AutoMon: Automatic Distributed Monitoring for Arbitrary Multivariate Functions

HADAR SIVAN
TECHNION Israel Institute of Technology

MOSHE GABEL
UNIVERSITY OF TORONTO

ASSAF SCHUSTER
TECHNION Israel Institute of Technology

SIGMOD 2022

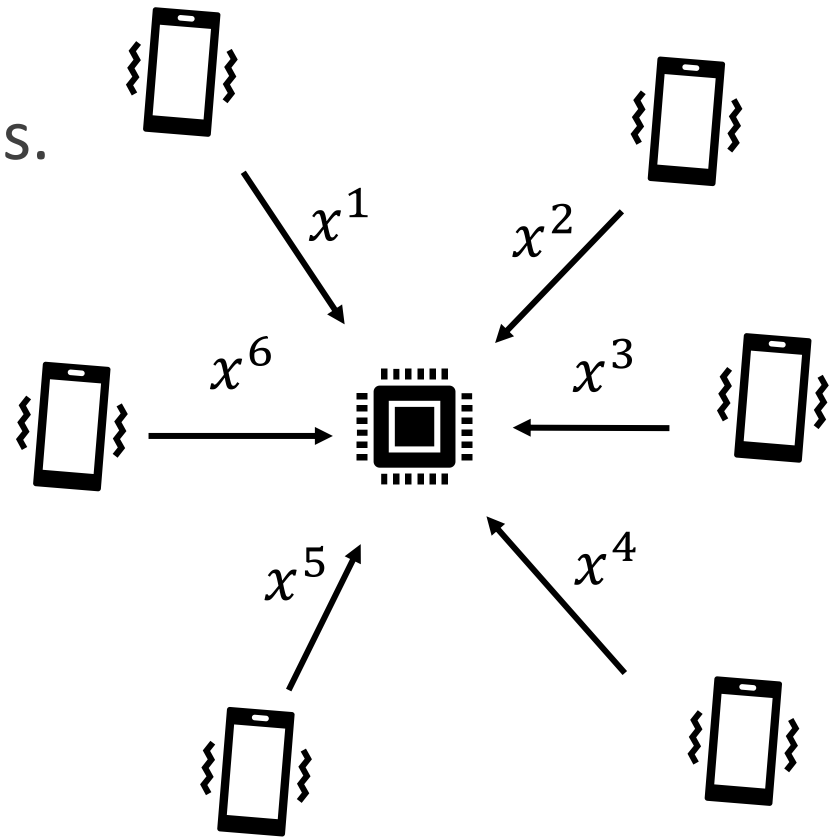
Hypothetical Startup: Earthquake Detector App

Warn users seconds before earthquake:

- Continuously collect accelerometer statistics.
  - $x^1, \dots, x^k$ dynamic statistics vectors of size d
- Aggregate data
  - $\bar{x} = \frac{1}{k}\sum_{j=1}^{k} x^j$ global average of local vectors
- Run through a neural network:

$$f_{nn}(\bar{x}) = W_3 \cdot \tanh\left(W_2 \cdot \tanh\left(W_1 \cdot \bar{x} + b_1\right) + b_2\right) + b_3$$

  - W_i, b_i network weights
  - tanh activation



Computing f_{nn} in Centralized Settings

- Straightforward in centralized settings:

```
def f_nn(x, W1, b1, W2, b2, W3, b3):
    return W3 @ tanh(W2 @ tanh(W1 @ x + b1) + b2) + b3
```

- **Data is not static!**
  - What will we do when $x^1 \dots x^k$ change?

f_{nn} Over Geo-distributed Streams

- Can't centralize all updates
  - Limited battery, bandwidth
  - Communication costs x1000s more energy than computation!
    [Anastasi et al., Ad hoc networks, 2009; Pottie et al., CACM, 2000]
  - Could overwhelm local datacenter
- $f_{nn}(x^j)$ does not reflect $f_{nn}(\bar{x})$
- Need a **communication-efficient** algorithm for f_{nn}

AutoMon

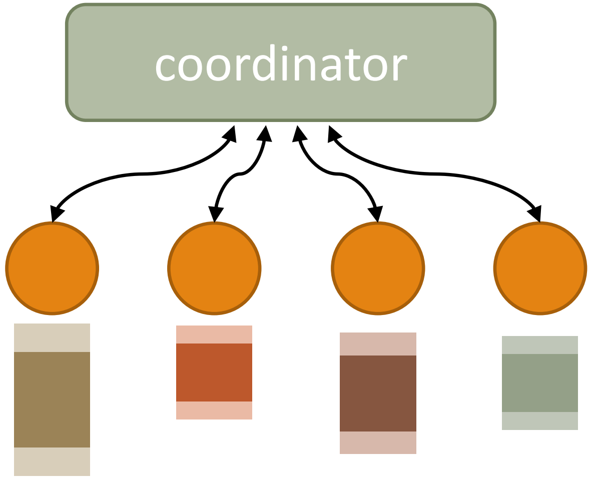
The first approach for monitoring that is automatic and general:

- Given **source code** for computing f from data…
- …**automatically** implements a communication-efficient distributed approximation protocol for $f(\bar{x})$

- Reduces communication by up to ×50
- Works on complicated, non-convex f
- No need for math

How AutoMon Works?

- **Setting**
  - n nodes with data streams
  - Nodes communicate with **coordinator**



- **AutoMon's Input:**
  - Source code for computing f from $\bar{x}$
  - Desired approximation error ϵ

```python
def f_inner_product(x):
    n = x.shape[0] // 2
    u = x[:n]
    v = x[n:]
    return u @ v
```

AutoMon Protocol Overview

The geometric monitoring protocol

[Alfassi et al., ICDE, 2021; Gabel et al., SIGKDD, 2017; Keren et al., TKDE, 2012]

1. Sync
2. Monitor
3. Report

- Collect data, $x_0 = \frac{1}{n}\sum x^j$, and compute **current approximation** $\boldsymbol{f_0 = f(x_0)}$
- **Intuition:** send data and update f_0 only if x^j changed "enough"

AutoMon Protocol Overview

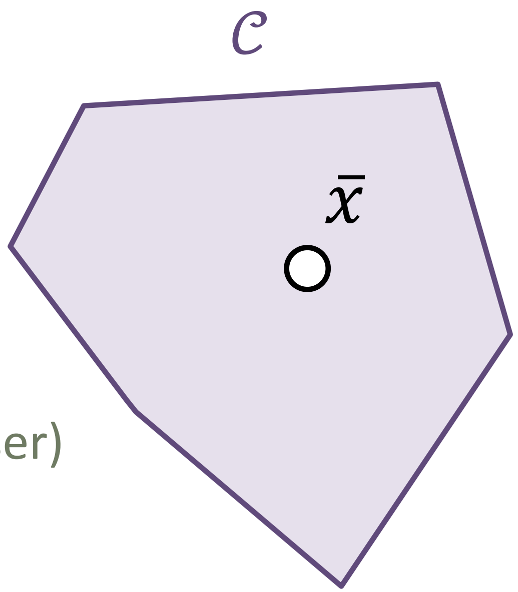
- Need constraint that guarantees $|f(\bar{x}) - f_0| \leq \epsilon$.
- Find a **safe zone**: convex set $\mathcal{C}$ such that:

$$\bar{x} \in \mathcal{C} \implies L \leq f(\bar{x}) \leq U$$

lower bound
$L = f_0 - \epsilon$
(determined by user)

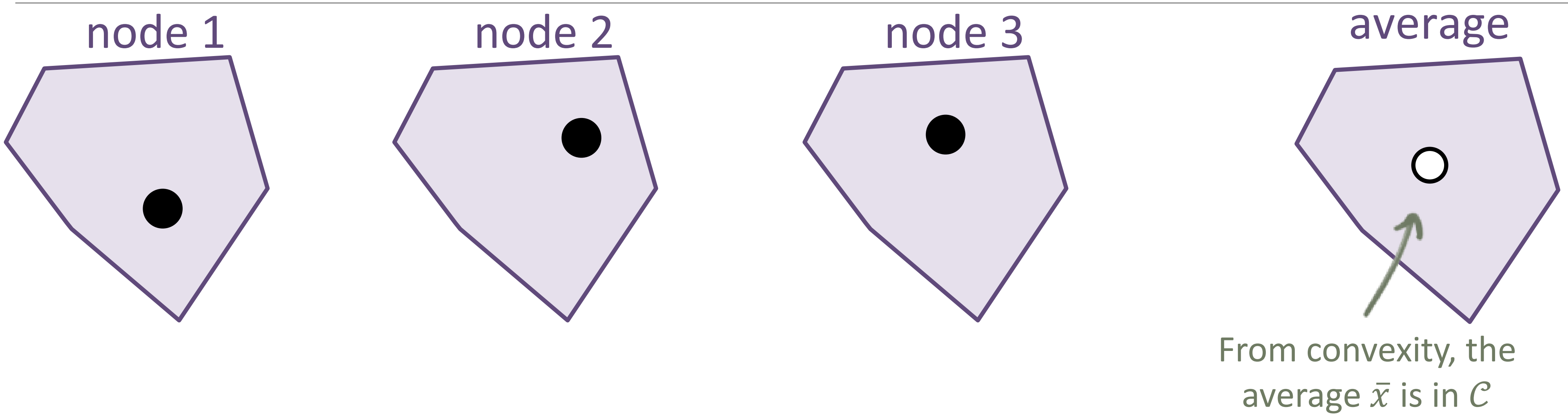
current exact value
(unknown!)

upper bound
$U = f_0 + \epsilon$
(determined by user)



1. Sync
2. Monitor
3. Report

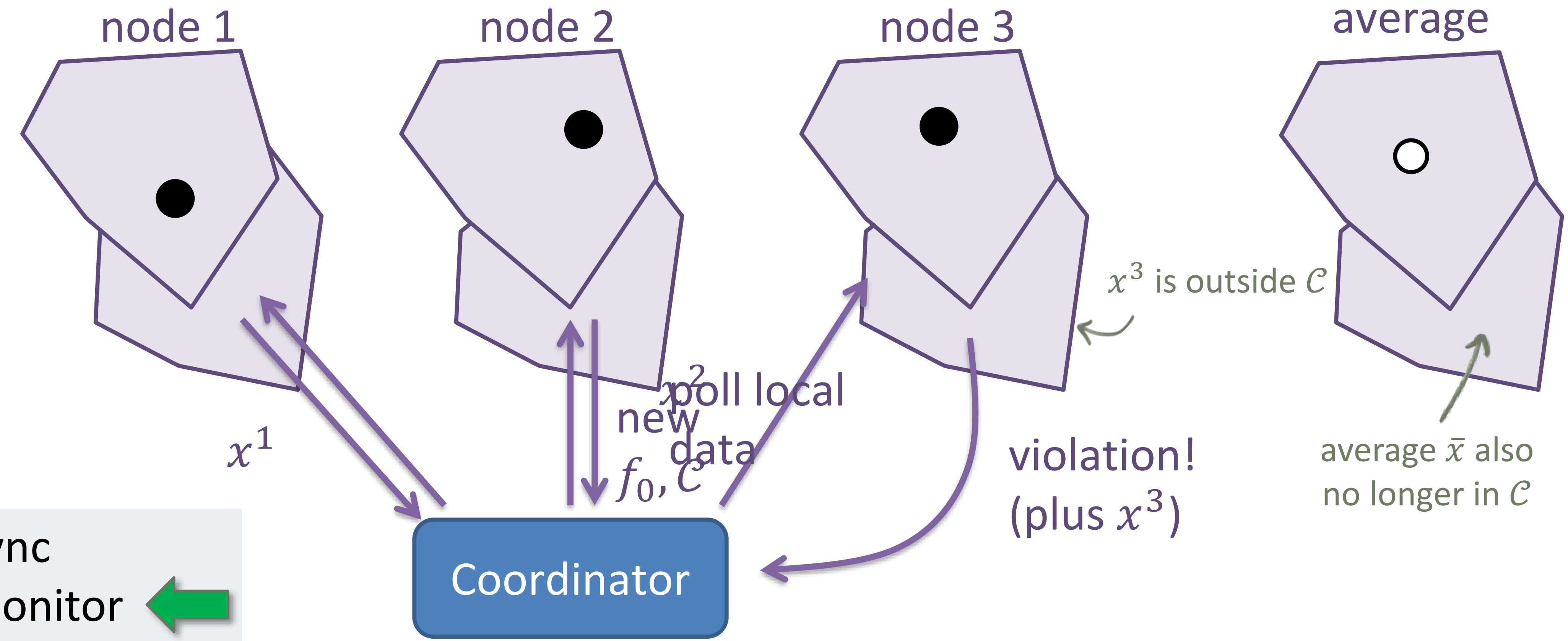
Protocol in Action



1. Sync
2. Monitor
3. Report

Coordinator

Protocol in Action



1. Sync
2. Monitor
3. Report

The Core of AutoMon

- During sync, $\mathcal{C}$ must be tailored to f such that
  $$\bar{x} \in \mathcal{C} \implies L \leq f(\bar{x}) \leq U$$

- The magic of AutoMon:
  finding a good $\mathcal{C}$ **automatically**, using only f's **source code**.

