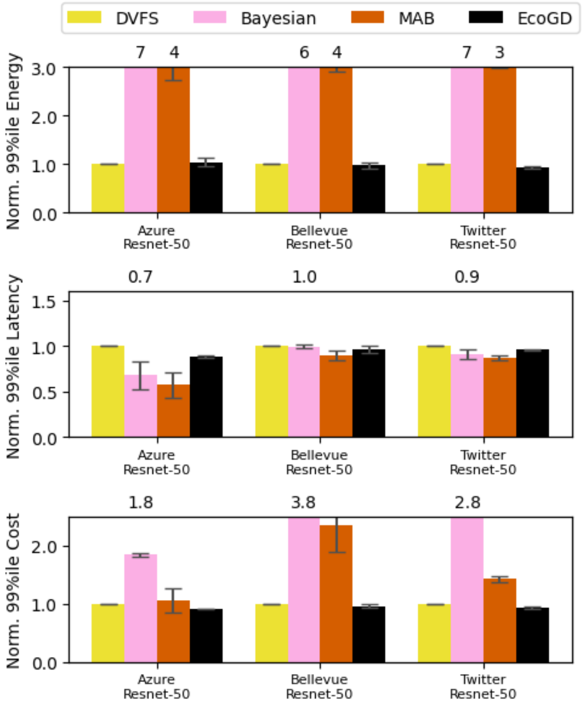


Metrics during convergence (overhead)

