Tight Bounds for Online Graph Partitioning
Monika Henzinger, Stefan Neumann, Harald Räcke, Stefan Schmid
The Problem
Online Graph Partitioning

Problem Definition

- Initially there is an empty graph with $n$ vertices
- Vertices are assigned to $\ell$ servers
- Each server has capacity for $(1 + \varepsilon) \cdot k$ vertices, where $k = n/\ell$
- Adversary inserts edges into the graph
- Vertices of connected components **must** be assigned to the same server
- Algorithm can re-assign components
Online Graph Partitioning

Problem Definition

- If **ALG** moves a connected component with $s$ vertices, it has to pay cost $s$

- **OPT** knows all edge insertions in advance and can move to final configuration at minimum cost

- **Goal:** Minimize the (strict) competitive ratio
  
  \[
  \frac{\text{cost}(\text{ALG})}{\text{cost}(\text{OPT})}
  \]
Online Graph Partitioning

- Introduced by Henzinger et al. (SIGMETRICS’19)
- Applications:
  - Resource allocation in the cloud
    - servers = datacenters
    - vertices = workloads
    (e.g., communicating virtual machines)
  - Implementing distributed union—find data structures
Our Results
Tight Randomized Algorithm

- We obtain a randomized algorithm with competitive ratio $O(\log \ell + \log k)$
- We derive a matching lower bound of $\Omega(\log \ell + \log k)$
  - Our bounds are asymptotically tight
- Exponentially better than the $O(\ell \cdot \log k \cdot \log \ell)$-competitive algorithm in Henzinger et al. (SIGMETRICS’19)
Tight Deterministic Algorithm

- Deterministic algorithm with competitive ratio $O(\ell \cdot \log k)$
- And matching lower bound of $\Omega(\ell \cdot \log k)$
- The bounds are asymptotically tight
- If $\epsilon > 1$ (i.e., servers can store $> 2k$ vertices), we get a (tight) competitive ratio of $O(\log k)$

$n$ vertices
$\ell$ servers
server capacity = $(1 + \epsilon)k$, where $k = n/\ell$
Technical Overview
General Problematic

- Reminder:
  - Adversary inserts edges
  - Vertices of connected components must be assigned to the same server
  - Goal: minimize the number of vertex moves

- What should we do when an edge is inserted?
  - If both components are on the server: do nothing
  - If components are on different servers: need to move vertices
If Components Are on Different Servers

- Connecting two components $C_1$ and $C_2$ might trigger a cascade:
  - Not enough server capacity to move $C_1$ to server of $C_2$ (and vice versa)
  - We have to move other components as well
- Which components shall we move?
Which Components Shall We Move?

• We build an ILP based on the server capacities and component sizes
• The ILP solution reveals how we shall assign the components to the servers
• We re-solve the ILP after each edge insertion

Sensitivity Analysis:
If components of sizes $|C_1|$ and $|C_2|$ get connected, the ILP re-assigns components of size at most $O(|C_1| + |C_2|)$

• Are we done?

\[
\begin{align*}
\text{min} & \quad \sum_{(r,m): r \not\subseteq m} x(r,m) \\
\text{s.t.} & \quad \sum_{(r,m)} x(r,m) r_i / \delta \geq V_i / \delta \quad \text{for all } i \\
& \quad \sum_r x(r,m) = Z_m \quad \text{for all } m
\end{align*}
\]
Is Sensitivity Analysis Enough?

• Unfortunately, no
• We identify two different types of components:
  • *easy* components are “costly” for OPT, OPT has to pay $\Omega(|C|)$
  • for *hard* components, OPT might have no cost at all
• Sensitivity analysis works for easy components, but it is too costly for hard components
• We can afford to move easy components, but we have to avoid moving hard components

$$\text{min } \sum_{(r,m):r \not\in C} x(r,m)$$
$$\text{s.t. } \sum_{(r,m)} x(r,m) \frac{r_i}{\delta} \geq V_i / \delta \text{ for all } i$$
$$\sum_r x(r,m) = Z_m \text{ for all } m$$
How to Deal With Hard Components?

- **First approach:**
  Can we make sure we never move hard components?
  ➡ No. Our lower bounds show that sometimes we must move hard components

- **Our approach:**
  Interleave (standard) ILP solving and **manual ILP solving**
  ➡ If the merged component is easy, use the standard ILP and sensitivity analysis
  ➡ If the merged component is hard, manually maintain an optimal ILP solution without moving any components

\[
\begin{align*}
\min \sum_{(r,m): r \not= m} x_{r,m} \\
\text{s.t.} \quad \sum_{(r,m)} x_{r,m} r_i / \delta & \geq V_i / \delta \quad \forall i \\
\sum_{r} x_{r,m} & = Z_m \quad \forall m
\end{align*}
\]
Summary
Tight Bounds for Online Graph Partitioning

- **Online graph partitioning:** Store $n$ vertices across $\ell$ servers while edges are inserted into the graph and connected components must be placed on the same server.

- Tight randomized algorithm, $\Theta(\log \ell + \log k)$-competitive.

- Tight deterministic algorithm, $\Theta(\ell \cdot \log k)$-competitive.

- **Open Problem:** Remove exponential dependency on $\epsilon$ in competitive ratio.

\[
\begin{align*}
\min \quad & \sum_{(r,m) : r \leq m} x(r,m) \\
\text{s.t.} \quad & \sum_{(r,m)} x(r,m) r_i / \delta \geq V_i / \delta \quad \text{for all } i \\
& \sum_{r} x(r,m) = Z_m \quad \text{for all } m
\end{align*}
\]