1. Sync ←
2. Monitor
3. Report

AutoMon's Safe Zone

1. DC DECOMPOSITION → 2. SAFE ZONE

- Convex difference:
  $$f(x) = \check{g}(x) - \check{h}(x)$$
  convex

[Lazerson et al., 2018]

$$\mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \check{g}(x) \leq U \\ \check{h}(x) \leq f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$$

convex sets

- Concave difference:
  $$f(x) = \hat{g}(x) - \hat{h}(x)$$
  concave

$$\mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \hat{h}(x) \geq f(x_0) + \nabla f(x_0)^T (x - x_0) - U \\ \hat{g}(x) \geq L \end{array} \right\}$$

Finding DC decomposition automatically!

Automatic DC Decomposition (ADCD)

$f(x)$

↓ Automatic Differentiation

H

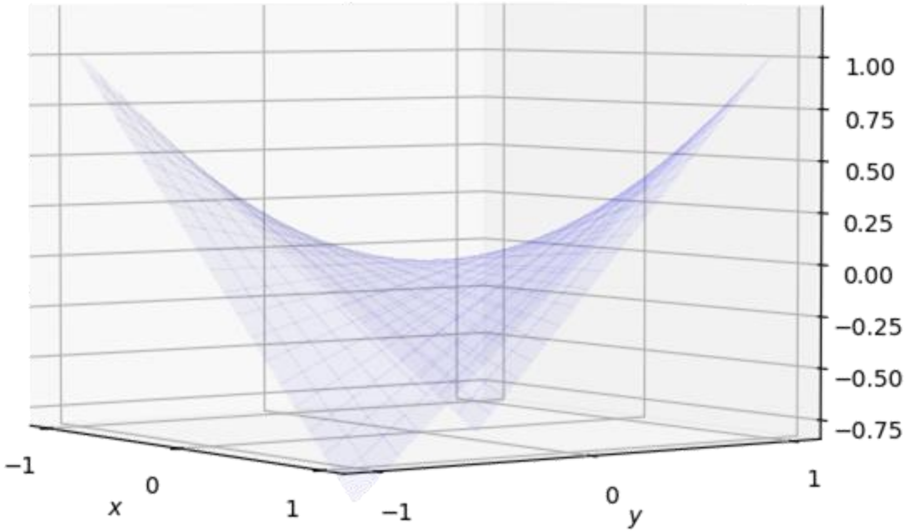
↓

ADCD-X

1. Numerical optimization → λ_X (e**X**treme eigenvalue)
2. Quadratic component: $h = -\frac{1}{2}\lambda_X \|x - x_0\|^2$ → $f + h = g$
3. DC decomposition: $f = g - h$

ADCD-X

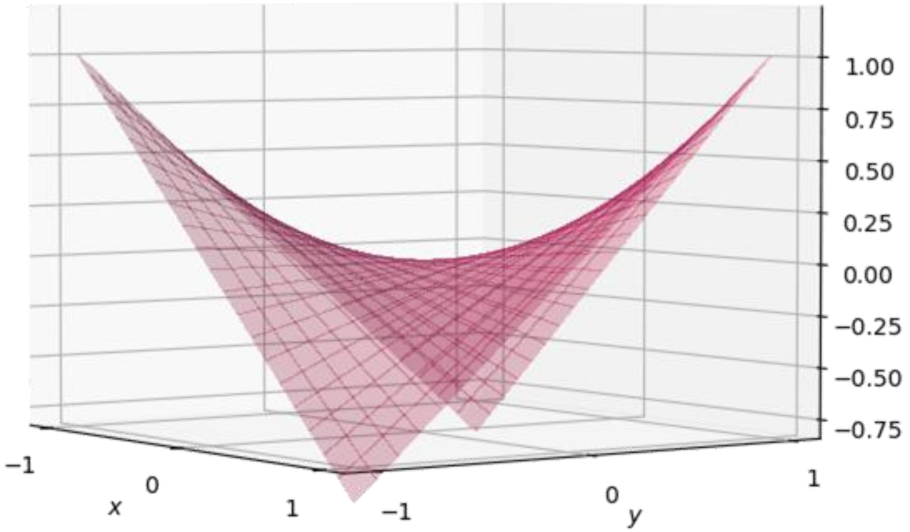
$f \rightarrow \breve{g}$



f : the function we monitor

ADCD-X

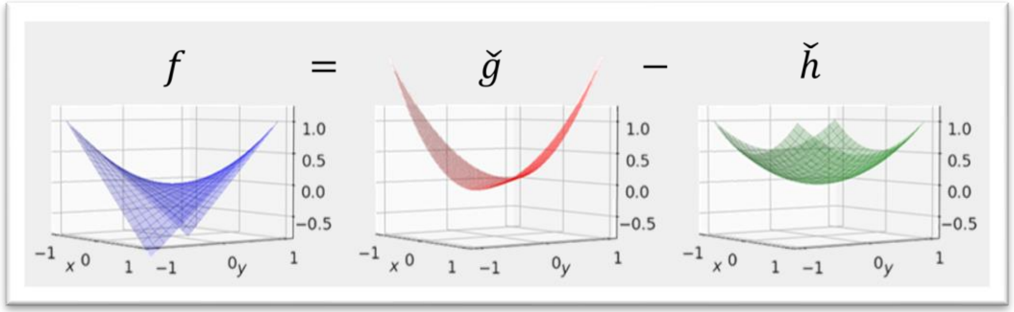
$f \rightarrow \breve{g}$



f : the function we monitor

$\breve{h}$: quadratic component, based on λ_X

$\breve{g} = f + \breve{h}$

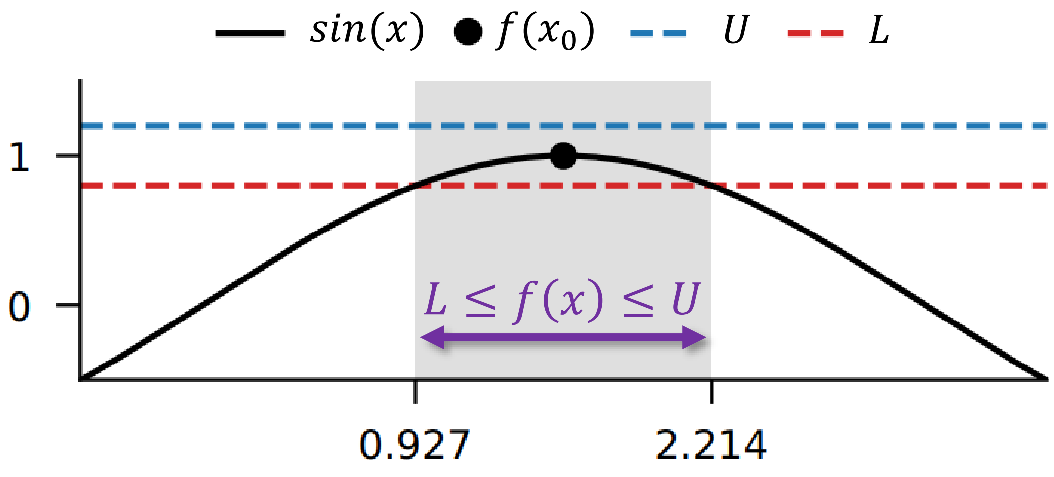


Example: ADCD to Safe-Zone

$$f(x) = \check{g}(x) - \check{h}(x) \quad \Rightarrow \quad \mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \check{g}(x) \leq U \\ \check{h}(x) \leq f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

- $f = \sin(x)$
  - Gray area: $L \leq f(x) \leq U$

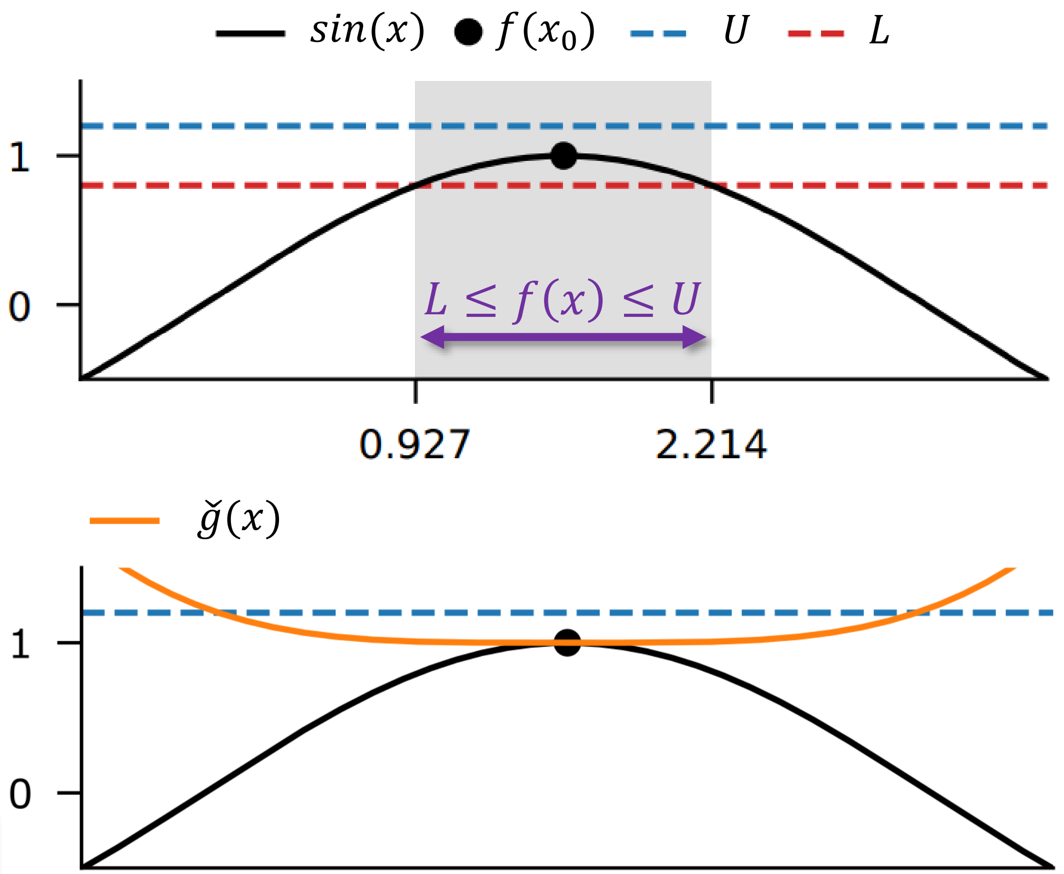


$$f(x) = \breve{g}(x) - \breve{h}(x) \Rightarrow \mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \breve{g}(x) \leq U \\ \breve{h}(x) \leq f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