Experimental Results: Real-World Traces

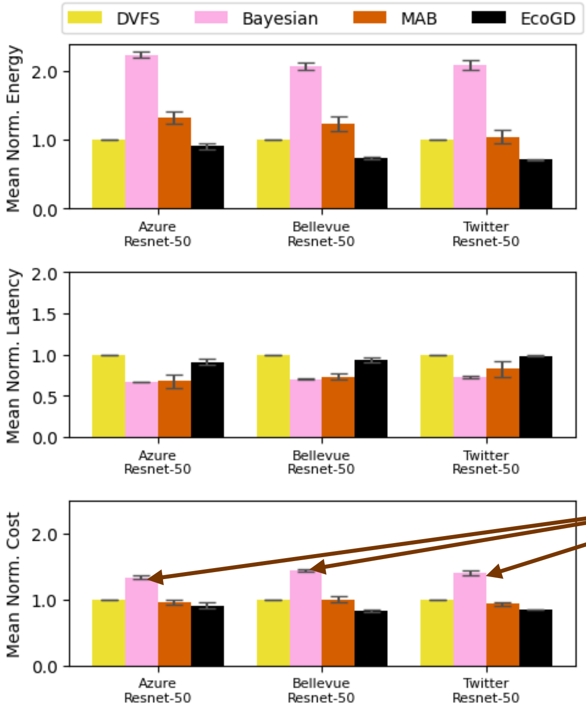


Metrics after convergence



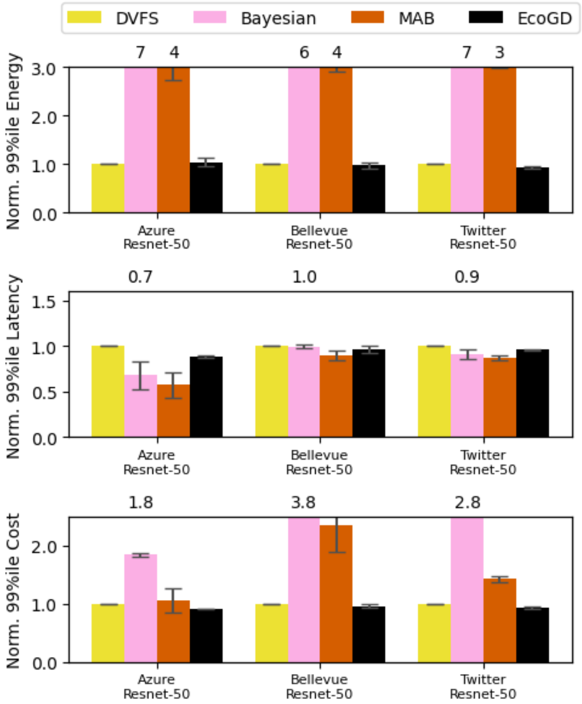
Tail metrics

Experimental Results: Real-World Traces



Bayesian

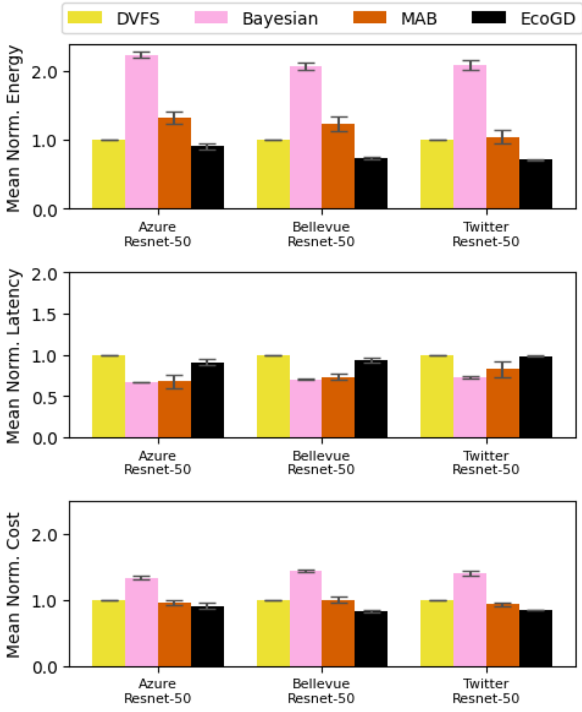
Not Optimal



Metrics after convergence

Tail metrics

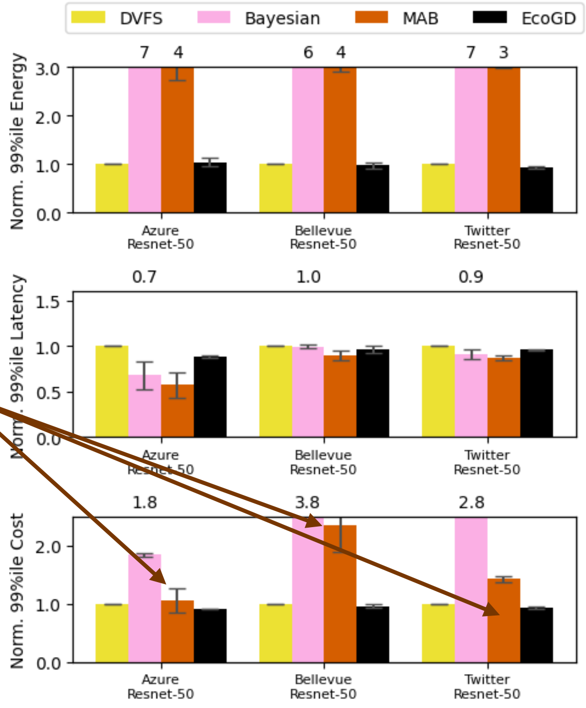
Experimental Results: Real-World Traces



Metrics after convergence

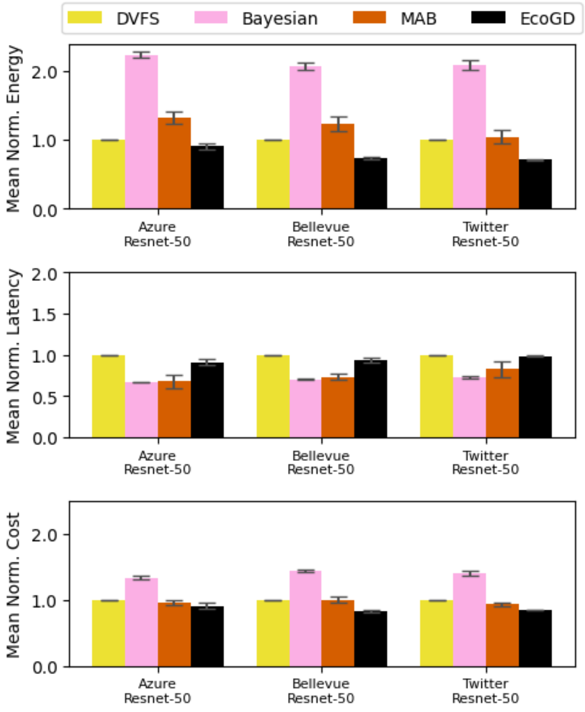
MAB

Higher tail metrics



Tail metrics

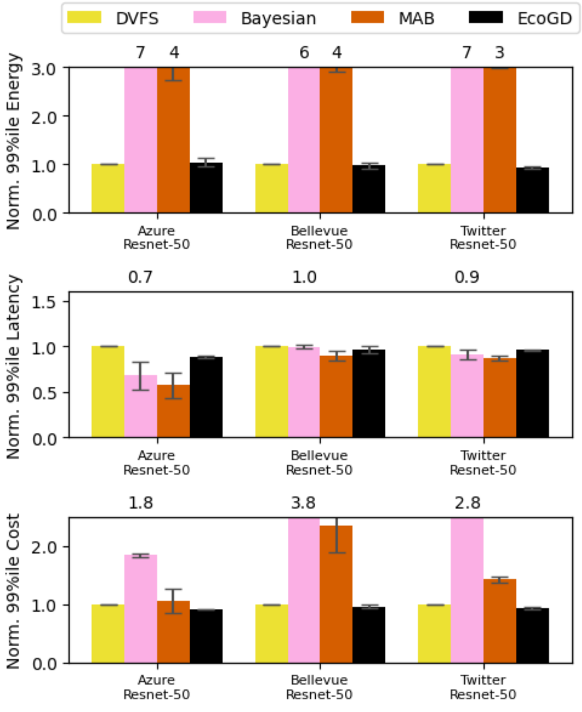
Experimental Results: Real-World Traces



Metrics after convergence

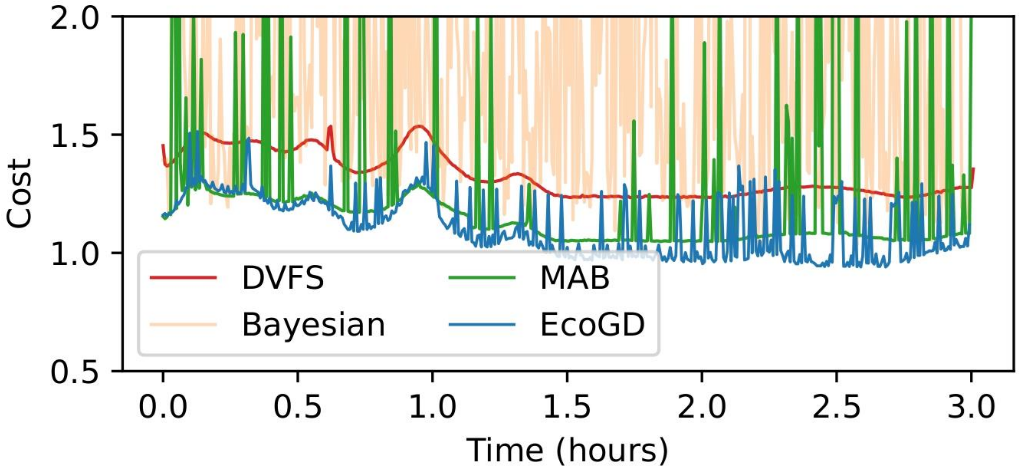
