

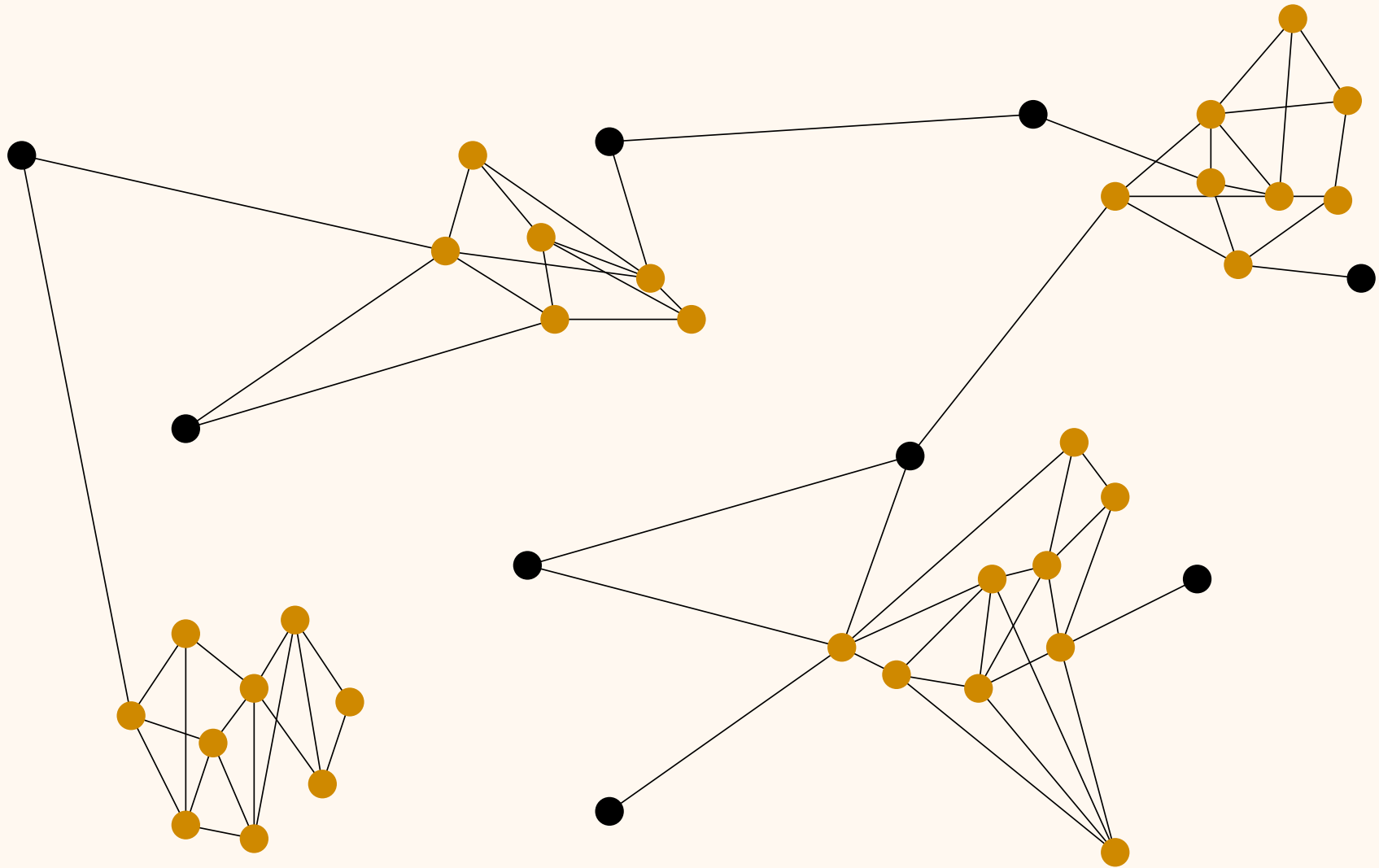
*On the Utility of Inference
Mechanisms*