- $f = \sin(x)$
  - Gray area: $L \leq f(x) \leq U$

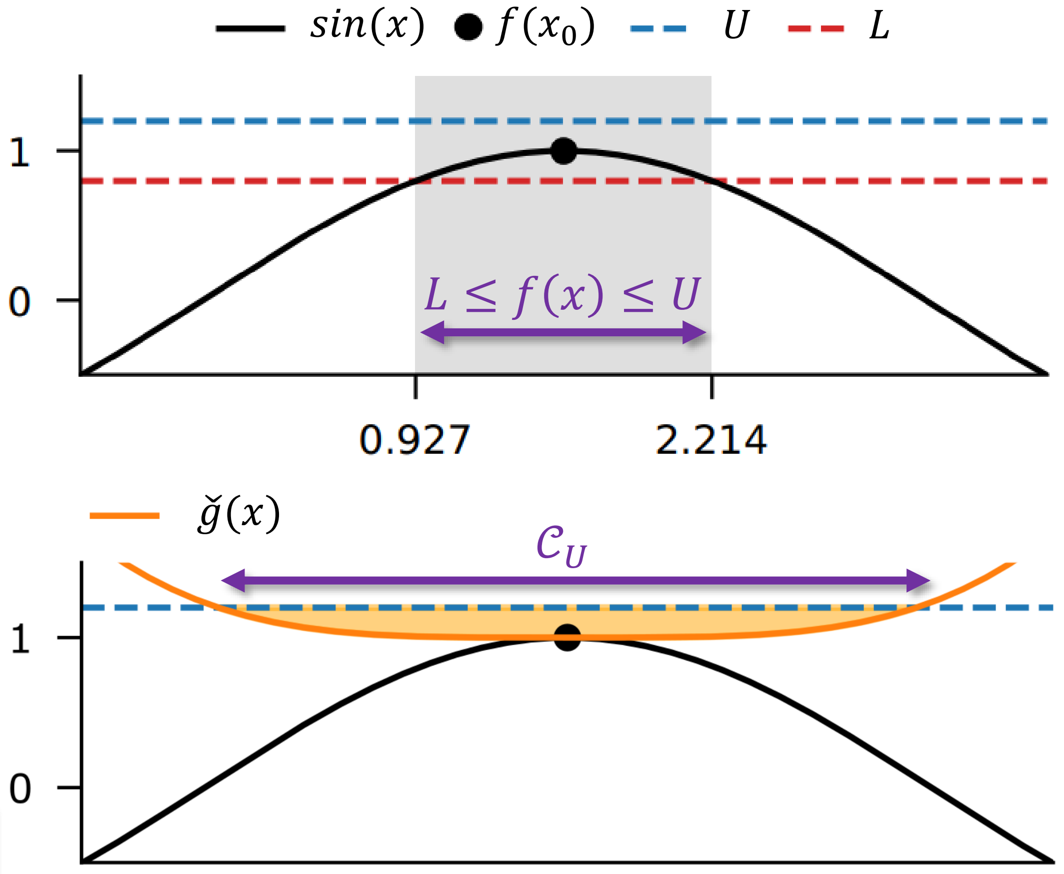
- Start with $\breve{g}(x)$



$$f(x) = \breve{g}(x) - \breve{h}(x) \Rightarrow \mathcal{C} = \left\{ x \,\middle|\, \begin{array}{l} \breve{g}(x) \leq U \\ \breve{h}(x) \leq f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

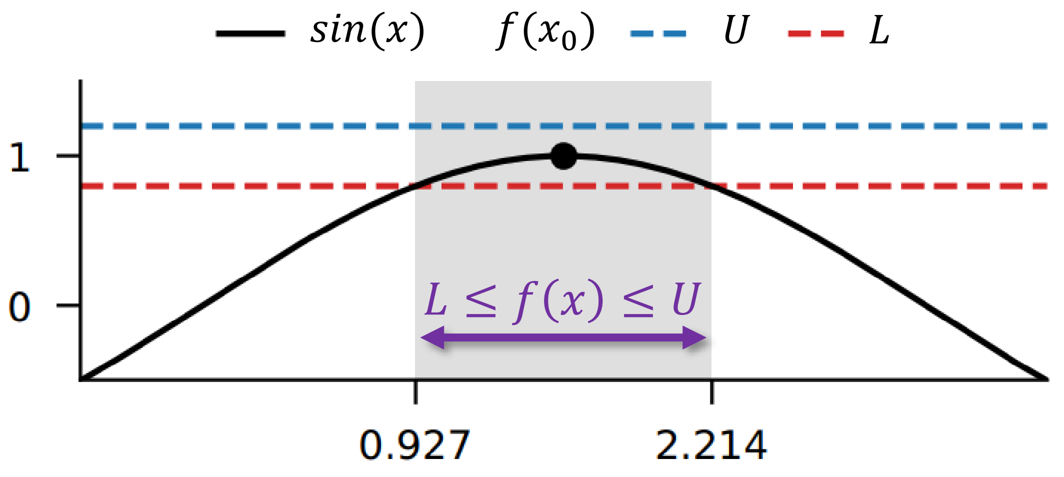
- $f = \sin(x)$
  - Gray area: $L \leq f(x) \leq U$
- Start with $\breve{g}(x)$
- Show area where $\breve{g}(x) \leq U$
  - Partial safe zone $\mathcal{C}_U$



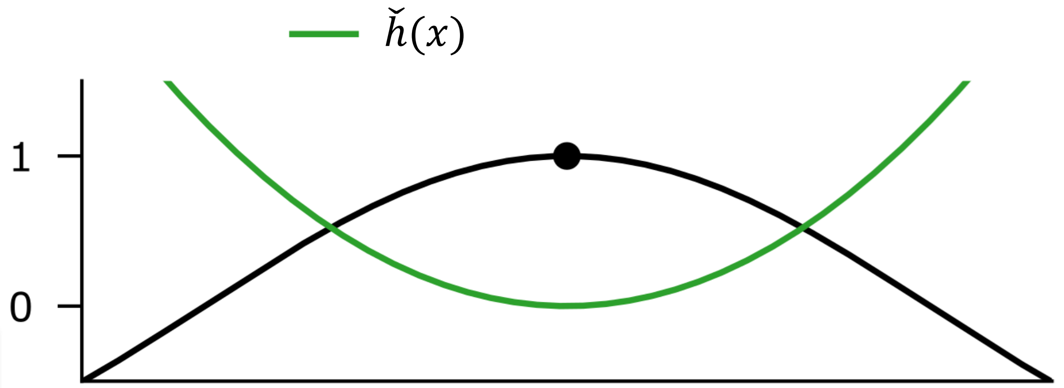
$$f(x) = \breve{g}(x) - \breve{h}(x) \Rightarrow \mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \breve{g}(x) \leq U \\ \breve{h}(x) \leq f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

- $f = \sin(x)$
  - Gray area: $L \le f(x) \le U$



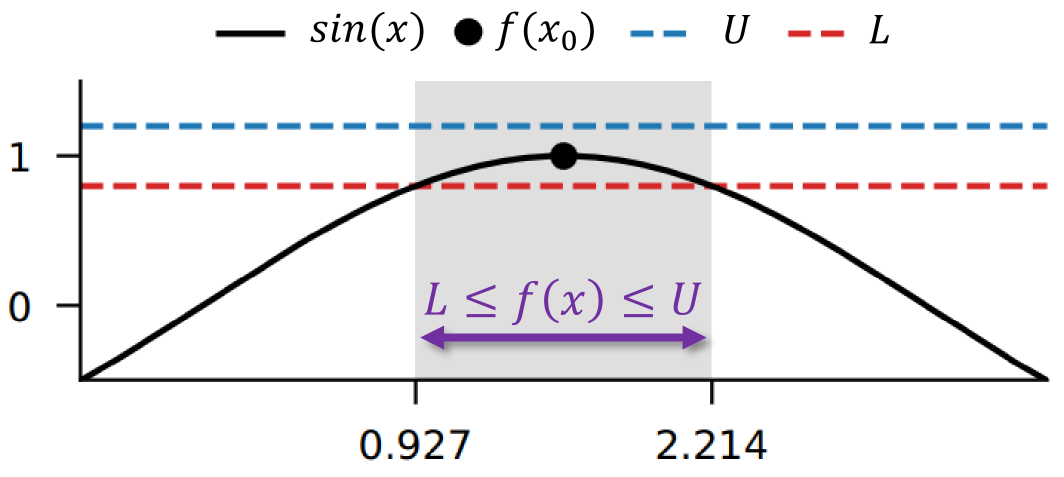
- Now $\check{h}(x)$



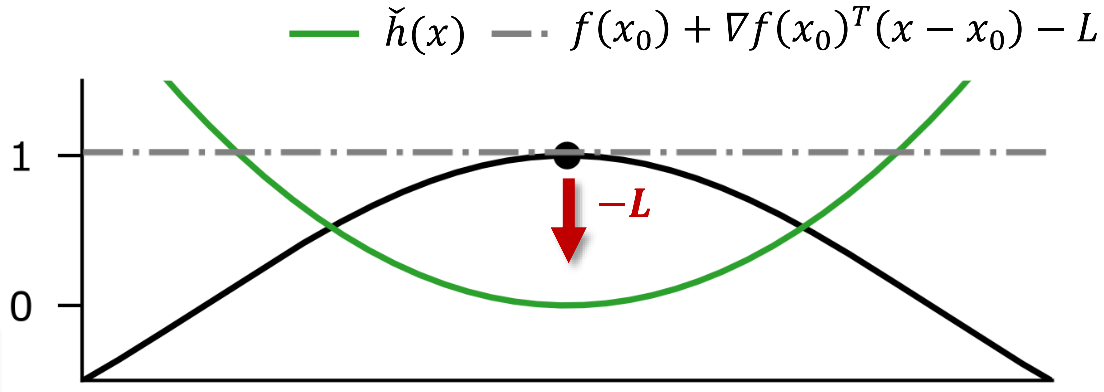
$$f(x) = \check{g}(x) - \check{h}(x) \Rightarrow \mathcal{C} = \left\{ x \,\middle|\, \begin{array}{l} \check{g}(x) \le U \\ \check{h}(x) \le f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

- $f = \sin(x)$
  - Gray area: $L \le f(x) \le U$



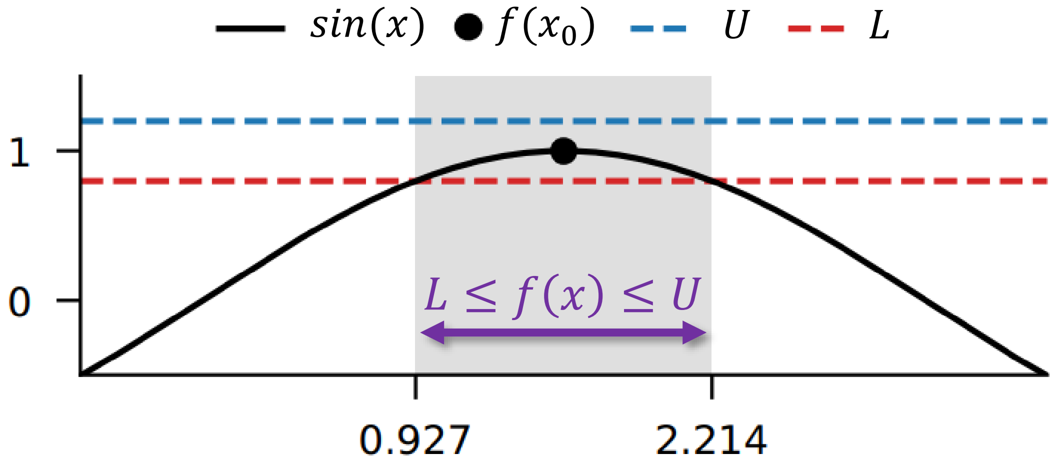
- Now $\check{h}(x)$ and tangent minus L $f(x_0) + \nabla f(x_0)^T (x - x_0) - L$