EcoGD

EcoGD gives the best performance



Tail metrics

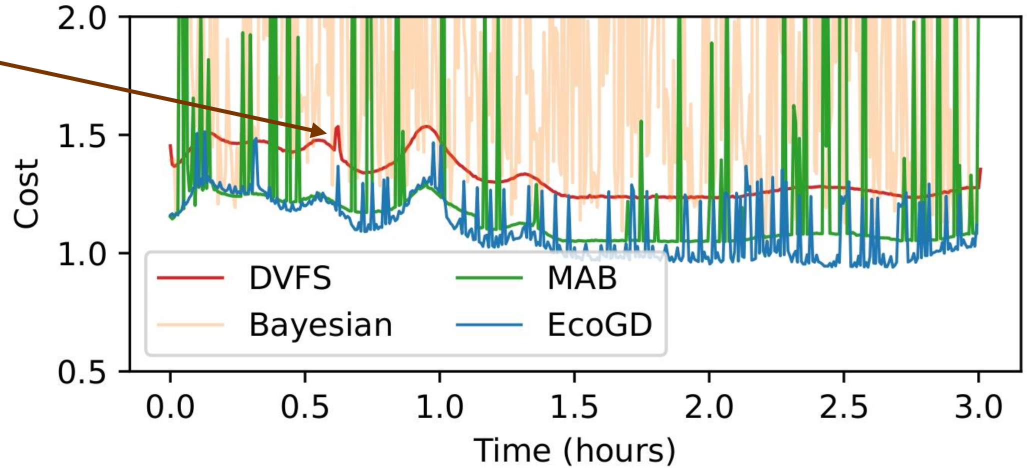
Experimental Results: Real-World Traces



Time series of cost of algorithms while running Bellevue trace with Resnet50

Experimental Results: Real-World Traces

DVFS > MAB, EcoGD

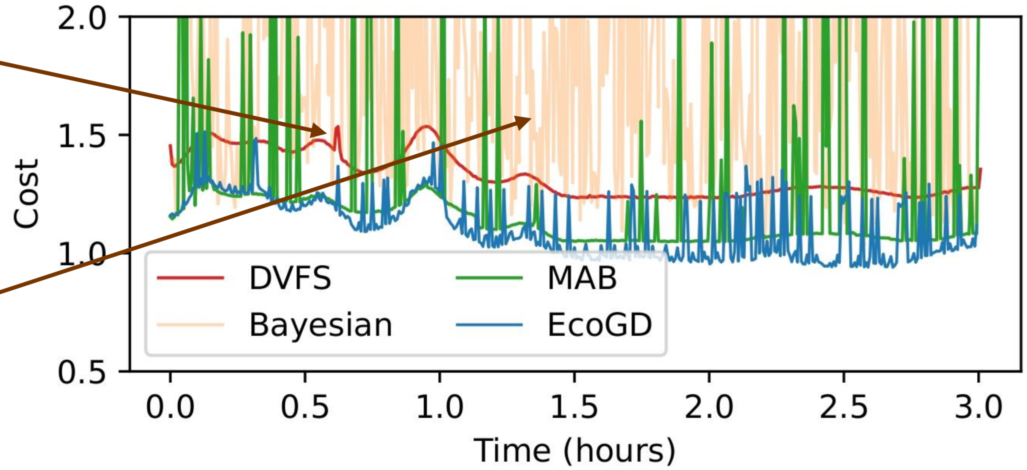


Time series of cost of algorithms while running Bellevue trace with Resnet50

Experimental Results: Real-World Traces

DVFS > MAB, EcoGD

Bayesian – high and keeps jumping

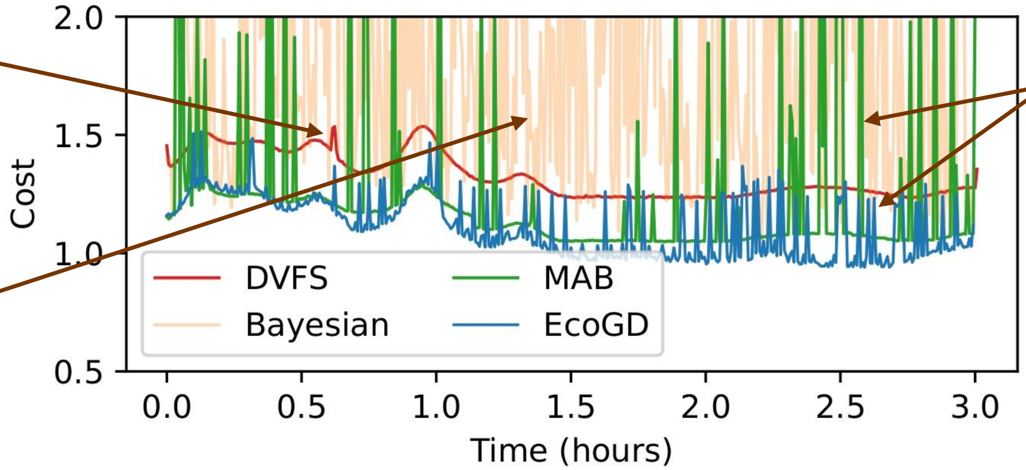


Time series of cost of algorithms while running Bellevue trace with Resnet50