Ethan Blanton Sonia Fahmy
Greg N. Frederickson

June 23, 2009

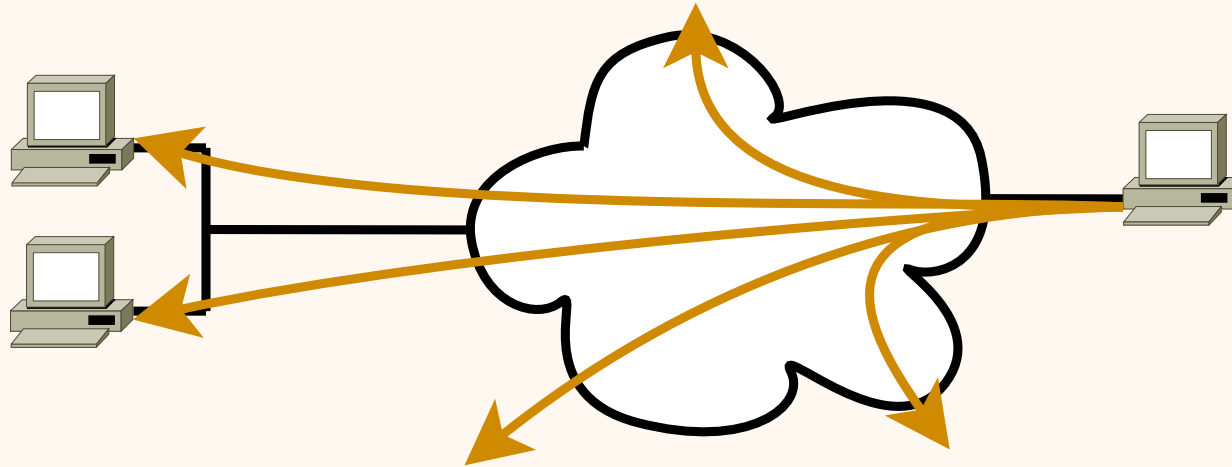
- We present an algorithm for determining when and if an **inference mechanism** can be used to reduce network measurement load on the network.
- Inference mechanisms reduce the network traffic used by **active measurements**, at the cost of accuracy.
- The overhead for inference mechanisms can be large.
- Our technique allows for inference to be used when it is beneficial, and more accurate direct measurements otherwise.
- We use spanning forests and estimation of edge connectivity to accomplish this.

Mixing Inference and Direct Measurement



- We are looking at **measurement infrastructures** which perform **on demand** active measurements for **end hosts**.
- Measurement schedules are not known *a priori*.
- Measurement infrastructure **providers** want to minimize active measurement traffic.
- Measurement infrastructure **users** are end host applications.

- The literature is full of inference mechanisms.
 - GNP¹, Vivaldi², IDMaps³, NetQuest⁴, etc.
 - latency, jitter, available bandwidth, etc.



¹T. S. Eugene Ng and Hui Zhang. Towards Global Network Positioning. ACM SIGCOMM Workshop on Internet Measurement 2001.

²Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A Decentralized Network Coordinate System. ACM SIGCOMM 2004.

³Paul Francis, Sugih Jamin, Cheng Jin, Yixin Jin, Danny Raz, Yuval Shavitt, and Lixia Zhang. IDMaps: A Global Internet Host Distance Service. IEEE/ACM Transactions on Networking. Oct. 2001.

⁴Han Hee Song, Lili Qiu, and Yin Zhang. NetQuest: a flexible framework for large-scale network measurement. ACM SIGMETRICS 2006.

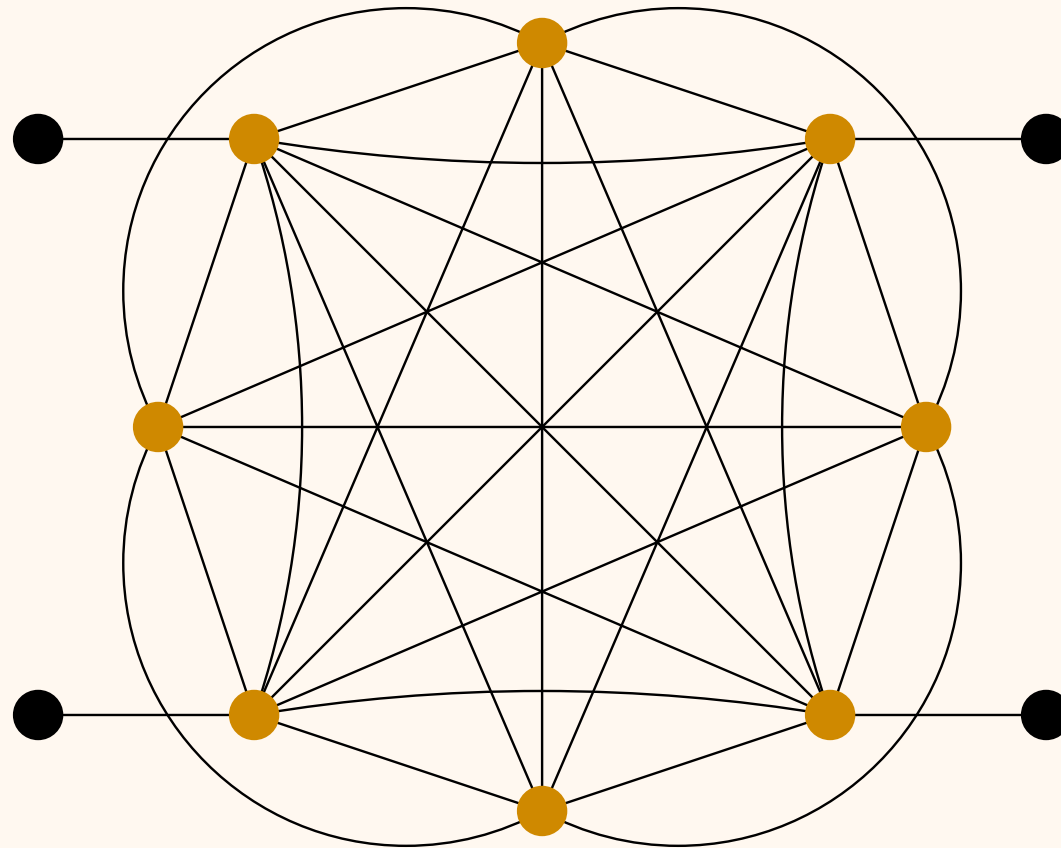
- **Observation:** Inference mechanisms lose fidelity over direct measurement.
- **Observation:** Due to the hidden constant, inference mechanisms can be **more expensive** than direct measurement.
- **Observation:** A measurement infrastructure sees measurement requests from disparate applications.

