# OBLIVIOUS ROUTING

## AN EXAMINATION OF VARIOUS TECHNIQUES OF MODELING, CONSTRUCTION & ANALYSIS

TUBA YILMAZ
RAJIV A IYER

PROF. DANA RANDALL
CS 6550 (DESIGN & ANALYSIS OF ALGORITHMS)
NOV 5, 2009
GEORGIA TECH.

# Overview

- Definitions
- Motivations
- Big Picture View
- Research Work (Old & New)
- Formulation as an LP & Solving using Ellipsoid Algorithm (Azar Et Al. STOC 2003)
- Online Version - multiple demand matrices over several days - Convex Optimization Method (Bansal Et Al. SPAA 2003)
- Hierarchical Decomposition (Racke FOCS 2002)
- Oblivious Routing on Geometric Networks (Busch Et Al. SPAA 2005)

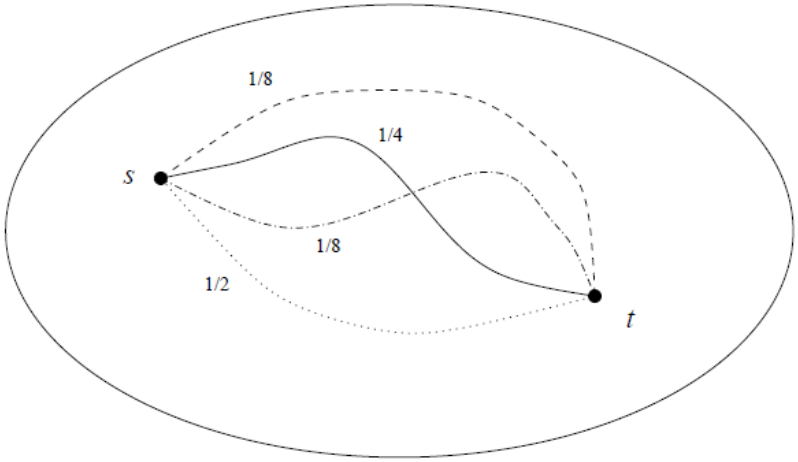
# Oblivious Routing - Definitions

**Basic Definition:** *Oblivious routing is the problem of picking a routing between each pair of nodes (or a set of flows), without knowledge of the traffic or demand between each pair, with the goal of minimizing the maximum congestion on any edge in the graph.*

**Definition (In terms of Probability):**

*An oblivious routing strategy is specified by a path system P and a function w assigning a weight to every path in P. w has the property that for every source-destination pair (s; t), the system of flow paths $P_{s;t}$ for (s; t) fulfills*

$$\sum_{q \in Ps;t} w(q) = 1$$



# Oblivious Routing: Motivation

- Tremendous expansion of the Internet – millions of nodes interconnected in a complex web & thousands of nodes added each day.
- Ideal but impractical to route optimally considering current load on the entire network (considering the astronomical scales)
- Hence, Oblivious Routing – the trick of selecting routes based on a probability distribution over all possible paths.
- Probability of a link to be selected is correlated with the bandwidth of that link.
- Oblivious Routing is memory-less.

# Oblivious Routing: Benefits/Merits

- Can be implemented easily in a Distributed scenario and therefore Scalable
- Doesn't need global co-ordination
- Appropriate for dynamic packet arrivals (i.e. online arrival of streaming packets)

# Big Picture: Major Research Works (Classic)

**CLASSIC/OLD RESULTS:**

- Kaklamanis Et Al. (SPAA 1990): Established tight lower bounds of $\Omega(\sqrt{N}/d)$ on deterministic oblivious routing on Hyper-cubes.
- Aspnes, Azar Et Al. (STOC 1993): General logarithm factor bounds on competitive ratio for online algorithms w.r.t. load balancing problems
- **Valiant, Brebner (STOC 1981):** Proposed/Examined Parallel Algorithm for Routing permutations in a Hypercube
- **Borodin, Hopcroft (STOC 1982):** Among other things, established a lower bound on the number of overlapping paths due to a permutation, which they used to establish $\Theta(\sqrt{n})$ bounds on congestion.