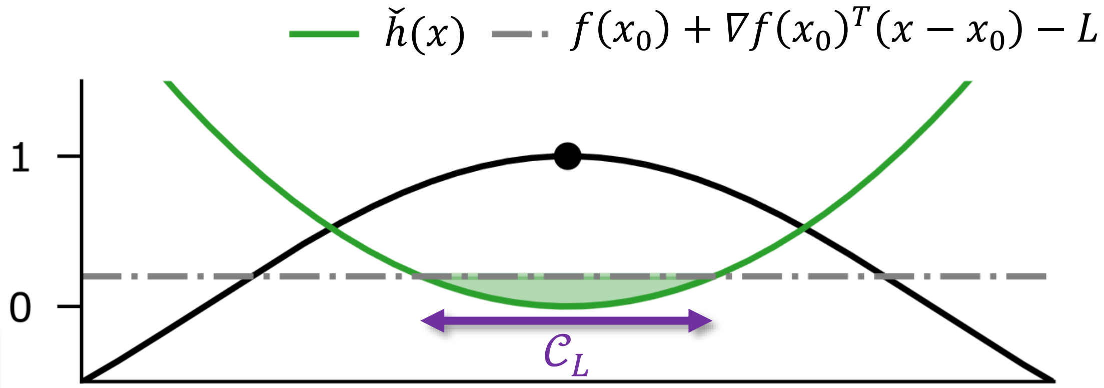
$f(x) = \check{g}(x) - \check{h}(x) \Rightarrow \mathcal{C} = \left\{ x \,\middle|\, \begin{array}{l} \check{g}(x) \le U \\ \check{h}(x) \le f(x_0) + \nabla f(x_0)^T (x - x_0) - L \end{array} \right\}$

Example: ADCD to Safe-Zone

- $f = \sin(x)$
  - Gray area: $L \leq f(x) \leq U$



- Now $\check{h}(x)$ and tangent minus L
  $f(x_0) + \nabla f(x_0)^T(x - x_0) - L$
- The area where $\check{h}(x) \leq$ tangent

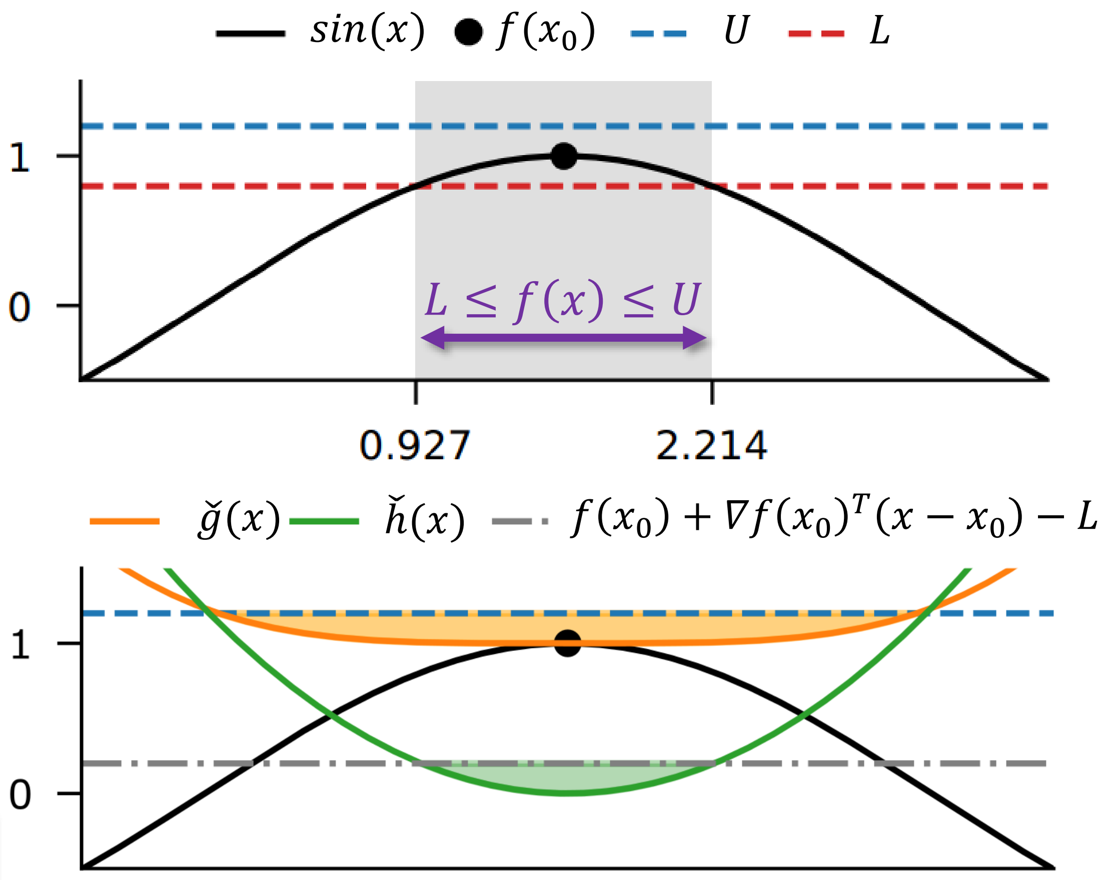


$$f(x) = \breve{g}(x) - \check{h}(x) \Rightarrow \mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \breve{g}(x) \leq U \\ \check{h}(x) \leq f(x_0) + \nabla f(x_0)^T(x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

- $f = \sin(x)$
  - Gray area: $L \leq f(x) \leq U$

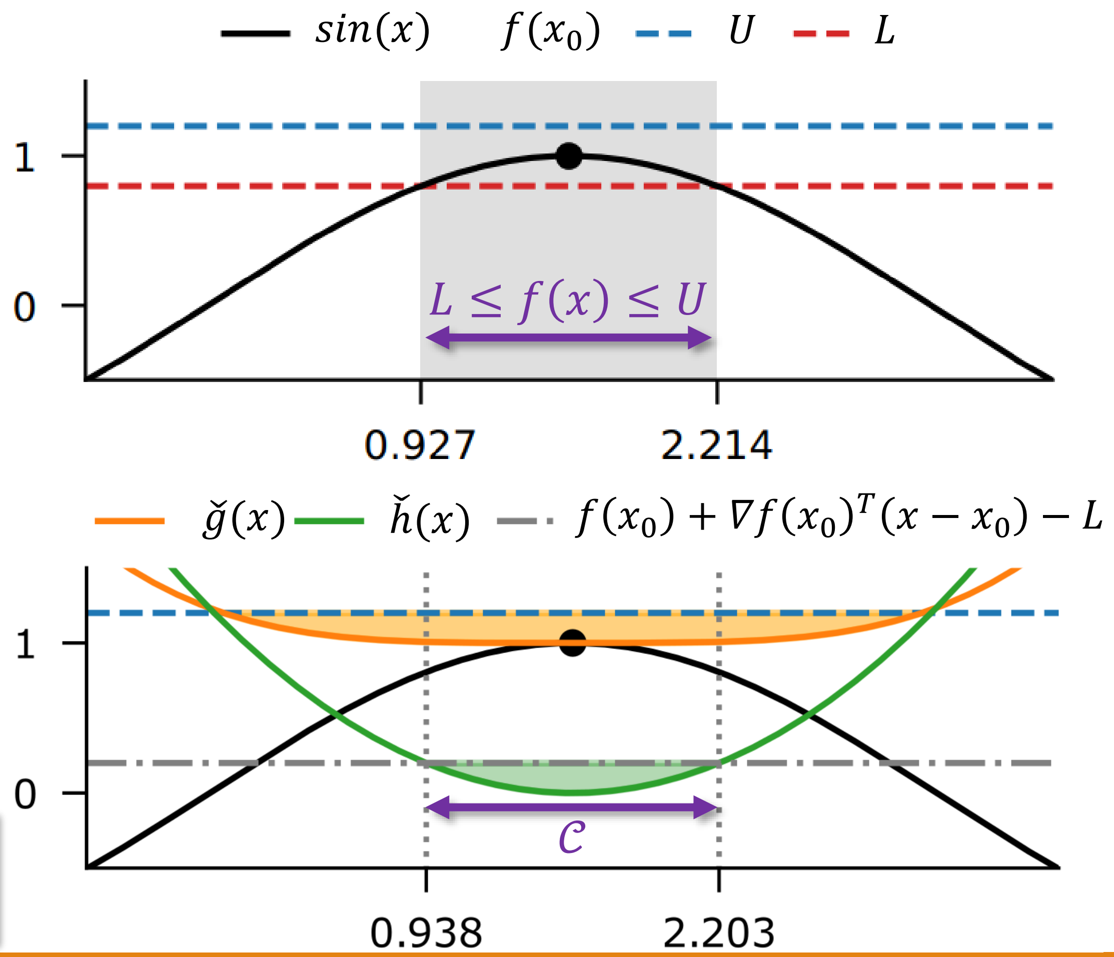
- Both areas



$$f(x) = \breve{g}(x) - \breve{h}(x) \Rightarrow \mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \breve{g}(x) \leq U \\ \breve{h}(x) \leq f(x_0) + \nabla f(x_0)^T(x - x_0) - L \end{array} \right\}$$

Example: ADCD to Safe-Zone

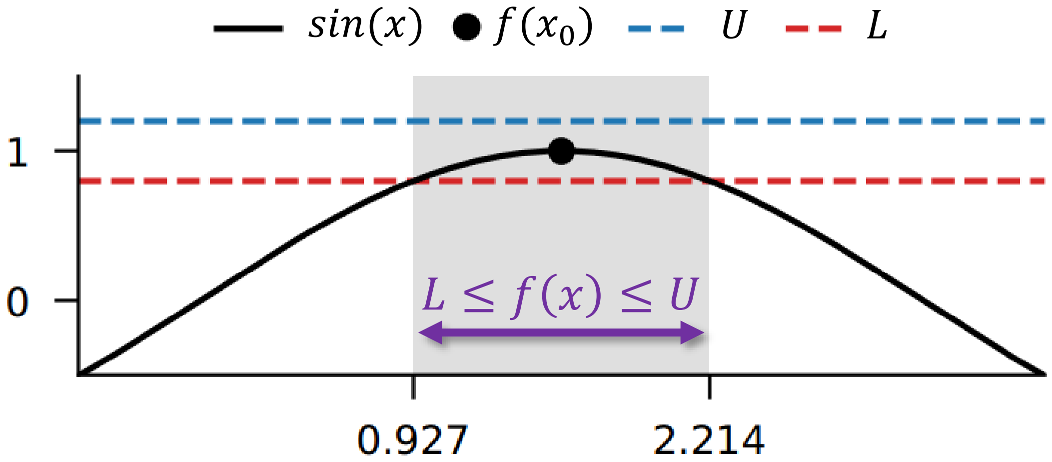
- $f = \sin(x)$
  - Gray area: $L \leq f(x) \leq U$