- **Observation:** Inference mechanisms lose fidelity over direct measurement.
- **Observation:** Due to the hidden constant, inference mechanisms can be **more expensive** than direct measurement.
- **Observation:** A measurement infrastructure sees measurement requests from disparate applications.
- **Conclusion:** A measurement infrastructure is uniquely situated to determine when a collection of direct measurement requests can be replaced by inference to reduce measurement load on the network.

- Recall that many inference mechanisms use $O(n)$ measurement probes to estimate the properties of $O(n^2)$ paths.
- Assume that we can identify the hidden constant in that $O(n)$.
 - It can be large.
 - We will call it k .
- Given a set of measurements from a measurement service and the constant k , we can now identify a **tipping point** at which the total number of measurements in the system is larger than an all-hosts inference.

When to use inference cont'd

● Inference ● Non-inference



$k = 3$

No inference: 32 measurements

All-hosts inference: 36 measurements

With inference: 28 measurements

- To determine if inference is beneficial, we have to answer a slightly different question.

“Does there exist a subset of hosts of size n , such that the number of measurements performed between those hosts is greater than $k \cdot n$?”
- In our previous example, the answer is “Yes, the gold-colored hosts.”
- The task, then, is to identify all such subsets.

Some facts:

- A k -regular graph is a graph in which each node has a degree of exactly k .
- This is precisely our tipping point.
- Finding a k -regular subgraph in a general graph is NP -complete.

Some facts:

- A k -regular graph is a graph in which each node has a degree of exactly k .
- This is precisely our tipping point.
- Finding a k -regular subgraph in a general graph is NP -complete.

Therefore, we seek a solution which trades off optimality for tractable computational complexity.

- We use a set of **minimum spanning forests** to identify edges which are likely to be part of low edge-count cuts of the graph.
- Edges identified as likely candidates are pruned from the graph.
- The **connected components** which remain are assumed to be sets of hosts which would benefit from having their internal direct measurements replaced with an inference.

- Represent hosts as vertices, and requested measurements as edges.
- Create a set of graphs identical to the unweighted input graph.
- Every edge in these new graphs is assigned a weight from a uniform random distribution.
- For each of these graphs, find a minimum spanning forest using the weights we just assigned.
- Remove any edges which appear in more of these forests than some threshold score from the input graph.
- Determine the remaining connected components, and return them.

- Random spanning forests give us a quick estimate of **edge connectivity**.
- For any edge, we can lower bound the probability that it occurs in a **most constricting cut**.
- Additionally, we can upper bound the probability that the actual score for an edge falls significantly below the lower bound on its expected score.

Let:

- e be an edge in the input graph.
- (V', E') be the connected component of the input graph containing e .
- V_1 and V_2 be disjoint sets of vertices in V' forming the two separated halves of a min cut of (V', E') .
- $c(V_1, V_2)$ be the number of edges of the cut between V_1 and V_2 in G .

Lemma 1 *The probability that edge e is an edge in F_i is at least $1/c(V_1, V_2)$, and the expected score for e is thus at least $f/c(V_1, V_2)$. Furthermore,*

$\Pr(\text{score}(e) \leq forests/c(V_1, V_2) - \alpha\sqrt{forests}) \leq \exp(-2\alpha^2)$ for any constant α .