# Big Picture: Major Research Works (Recent)

**RECENT RESULTS:**

- Maggs Et Al. (FOCS 1997) gave an algorithm with $C = O(C^* \cdot d \cdot \log n)$ & also established lower bound on 'C' as: $\Omega(C^* \cdot \frac{1}{d} \cdot \log n)$ - these results are for d-dim mesh networks
- **Racke (FOCS 2002)** – Existential Result: $O(\log^3 n)$ (gen. undir. N/W or arbitrary graphs & w.r.t. EDGE congestion) – Hierarchical Decomposition Method (non-polynomial time construction algorithm)
- Bienkowski, Racke Et Al. (SPAA 2003): Polynomial Time construction algorithm for Hierarchical Decomposition Method.
- **Azar Et Al. (STOC 2003)**: Polynomial Time Optimal Oblivious Algorithm (constructive) using Ellipsoid Algorithm to solve an LP formulation.
- **Bansal Et Al. (SPAA 2003):** Online Version with Optimal Oblivious Routing over multiple demand matrices over multiple days. Modeled & solved using Convex Optimization.
- **Busch Et Al. (SPAA 2005):** Various strategies for Oblivious Routing on Geometric Networks (i.e. networks lying in Eucledian plane).
- Hajiaghayi, Racke Et Al. (STOC 2005): Proposed algorithm with competitive bound of $O(\log^2 n)$ w.h.p. for oblivious routing in dir. Graphs with random demands & proved Obl Rout not possible in $o(\log(n)/\log(\log(n)))$ with constant probability.

# Graph Formulation: Informal Definitions

- **Input Sequence:** A sequence of online routing requests σ to route messages between the nodes have to be served by specifying a path connecting the nodes.

- **Absolute load:** For an edge *e* ϵ E for an input sequence, for a given algorithm, it is the **amount of data that is transferred across *e* when σ is processed** by the algorithm.

- **Relative load:** For an edge *e* ϵ E for an input sequence σ , for a given algorithm, it is the **absolute load in the link divided by its bandwidth**.

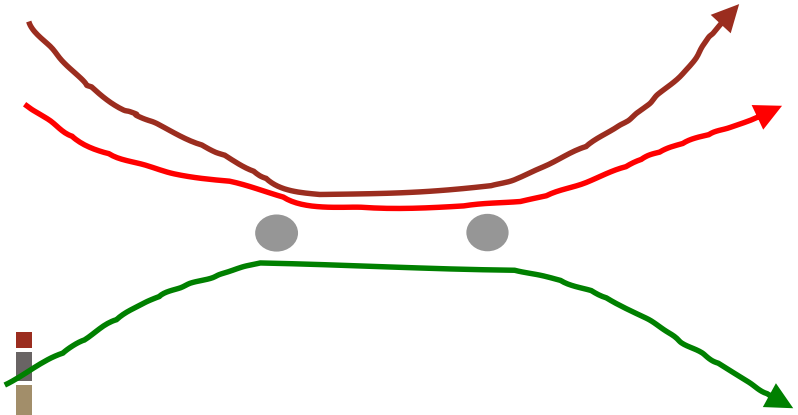
- **Congestion:** It is the **maximum over the relative loads** of all edges in the network

- **Algorithm Goal:** **Design an (efficient) algorithm to minimize congestion.**