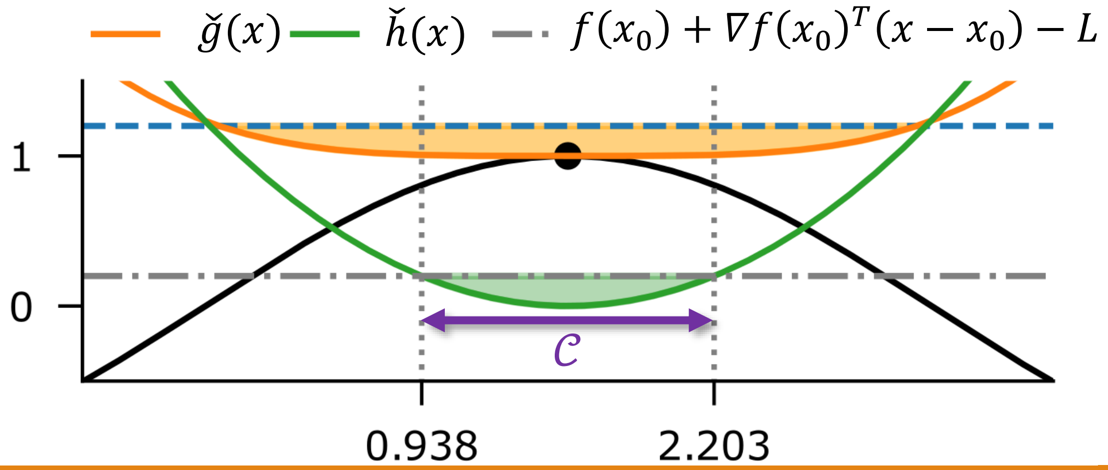
- Both areas
- Final $\mathcal{C}$ is intersection $\mathcal{C}_L \cap \mathcal{C}_U$

$$f(x) = \breve{g}(x) - \breve{h}(x) \Rightarrow \mathcal{C} = \left\{ x \;\middle|\; \begin{array}{l} \breve{g}(x) \leq U \\ \breve{h}(x) \leq f(x_0) + \nabla f(x_0)^T(x - x_0) - L \end{array} \right\}$$

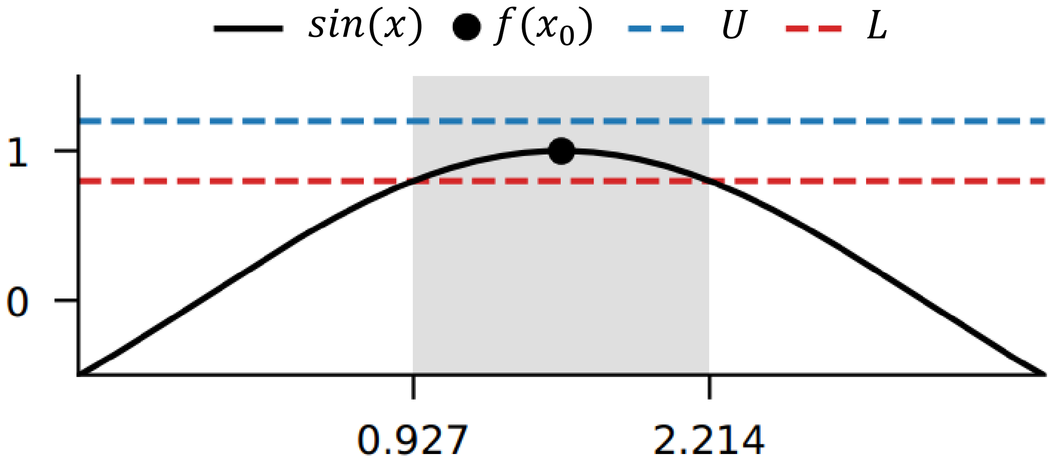
Convex vs. Concave Difference



Convex difference

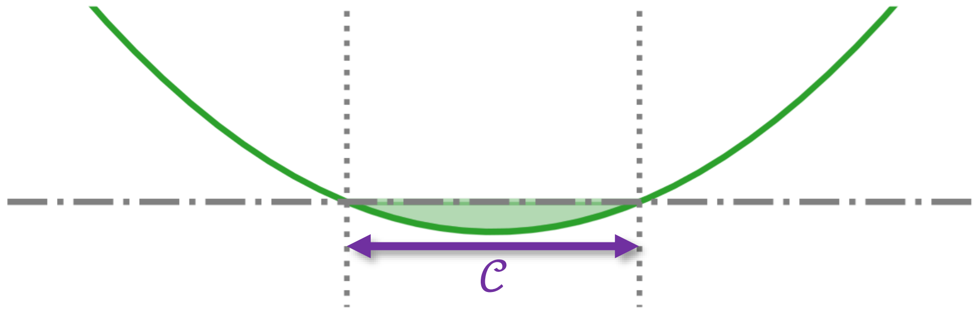


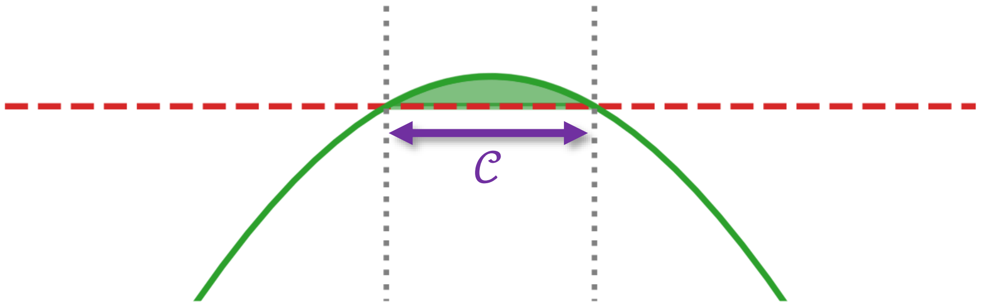
Convex vs. Concave Difference



Convex difference

Concave difference



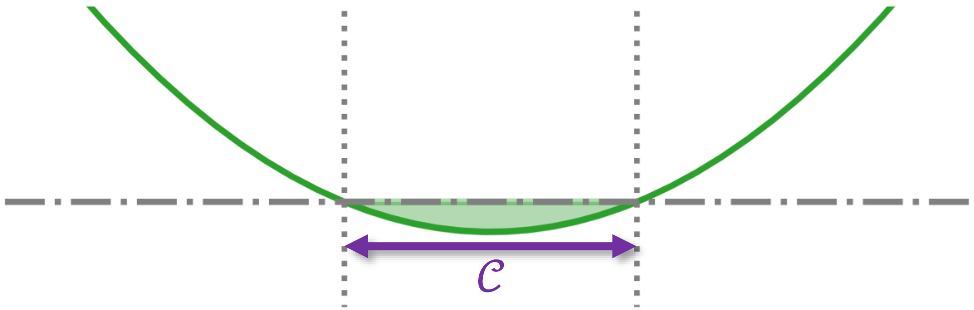


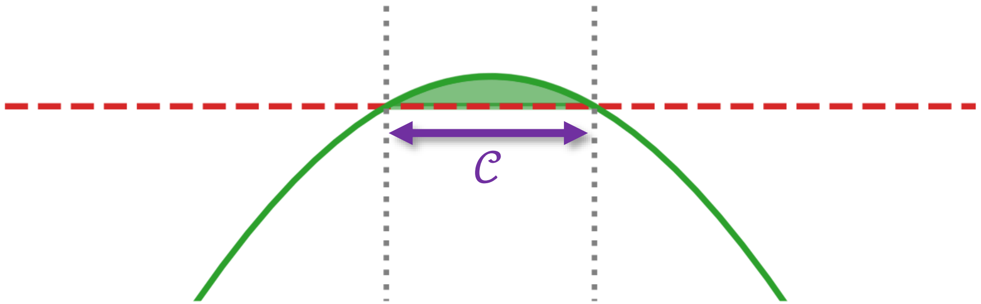
Convex vs. Concave Difference

The DC Heuristic

Convex difference

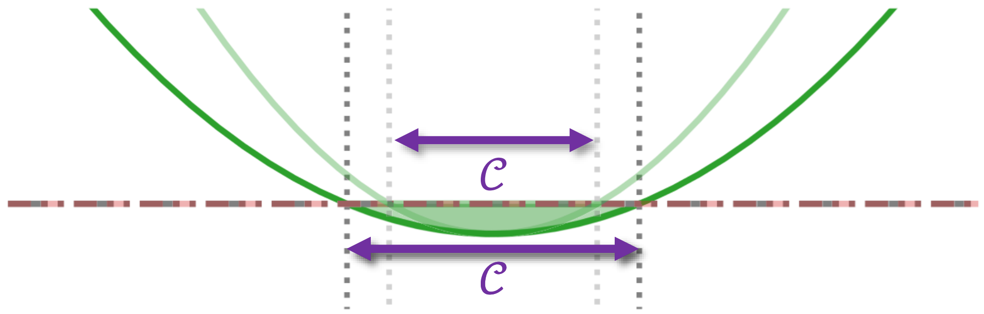
Concave difference





The DC Heuristic

Convex difference



Concave difference

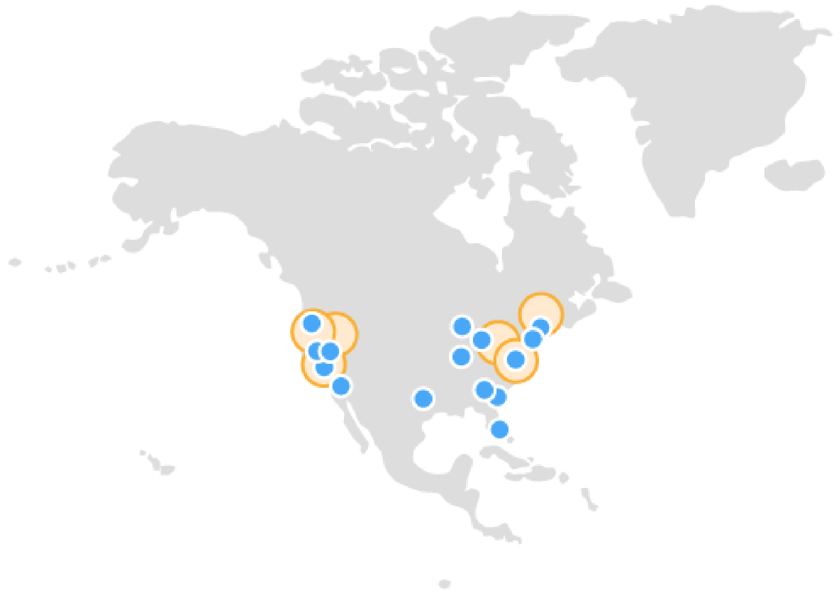
✓ Reduces safe zone violations by up to 30%

Optimizations and Correctness Guarantees

- Approximation correctness guarantees:
  - For constant H and convex/concave functions
  - For others, not guaranteed → in practice error is small
- “Lazy” violation resolution avoids extra syncing
- Optimization on neighborhood of x_0
  - Tradeoff parameter that affects safe zone effectiveness
  - Automatic tuning algorithm

Evaluation Setup

- Different applications
- One coordinator
- n nodes (5 to 1000)
- Vector size d (10 to 200)
- 1K to 311K datapoints



Applications (Functions & Datasets)

INTRUSION DETECTION + DNN

- KDDCup-99 network data
- 5-layer DNN [512, 64, 32, 16, 8] with ReLU activation

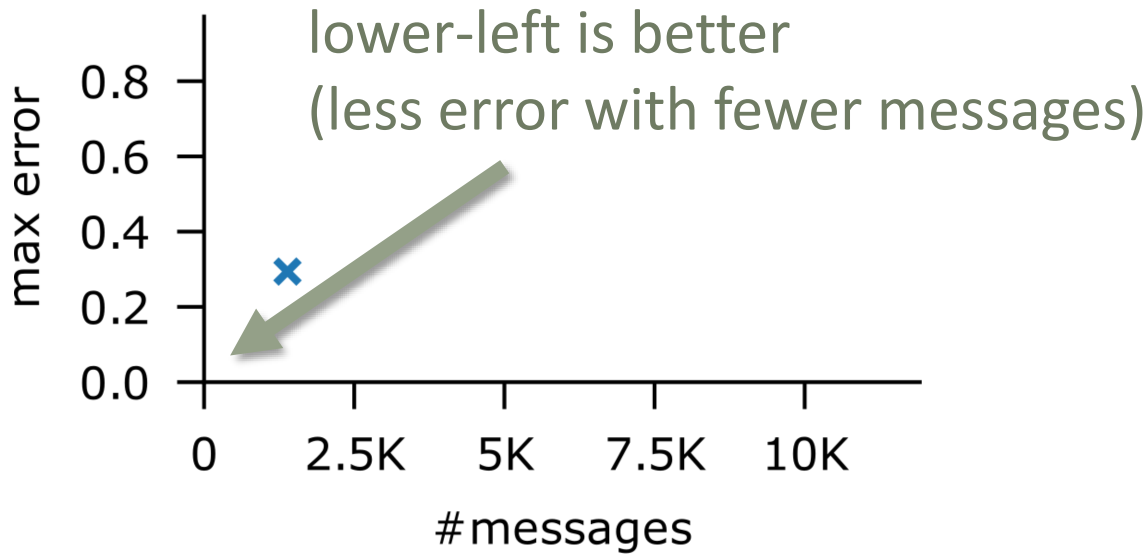
POLLUTION MONITORING + KLD

- Beijing air-quality dataset [Zhang et al, ’17]
- KLD, $D_{KL}(P||Q)$, of PM10 and PM2.5