- Lemma 1 shows us that the expected score of an edge is a function of both the number of spanning forests, which we choose, and $c(V_1, V_2)$, which is dependent on the structure of the graph.
- The number of forests trades off **complexity** and **accuracy**.
- The bounds on our expected score indicate that the threshold score must be essentially proportional to the number of forests.
- Additionally, the threshold score must be inversely related to k , as higher values of k indicate higher cost of inference, thus requiring more aggressive pruning.

We evaluate this algorithm on a number of graphs:

- Simple graphs of interesting topology.
- Large graphs having properties taken from the popular UUSee streaming television service in China.

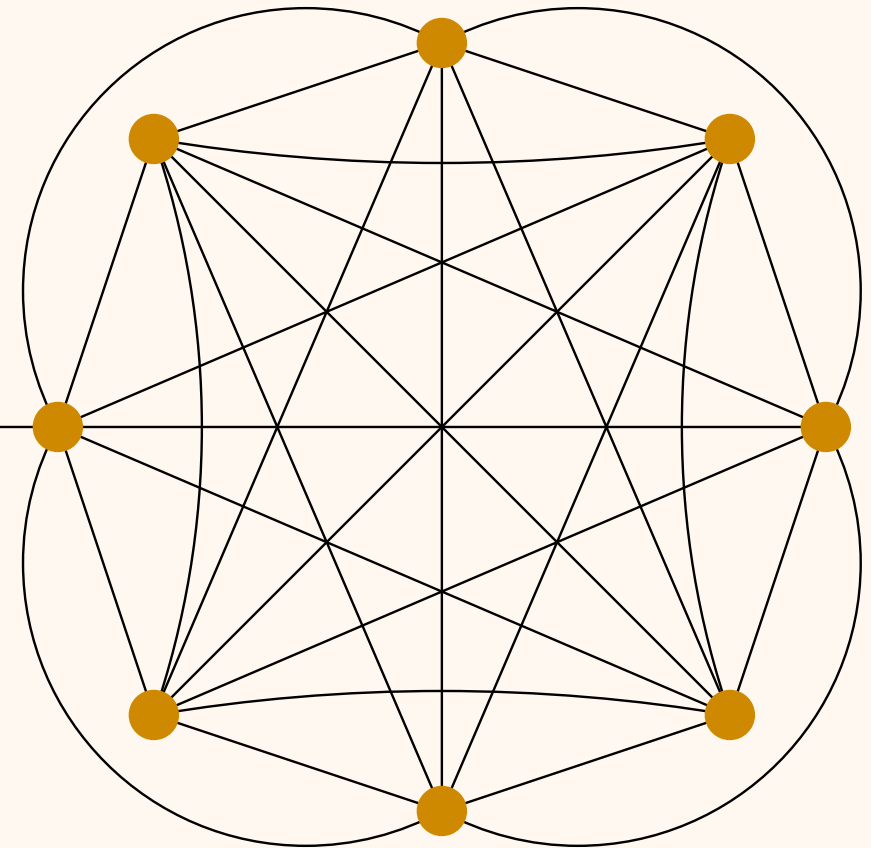
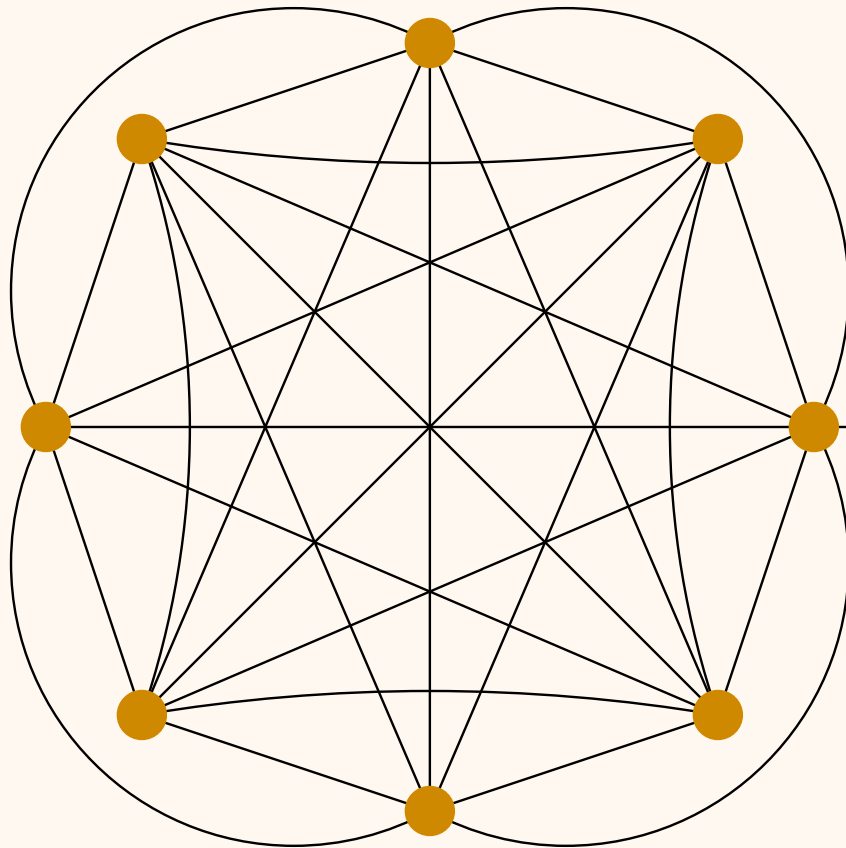
We evaluate this algorithm on a number of graphs:

