Hop-Constrained Oblivious Routings

Speaker: Goran Zuzic

STOC 2021

Mohsen Ghaffari
ETH Zürich

Bernhard Haeupler
CMU / ETH Zürich

Goran Zuzic
ETH Zürich
1 Motivating problem

2 Background

3 Main technical ideas

4 Conclusion
Motivating problem

- **Graph** $G$ (undirected, unweighted).

Input: source-sink demands.
Output: Choose paths.
Objective: min. makespan. Paths are chosen obliviously.
Motivating problem

- **Graph** $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan. Paths are chosen obliviously.
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Objective:** min. makespan.

Paths are chosen obliviously.
Motivating problem

- **Graph G** (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan. Paths are chosen obliviously.
Motivating problem

- **Graph G** (undirected, unweighted).
- **Input:**
  source-sink demands.
- **Output:**
  Choose paths.

![Graph Diagram]
Motivating problem

- **Graph** $G$ (undirected, unweighted).
- **Input:**
  source-sink demands.
- **Output:**
  Choose paths.

\[ s_1, t_1, s_2, t_2, s_3, t_3 \]
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.

Objective: $\min.$ makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- Input: source-sink demands.
- Output: Choose paths.
- Objective: min. makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- **Graph** $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- Graph \( G \) (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- Input: source-sink demands.
- Output: Choose paths.
- Objective: min. makespan.
Motivating problem

- **Graph** $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:**
  - source-sink demands.
- **Output:**
  - Choose paths.
- **Objective:**
  - $\text{min. makespan.}$
Motivating problem

- **Graph** $G$ (undirected, unweighted).

- **Input:**
  - source-sink demands.

- **Output:**
  - Choose paths.

- **Objective:**
  - min. makespan.

![Diagram](image-url)
Motivating problem

- **Graph** $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- **Graph** $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
Motivating problem

- Graph $G$ (undirected, unweighted).
- **Input:** source-sink demands.
- **Output:** Choose paths.
- **Objective:** min. makespan.
- Paths are chosen obliviously.
Choosing paths obliviously

**Intuition:** each driver asks an offline mobile navigation app to produce a path (given a starting point $s_i$ and destination $t_i$).
Choosing paths obliviously

**Intuition:** each driver asks an offline mobile navigation app to produce a path (given a starting point $s_i$ and destination $t_i$).

**Formally:**

**Definition**

Given $G = (V, E)$, an **oblivious routing** $R$ is a collection of $|V|^2$ distributions $R = \{R_{u,v}\}_{u,v \in V}$, where for each pair of nodes $u, v \in V$ we have a distribution $R_{u,v}$ of paths between $u$ and $v$. 
Choosing paths obliviously

**Intuition:** each driver asks an *offline* mobile navigation app to produce a path (given a starting point $s_i$ and destination $t_i$).

**Formally:**

**Definition**

Given $G = (V, E)$, an *oblivious routing* $R$ is a collection of $|V|^2$ distributions $R = \{R_{u,v}\}_{u,v \in V}$, where for each pair of nodes $u, v \in V$ we have a distribution $R_{u,v}$ of paths between $u$ and $v$.

*How do drivers pick a path:* Each driver going from $s$ to $t$ samples a random path from $R_{s,t}$ and drives along it.
Choosing paths obliviously

**Intuition:** each driver asks an offline mobile navigation app to produce a path (given a starting point $s_i$ and destination $t_i$).

**Formally:**

**Definition**

Given $G = (V, E)$, an **oblivious routing** $R$ is a collection of $|V|^2$ distributions $R = \{R_{u,v}\}_{u,v \in V}$, where for each pair of nodes $u, v \in V$ we have a distribution $R_{u,v}$ of paths between $u$ and $v$.

How do drivers pick a path: Each driver going from $s$ to $t$ samples a random path from $R_{s,t}$ and drives along it.

**Obliviousness:** All drivers sample from the same $R$. Note: path chosen by driver $i$ is independent (i.e. **oblivious**) of the path chosen by driver $j$. 
Question—informal

Given $G$, does there exist a single oblivious routing $R(G)$ whose makespan is $\tilde{O}(1)$-competitive with offline optimum for all demands?
Question—informal

Given $G$, does there exist a single oblivious routing $R(G)$ whose makespan is $\tilde{O}(1)$-competitive with offline optimum for all demands?

Impossible! No single oblivious routing suffices! [Räcke, Thesis, ’03]

- 1 demand $\rightarrow$ send along short path. Makespan $= 1$.
- $M$ demands $\rightarrow$ send along long paths. Makespan $= \sqrt{M}$. 
Perfect obliviousness is a very strict constraint. We can get the next best thing!
Perfect obliviousness is a very strict constraint. We can get the next best thing!

**Our result—informal**

For every graph $G$ and $\text{OPT} > 0$, there exists a **single** oblivious routing $R(G, \text{OPT})$ whose makespan is $\tilde{O}(\text{OPT})$ for all demands whose offline makespan is $\tilde{\Theta}(\text{OPT})$. 
Perfect obliviousness is a very strict constraint. We can get the next best thing!

**Our result—informal**

For every graph $G$ and $\text{OPT} > 0$, there exists a **single** oblivious routing $R(G, \text{OPT})$ whose makespan is $\tilde{O}(\text{OPT})$ for all demands whose offline makespan is $\tilde{\Theta}(\text{OPT})$.

The above (near-) oblivious routing typically good enough.