(we also have inner product, quadratic form, MLP, Rozenbrock, …)

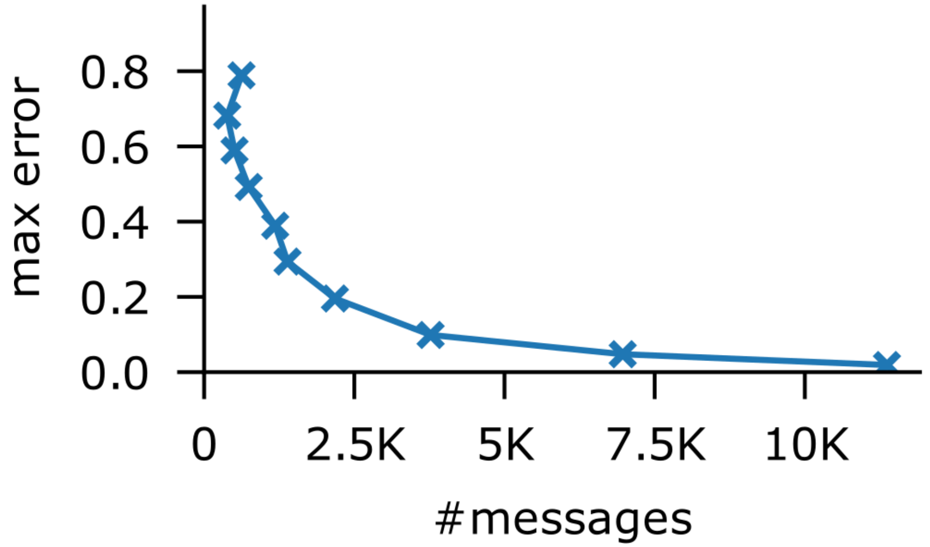
Results: Error-Communication Tradeoff

- One run with specific ϵ
  - X axis – total sent messages
  - Y axis – max error across run



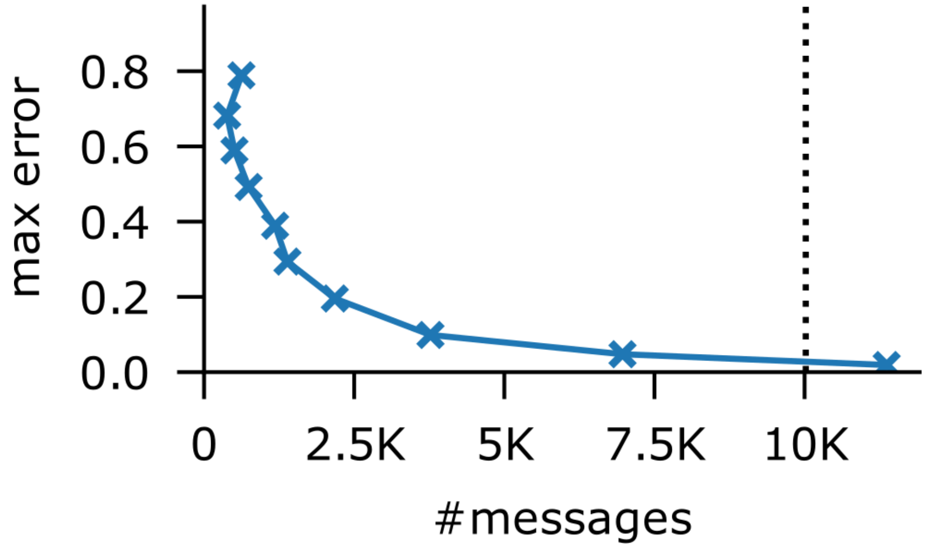
Results:
Error-Communication Tradeoff

- Test on a range of ϵ
- **AutoMon**'s trade-off curve of on this data and function



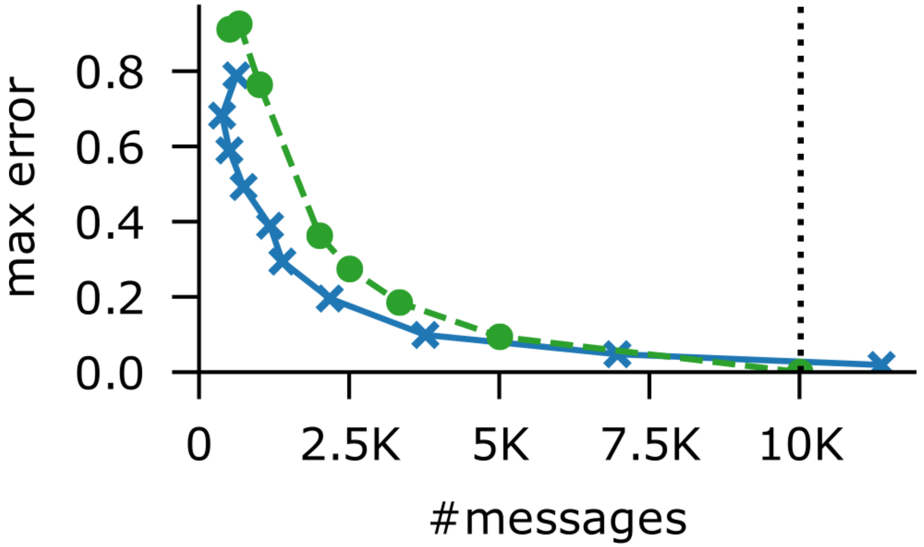
Results: Error-Communication Tradeoff

- Test on a range of ϵ
- **AutoMon**
- **Centralization**:
  just send all data updates.
  - No error
  - State-of-the-art for sketches (they reduce message size, not number of messages)



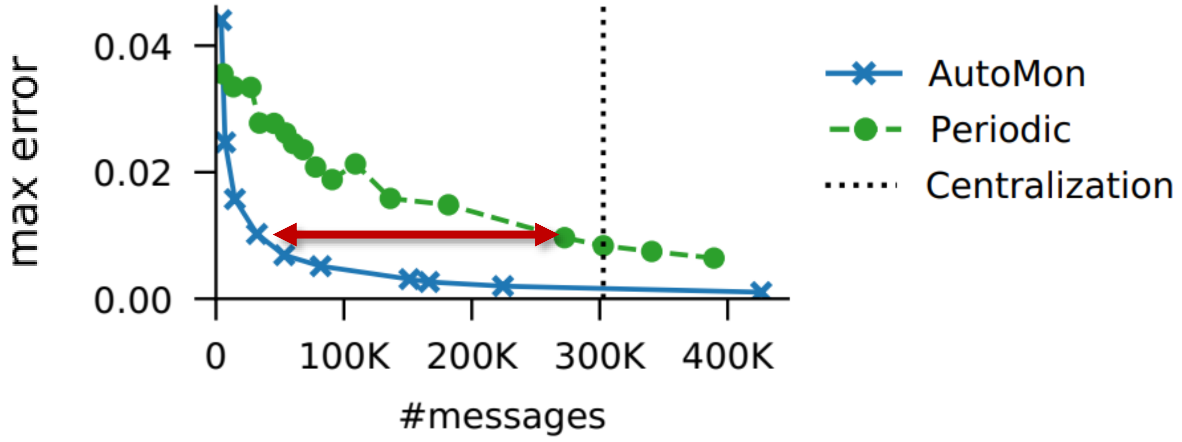
Results:
Error-Communication Tradeoff

- Test on a range of ϵ
- **AutoMon**
- **Centralization**
- **Periodic**: send every N updates
  - Non-adaptive
  - The common approach



Results:
Error-Communication Tradeoff

DNN (intrusion detection)

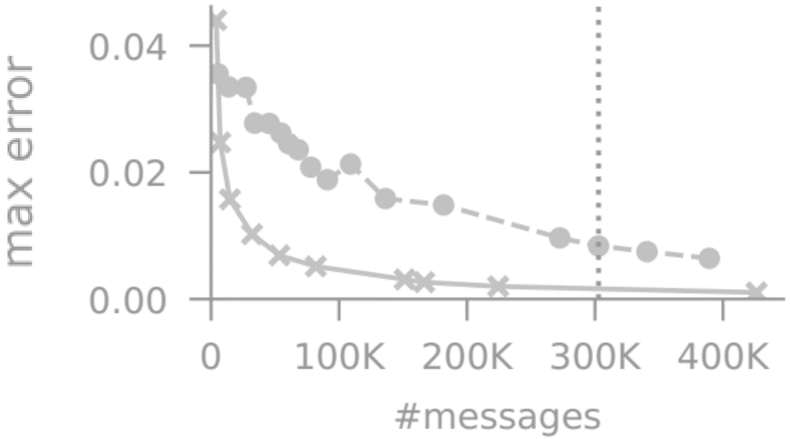


Value changes **slowly:**

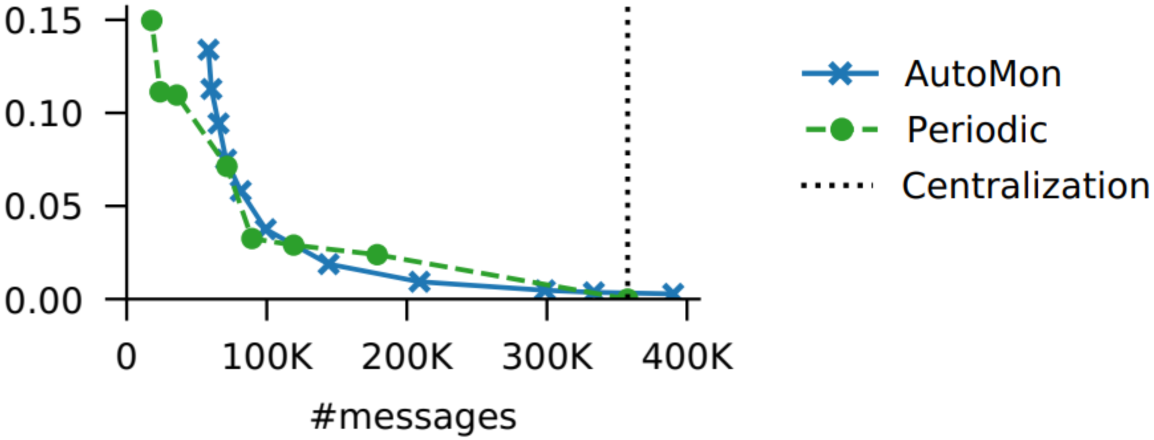
- Periodic wastes messages
- AutoMon is adaptive and communication-efficient
- **2% comm** with low error

Results:
Error-Communication Tradeoff

DNN (intrusion detection)



KLD (pollution monitoring)



Value changes **slowly:**

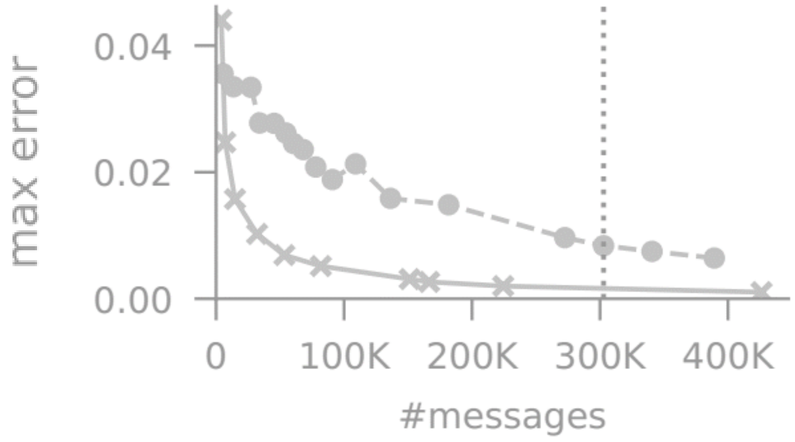
- Periodic wastes messages
- AutoMon is adaptive and communication-efficient
- **2% comm** with low error