- Simple graphs of interesting topology.
- Large graphs having properties taken from the popular UUSee streaming television service in China.

The key measure of comparison is the **number of measurements** required to provide measurement results for all requested paths.

● Inference

● Non-inference



$k = 3, n = 17$

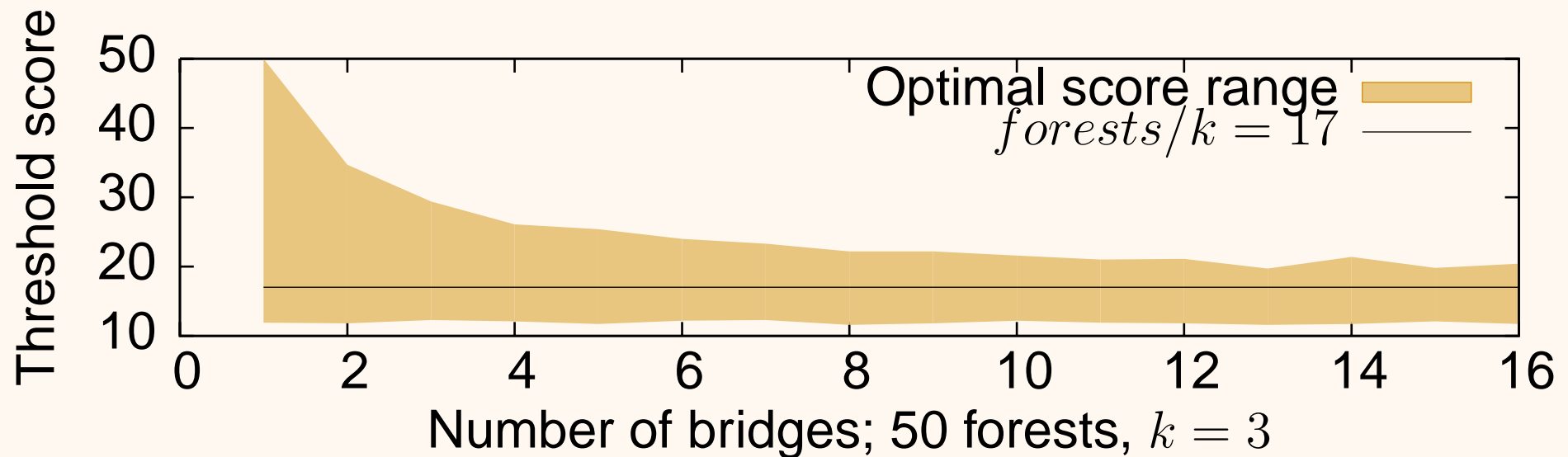
Whole-graph inference: $kn = 51$

No inference: 58 measurements

Optimal inference: 50 measurements

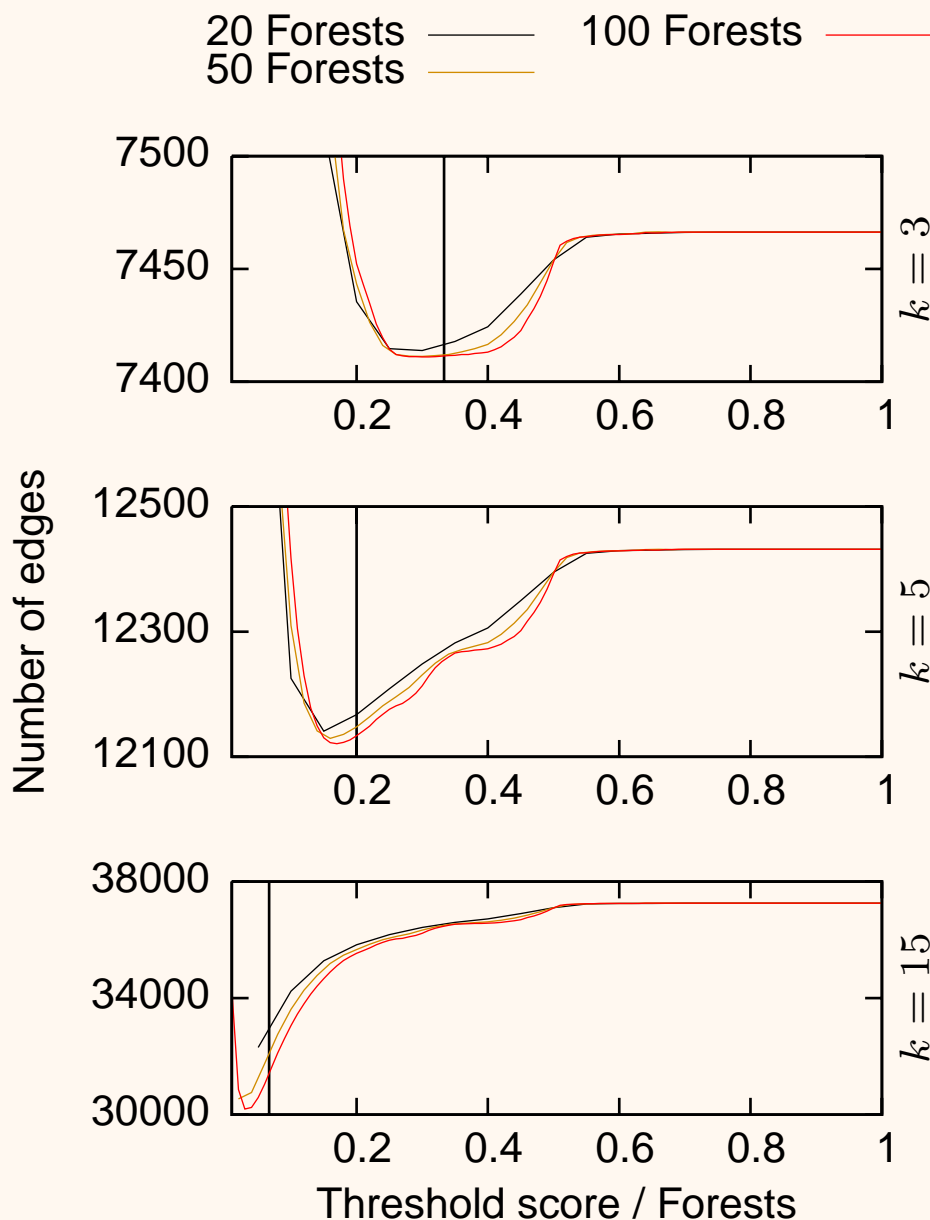
- This graph is trivial, because the two “bridge” edges appear in every spanning tree.
- Note that, for every run, there is a threshold score above which *all* threshold scores yield an optimal result.
- Things get trickier if we add more bridges between the fully connected subgraphs.

- This graph is trivial, because the two “bridge” edges appear in every spanning tree.
- Note that, for every run, there is a threshold score above which *all* threshold scores yield an optimal result.
- Things get trickier if we add more bridges between the fully connected subgraphs.



- Ten random **small world** graphs representing a single UUSee⁵ channel.
 - 2,500 vertices each.
 - About 50,000 edges each.
 - Clustering coefficient of about 2.5.
 - Average path length of about 2.3.
- As before, our metric of interest is the number of **total measurements** required.
- Unfortunately, in this case we do not know the ideal solution, so we can compare only against full-graph inference and no inference.

⁵Chuan Wu, Baochun Li and Shuqiao Zhao. Magellan: Charting Large-Scale Peer-to-Peer Live Streaming Topologies. ICDCS 2007.



- Substantial savings are realized over the original 53,000 edge topologies.
- Savings over full-graph inference increase with k .
- Note the minimum at threshold score values just smaller than the number of forests over k (vertical line).
- Larger numbers of forests produce lower minimums and broader troughs.

- Inference mechanisms can induce a heavier load than sparse direct measurements.
- Given a set of hosts and measurements between them, we can efficiently identify subsets of hosts which would benefit from using an established inference algorithm rather than performing direct measurements.
- Savings on small-world graphs representative of popular Peer-to-peer applications' communication patterns are measurable.
- More accurate direct measurements can be preserved where inference is not of sufficient benefit.

Thanks for listening.

Questions?

More precisely:

We construct a graph, representing measurement hosts as vertices, and requested measurements as edges.

Given such a graph $G = (V, E)$, our goal is to transform it into another graph $G' = (V, E')$ such that we minimize $|E'|$, where $0 \leq |E'| \leq |E|$ and $0 \leq |E'| \leq k|V|$.

More precisely:

We construct a graph, representing measurement hosts as vertices, and requested measurements as edges.