Experimental Results: Real-World Traces

DVFS > MAB, EcoGD

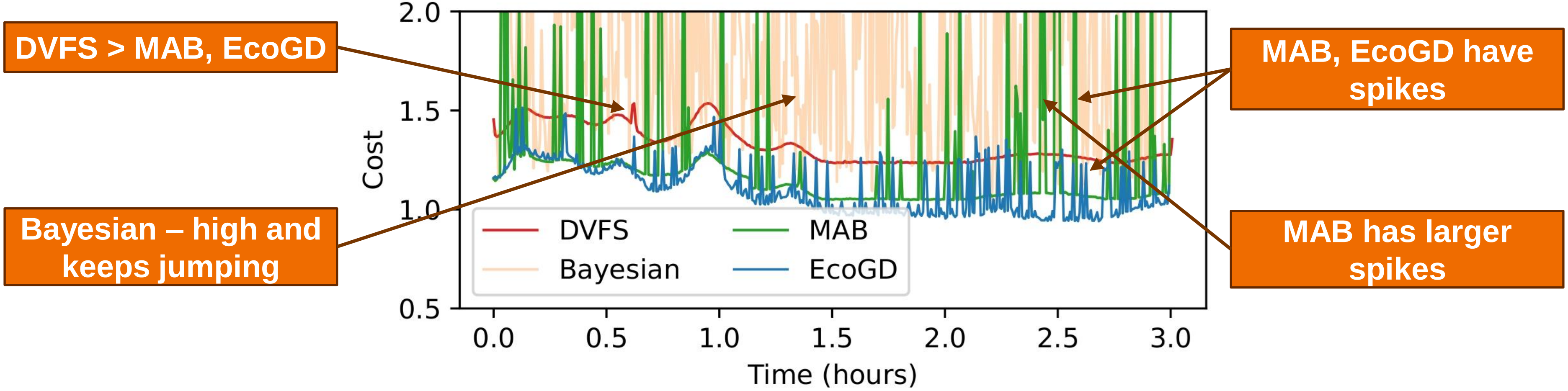
Bayesian – high and keeps jumping



MAB, EcoGD have spikes

Time series of cost of algorithms while running Bellevue trace with Resnet50

Experimental Results: Real-World Traces



Time series of cost of algorithms while running Bellevue trace with Resnet50

Conclusion

- **Trade-offs** exist between energy consumption and latency
- **Time varying workloads** need dynamic and adaptive solutions
- **EcoEdgeInfer** - easy to use framework for tuning Frequencies and Batch Size
- **EcoGD** - algorithm to optimize for latency and energy
  - Mean cost reduction as much as 55% (19% average reduction)
  - Tail cost reduction as much as 90% (36% average reduction)
- **Wide applicability** with easy to use python APIs and opensourced code

- Future directions include
  - **More devices** with accelerators like TPUs and server grade hardware
  - **Mobile devices** with Tensor core – iPhones, Pixels etc
  - **Custom objectives** like SLOs, cost-effectiveness
  - **Load balancing** for heterogenous multi-device setup

Thanks for your attention

Any Questions?

GitHub Repo

Key Points-

- Trade-offs: energy/latency
- Time varying workloads
- EcoEdgeInfer framework
- EcoGD algorithm
- Wide Applicability