Value changes **gradually:**

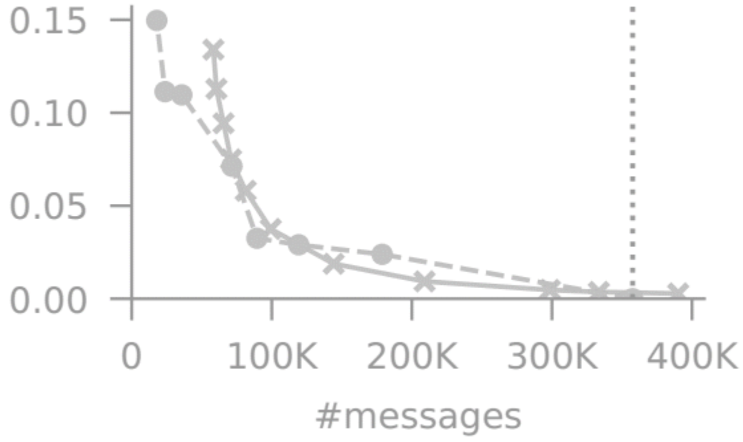
- AutoMon performance similar to Periodic
- AutoMon guarantees error, and is adaptive

Results:
Error-Communication Tradeoff

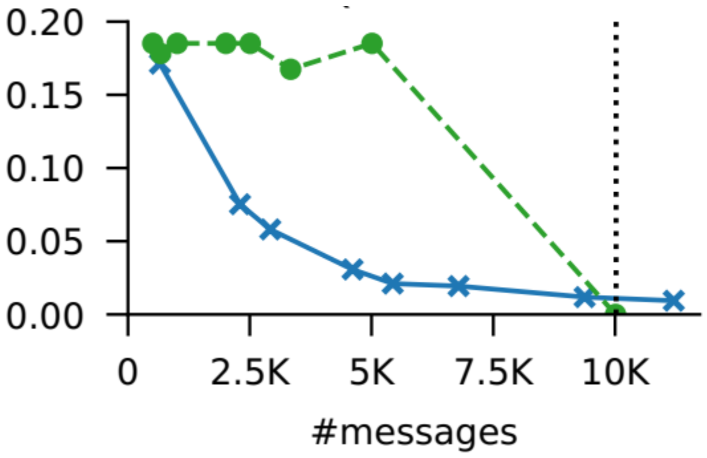
DNN (intrusion detection)



KLD (pollution monitoring)



Quadratic



Value changes **slowly:**

- Periodic wastes messages
- AutoMon is adaptive and communication-efficient
- **2% comm** with low error

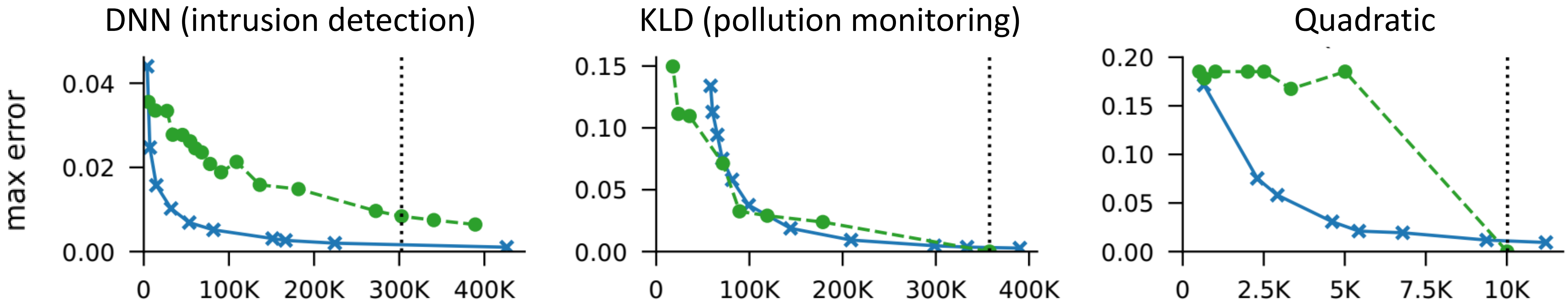
Value changes **gradually:**

- AutoMon performance similar to Periodic
- AutoMon guarantees error, and is adaptive

Value changes **quickly:**

- Large error in Periodic
- AutoMon is adaptive: smooth, superior tradeoff

Results: Error-Communication Tradeoff

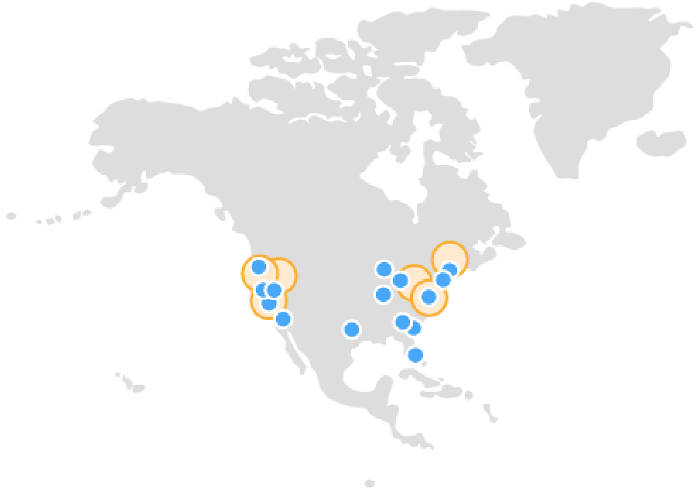


AutoMon:

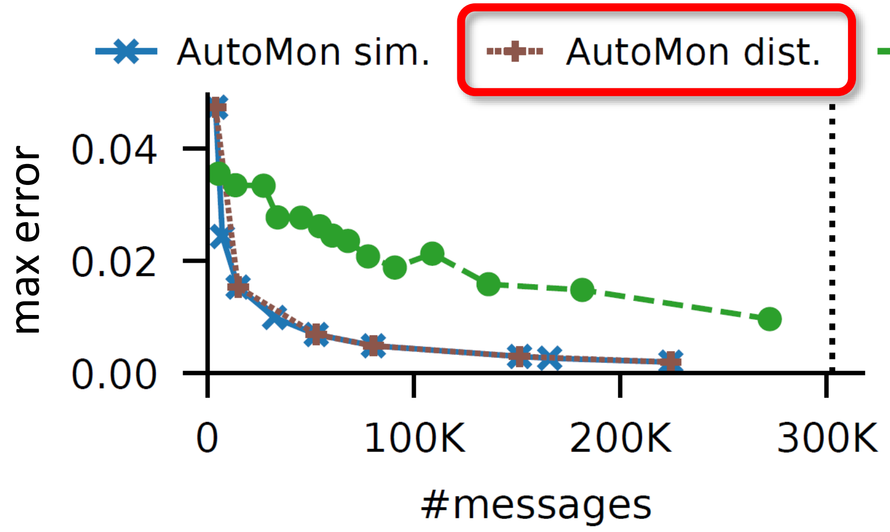
- ✓ provides equivalent or **superior tradeoff** to current approaches …
- ✓ **automatically** from source code.

Error-Bandwidth Tradeoff

- AWS experiments:
  - Coordinator on us-west-2
  - Nodes on us-east-2
  - Round trip time = 56ms
- Investigate:
  1) Validity of simulation
  2) Error-bandwidth tradeoff
  3) Estimate reduction in traffic

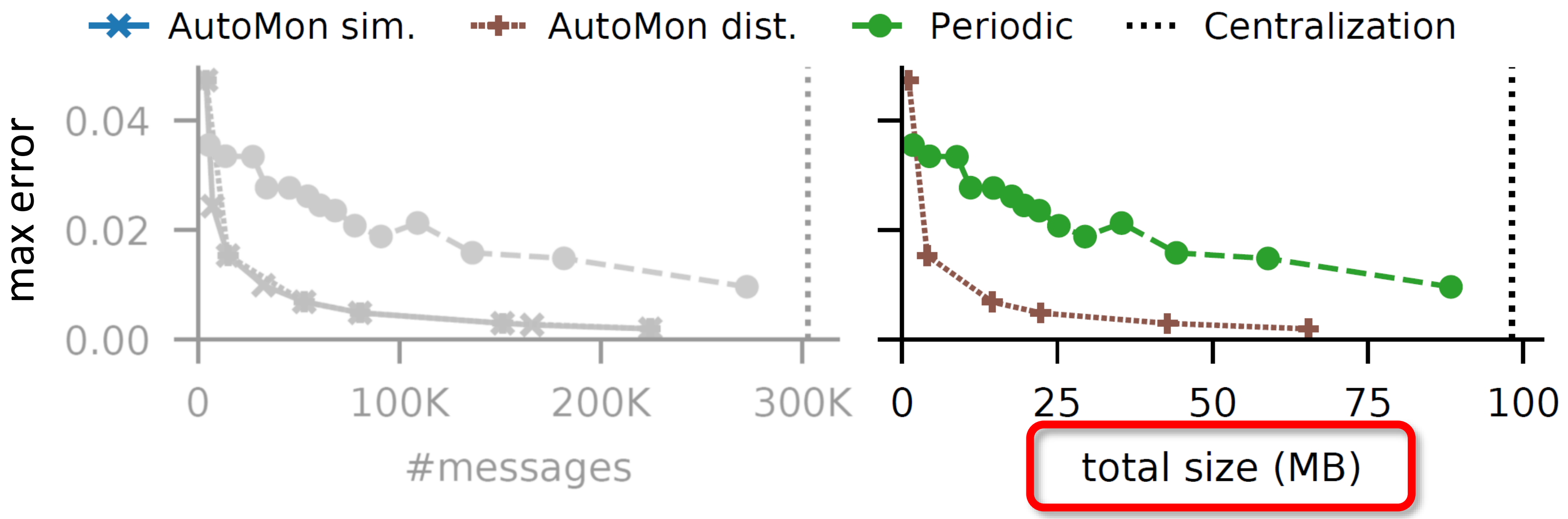


DNN Results: Error-Bandwidth Tradeoff



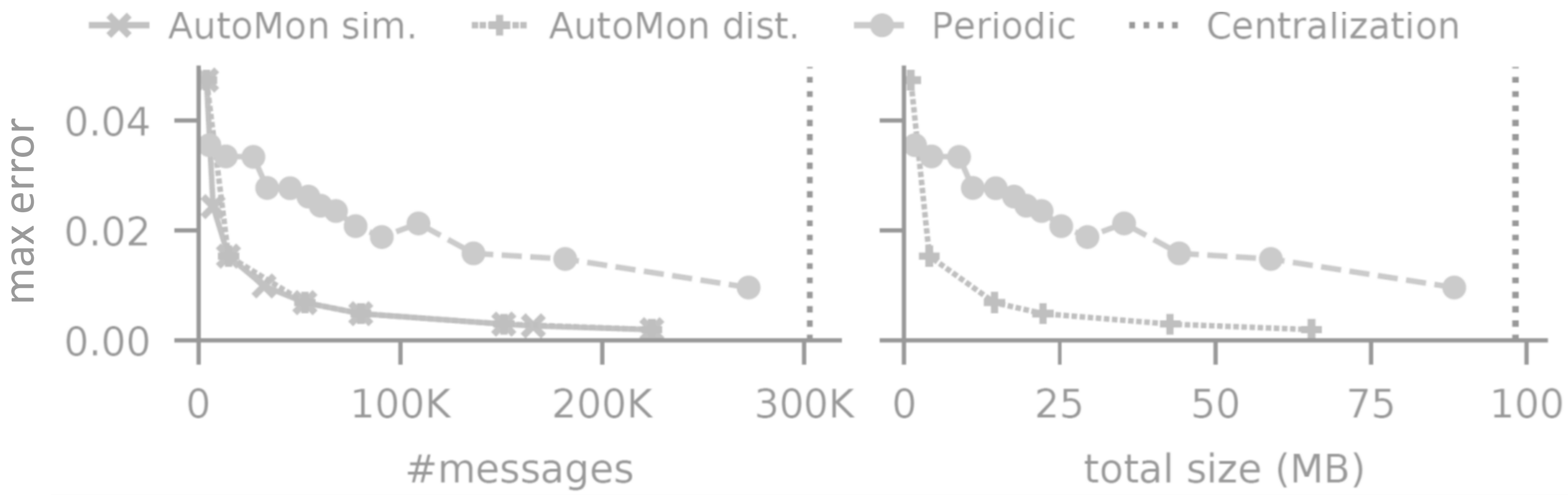
✓ Similar reduction in messages (0 to 16% error)