Given such a graph $G = (V, E)$, our goal is to transform it into another graph $G' = (V, E')$ such that we minimize $|E'|$, where $0 \leq |E'| \leq |E|$ and $0 \leq |E'| \leq k|V|$.

The only operation we can use in this minimization is replacing all edges among subsets V_i of V by $k|V_i|$ edges. (That is, replace direct measurements with inference.)

$$\text{score}(e) = \sum_{i=1}^f \begin{cases} 1 & : e \in F_i \\ 0 & : e \notin F_i \end{cases}$$

```
def inference_groups( $G, f, s$ )
for  $i \in 1..f$  do
  let  $V(G_i) = V(G)$ 
  let  $E(G_i) = E(G)$ 
  for  $e \in E(G_i)$  do
    let  $wt(e) = \text{random}()$ 
  let  $F_i = \text{MSF}(G_i)$ 
for  $e \in E(G)$  do
  if  $\text{score}(e) > s$  then
    let  $E(G) = E(G) \setminus e$ 
return components( $G$ )
```

Inputs:

- G is the input graph.
- f is the number of minimum spanning forests to be created.
- s is the number of forests in which an edge must appear to be pruned.

Outputs:

- Connected components for which internal measurements should be replaced by inference.

$$\text{score}(e) = \sum_{i=1}^f \begin{cases} 1 & : e \in F_i \\ 0 & : e \notin F_i \end{cases}$$

```
def inference_groups( $G, f, s$ )
for  $i \in 1..f$  do
  let  $V(G_i) = V(G)$ 
  let  $E(G_i) = E(G)$ 
  for  $e \in E(G_i)$  do
    let  $wt(e) = \text{random}()$ 
  let  $F_i = \text{MSF}(G_i)$ 
for  $e \in E(G)$  do
  if  $\text{score}(e) > s$  then
    let  $E(G) = E(G) \setminus e$ 
return components( $G$ )
```

Create a set of f graphs identical to the unweighted graph G
...

$$\text{score}(e) = \sum_{i=1}^f \begin{cases} 1 & : e \in F_i \\ 0 & : e \notin F_i \end{cases}$$

```
def inference_groups( $G, f, s$ )
for  $i \in 1..f$  do
  let  $V(G_i) = V(G)$ 
  let  $E(G_i) = E(G)$ 
  for  $e \in E(G_i)$  do
    let  $wt(e) = \text{random}()$ 
  let  $F_i = \text{MSF}(G_i)$ 
for  $e \in E(G)$  do
  if  $\text{score}(e) > s$  then
    let  $E(G) = E(G) \setminus e$ 
return components( $G$ )
```

... except that in each graph every edge is assigned a weight from a uniform random distribution.

$$\text{score}(e) = \sum_{i=1}^f \begin{cases} 1 & : e \in F_i \\ 0 & : e \notin F_i \end{cases}$$

```
def inference_groups( $G, f, s$ )
for  $i \in 1..f$  do
  let  $V(G_i) = V(G)$ 
  let  $E(G_i) = E(G)$ 
  for  $e \in E(G_i)$  do
    let  $wt(e) = \text{random}()$ 
  let  $F_i = \text{MSF}(G_i)$ 
for  $e \in E(G)$  do
  if  $\text{score}(e) > s$  then
    let  $E(G) = E(G) \setminus e$ 
return components( $G$ )
```

For each of these graphs, find a minimum spanning forest using the weights we just assigned.

$$\text{score}(e) = \sum_{i=1}^f \begin{cases} 1 & : e \in F_i \\ 0 & : e \notin F_i \end{cases}$$

```
def inference_groups( $G, f, s$ )
for  $i \in 1..f$  do
  let  $V(G_i) = V(G)$ 
  let  $E(G_i) = E(G)$ 
  for  $e \in E(G_i)$  do
    let  $wt(e) = \text{random}()$ 
  let  $F_i = \text{MSF}(G_i)$ 
for  $e \in E(G)$  do
  if  $\text{score}(e) > s$  then
    let  $E(G) = E(G) \setminus e$ 
return components( $G$ )
```

Remove any edges which appear in more than s of these forests.