- Guess $\text{OPT}$.
- Drivers sample a path from $R(G, \text{OPT})$ and drive along it.
- If successful, we are done! Otherwise, double $\text{OPT}$.
- Guessing $\text{OPT}$ loses an insignificant $\tilde{O}(1)$ factor.
1 Motivating problem

2 Background
   - Prior work

3 Main technical ideas

4 Conclusion
Oblivious makespan minimization (also called oblivious congestion + dilation or $C + D$ minimization):
Oblivious makespan minimization (also called oblivious congestion + dilation or $C + D$ minimization):

- Hypercubes has $O(\log n)$-competitive makespan-minimizing oblivious routings.
  - “Valiant’s trick”
  - Each drivers $s \rightarrow t$ picks a uniformly random intermediate $m$.
  - Greedy route $s \rightarrow m$ and greedy route $m \rightarrow t$. 
Oblivious makespan minimization (also called oblivious congestion + dilation or $C + D$ minimization):

- Hypercubes has $O(\log n)$-competitive makespan-minimizing oblivious routings.
  - “Valiant’s trick”
  - Each drivers $s \rightarrow t$ picks a uniformly random intermediate $m$.
  - Greedy route $s \rightarrow m$ and greedy route $m \rightarrow t$.

- Similarly, expanders.

- Grids, fat trees, etc.
[Aspnes et al., 2006] titled “Eight open problems in distributed computing”:

Another important open problem is to find classes of networks in which oblivious routing gives $C+D$ [congestion + dilation] close to the off-line optimal... Such a result have immediate consequences in packet scheduling algorithms.

It seems like our result for all graphs $G$ was missed.

- In spite of being a prominent open problem and special graphs having received considerable attention.
- Probably due to the impossibility result.
- Simply showing the existence is quite technically involved.
1 Motivating problem

2 Background

3 Main technical ideas
   • Barrier: tree-based routings do not suffice
   • Solution: Partial tree embeddings

4 Conclusion
Definition (Tree-based routing $R$)

There is a collection of trees $T_1, \ldots, T_k$. Each demand $s, t$ picks a random tree and routes along it.

Barrier: tree-based routings do not suffice
Barrier: tree-based routings do not suffice

**Definition (Tree-based routing $R$)**

There is a collection of trees $T_1, \ldots, T_k$. Each demand $s, t$ picks a random tree and routes along it.

All previously considered constructions of oblivious routings were tree-based.

**Barrier**

There exists a graph $G$ such that there exists no $\tilde{O}(1)$-competitive tree-based routing.
Partial trees: different trees embed different sets of nodes.

Idea: Partial tree distributions can support “routing with errors” [in the paper: $D^{(1)}$-routers].
Partial trees: different trees embed different sets of nodes.

Idea: Partial tree distributions can support “routing with errors” [in the paper: $D^{(1)}$-routers].

Theorem

For any graph $G$ and OPT, there is a distribution over partial tree embeddings such that 50% of all demands that can be routed in $\tilde{O}(OPT)$ time are routed in $\tilde{O}(OPT)$ time.

Note: if the source $s$ or $t$ are not in tree, this is an “error”.

Error correction: one can fully eliminate errors with a complicated scheme described in the paper.
Motivating problem

Background

Main technical ideas

Conclusion
  • Connections with other areas
Application: Universally-optimal distributed algorithms
(original motivation)

- Problem: distributed minimum spanning tree, SSSP, min-cut...
- Goal: an algorithm that is as fast as possible for a given network $G$ (up to polylogs).
- We get [HWZ, STOC’21]: if the network $G$ is known in advance (but not the input!), there is a single algorithm that is fast as possible on all networks.
- Open question: efficient construction of hop-constrained oblivious routings $\implies$ a single distributed algorithm that is optimal on all networks.
- Connection: Many problems are (up to polylogs) equivalent to simple pairwise communication problems.
Connections with other areas 2/2

**Bigger picture:** Bi-criteria optimization of congestion and dilation.
Generally very interesting but very hard questions.
Cross-disciplinary. More research, better understanding, and new tools are needed.

First few results of this kind:
**Bigger picture:** Bi-criteria optimization of congestion and dilation.

Generally very interesting but very hard questions.
Cross-disciplinary. More research, better understanding, and new tools are needed.

First few results of this kind:
- Tree embeddings for hop-constrained network design [HHZ, STOC’21]
  - General-purpose tree embeddings for problems with hop-constraints.
  - Bi-criteria guarantees for: Steiner tree, Steiner forest, group Steiner tree, ...
Bigger picture: Bi-criteria optimization of congestion and dilation.
Generally very interesting but very hard questions.
Cross-disciplinary. More research, better understanding, and new tools are needed.

First few results of this kind:
- Tree embeddings for hop-constrained network design [HHZ, STOC’21]
  - General-purpose tree embeddings for problems with hop-constraints.
  - Bi-criteria guarantees for: Steiner tree, Steiner forest, group Steiner tree, ...
- Network Coding Gaps for Makespan minimization: [HWZ, FOCS’20]
  - How much does network coding help vs. routing in communication?
- Your next application?
Connections with other areas 2/2

**Bigger picture:** Bi-criteria optimization of congestion and dilation. Generally very interesting but very hard questions. Cross-disciplinary. More research, better understanding, and new tools are needed.

First few results of this kind:
- Tree embeddings for hop-constrained network design [HHZ, STOC’21]
  - General-purpose tree embeddings for problems with hop-constraints.
  - Bi-criteria guarantees for: Steiner tree, Steiner forest, group Steiner tree, ...
- Network Coding Gaps for Makespan minimization: [HWZ, FOCS’20]
  - How much does network coding help vs. routing in communication?
- Your next application?

Thank you!