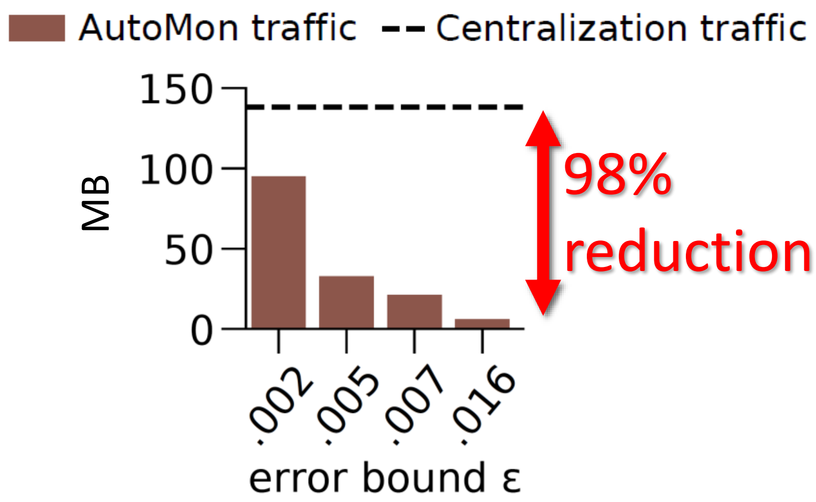
DNN Results: Error-Bandwidth Tradeoff



- ✓ Same reduction in messages (0 to 16% error)
- ✓ Error-BW tradeoff agrees with error-messages tradeoff

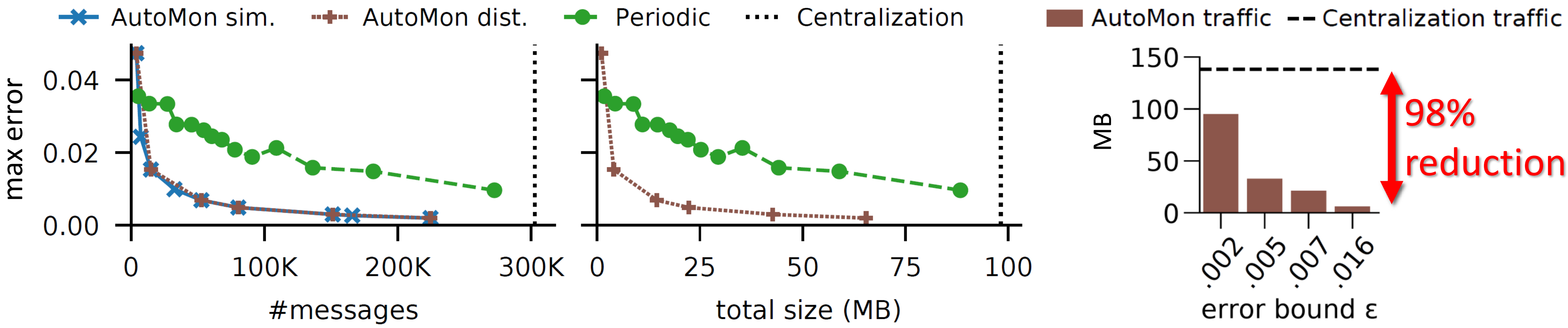
DNN Results:
Error-Bandwidth Tradeoff





- ✓ Same reduction in messages (0 to 16% error)
- ✓ Error-BW tradeoff agrees with error-messages tradeoff
- ✓ Traffic reduced to 1/3 on average, and up to 98%

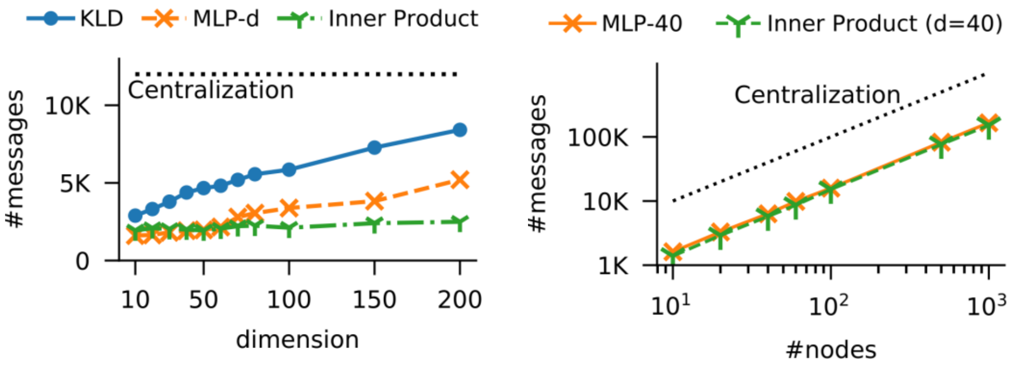
DNN Results: Error-Bandwidth Tradeoff



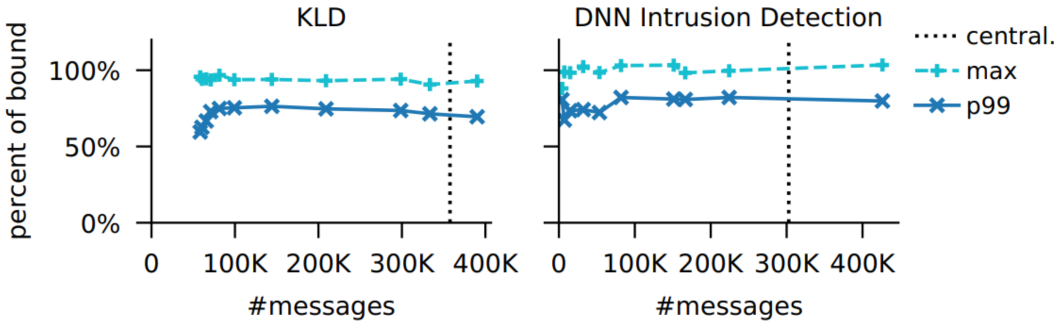
- ✓ Same reduction in messages (0 to 16% error)
- ✓ Error-BW tradeoff agrees with error-messages tradeoff
- ✓ Traffic reduced to 1/3 on average, and up to 98%

Additional Experiments

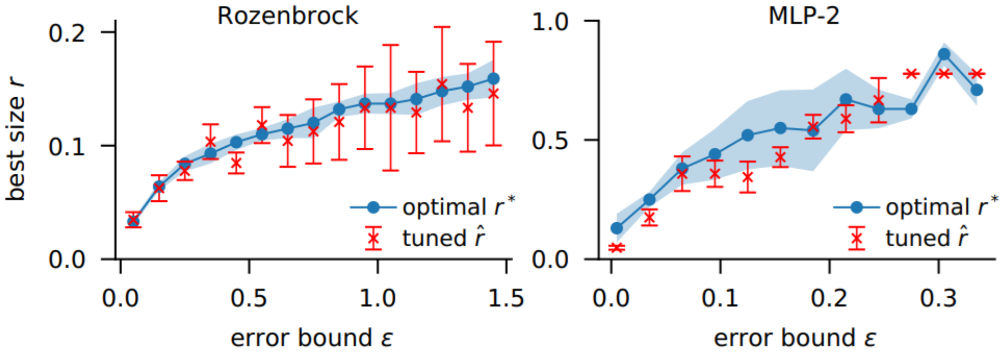
Scalability



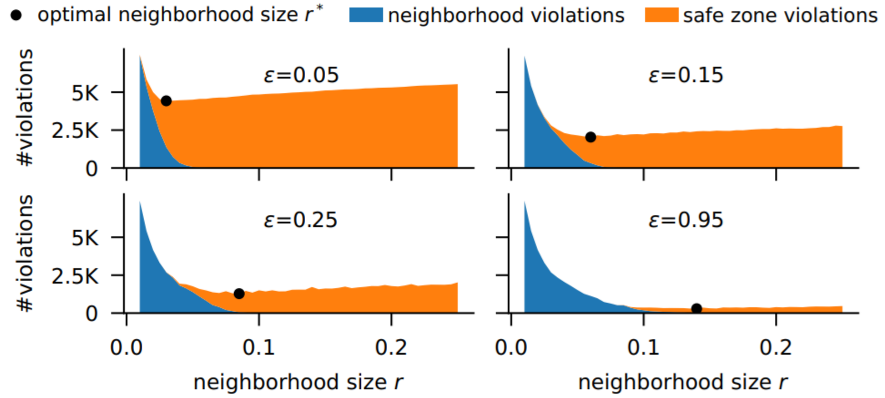
Approximation guarantees and empirical errors



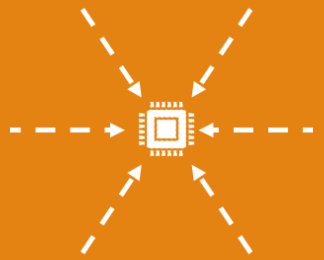
Automatic parameter tuning



Neighborhood size tradeoffs

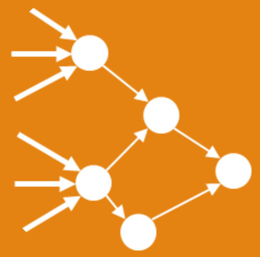


Related Work



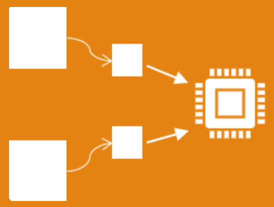
Sampling

- Send only some updates
- Delay warnings
- Miss transients



Distributed Dataflow, Query Planning

- Combine operators
- No built-in operator for f_{nn}



Sketching

- Compress updates, approximate f_{nn}
- No sketch for f_{nn}
- Reduce bandwidth, not number of messages

Geometric Monitoring

- Avoid sending updates
- No bound for f_{nn}
- Devs are not PhDs

easy to apply ⟷ hard to apply

AutoMon vs. Universal Sketches

- *Liu et al., One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon, SIGMOD ‘16*
- Universal sketch: just implement a function $g(x_i)$
- Limited to *Stream-PolyLog* functions in the turnstile model:
  - $f(x) = \sum g(x_i)$
  - x_i are frequencies
  - g monotonic, bounded by $O\left(x_i^2\right)$
- AutoMon supports wider variety of x and f

Summary

- AutoMon: first truly automatic distributed monitor
  - ✓ **Automatic**: Arbitrary functions of $\bar{x}$.
  - ✓ **Accessible**: Works from source code.
  - ✓ **Efficient**: Superior error-communication tradeoff to existing methods.
- Allows difficult functions without hand-crafted solution
  - For example, for DNN, reduces communication by up to ×50.
- Open source: https://github.com/hsivan/automon

hadarivan@cs.technion.ac.il mgabel@cs.toronto.edu assaf@technion.ac.il