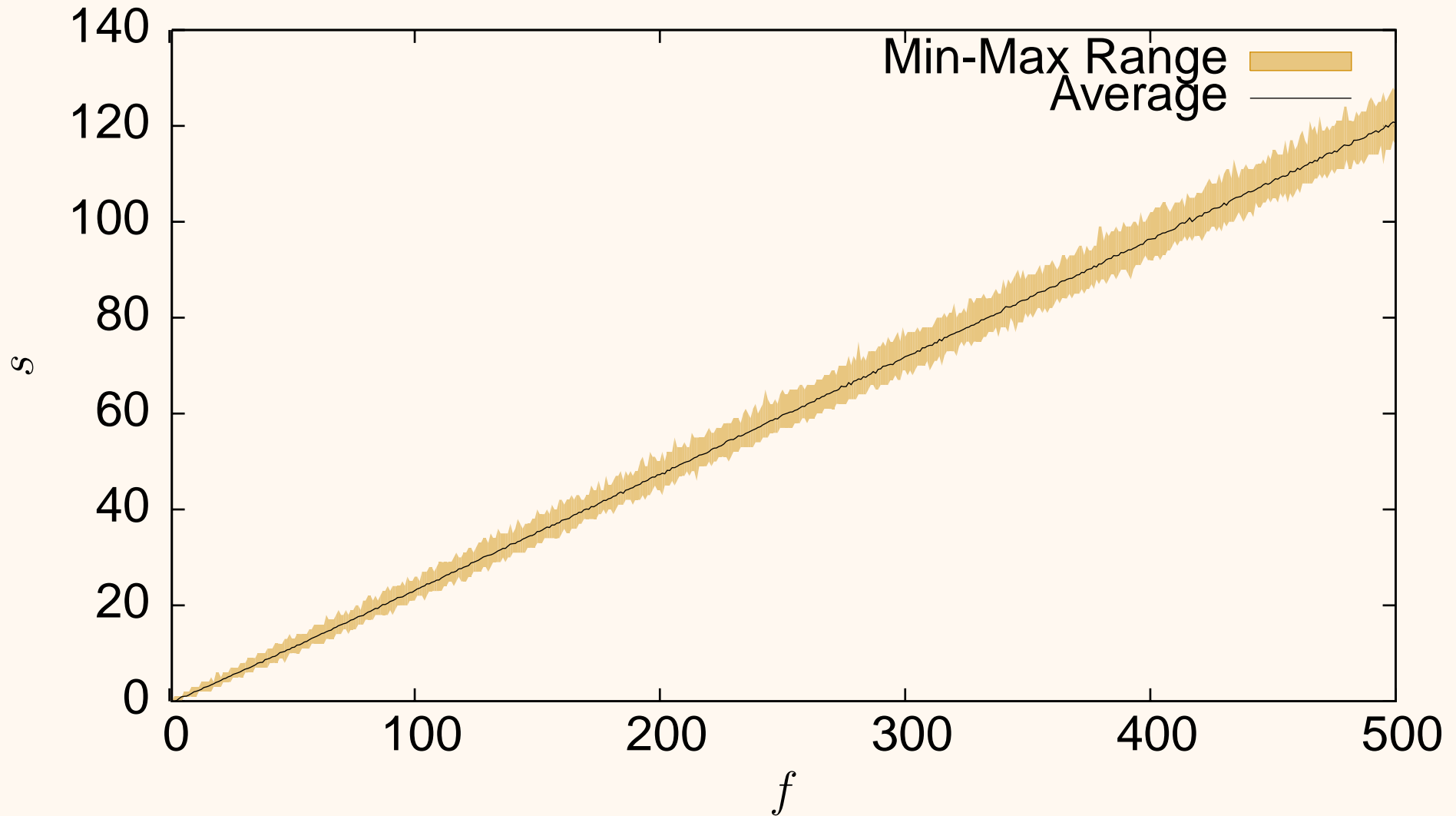
$$\text{score}(e) = \sum_{i=1}^f \begin{cases} 1 & : e \in F_i \\ 0 & : e \notin F_i \end{cases}$$

```
def inference_groups( $G, f, s$ )
for  $i \in 1..f$  do
  let  $V(G_i) = V(G)$ 
  let  $E(G_i) = E(G)$ 
  for  $e \in E(G_i)$  do
    let  $wt(e) = \text{random}()$ 
  let  $F_i = \text{MSF}(G_i)$ 
for  $e \in E(G)$  do
  if  $\text{score}(e) > s$  then
    let  $E(G) = E(G) \setminus e$ 
return components( $G$ )
```

Finally, determine the remaining connected components, and return them.

- The complexity of a minimum spanning forest algorithm such as Kruskal's Algorithm is $O(|E| \log |E|)$.
- This is too slow for on-line computation of incoming measurement requests in a large measurement infrastructure.

- The complexity of a minimum spanning forest algorithm such as Kruskal's Algorithm is $O(|E| \log |E|)$.
- This is too slow for on-line computation of incoming measurement requests in a large measurement infrastructure.
- However, there exist algorithms which compute *updates* to a minimum spanning forest in amortized $O(\log^4(|V|))$ time per insertion or deletion, with a worst case of $O(|V|^{1/2})$.
- With these algorithms, we expect to compute on-line changes of very large graphs in reasonable time.



f versus the value of s above which all results are optimal for Synthetic Topology 1, aggregate of ten runs.