# Edge Congestion & Node Congestion

**Edge congestion**

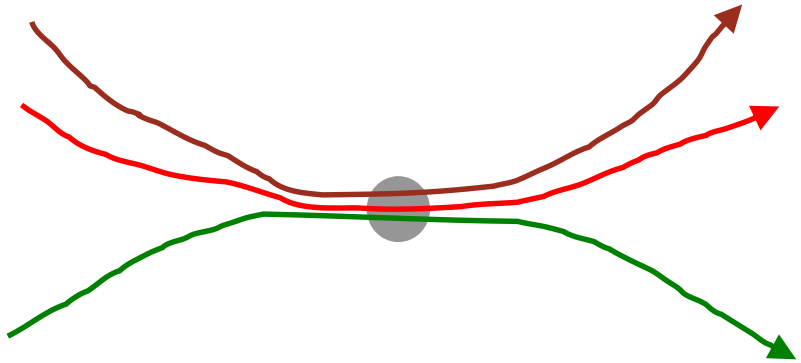
$C_{edge}$



maximum number of
paths that use any edge

**Node congestion**

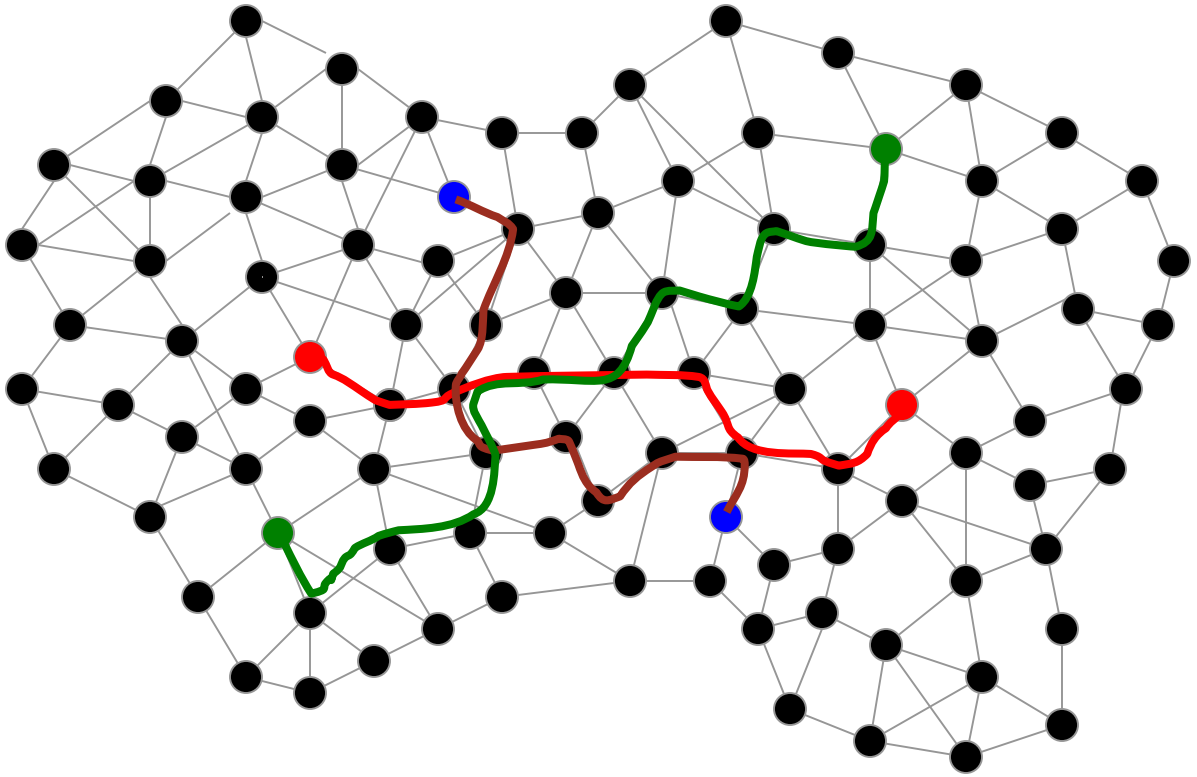
$C_{node}$



maximum number of
paths that use any node

# Oblivious Routing – Sample Graph Demo

Each packet path choice is independent of other packet path choices



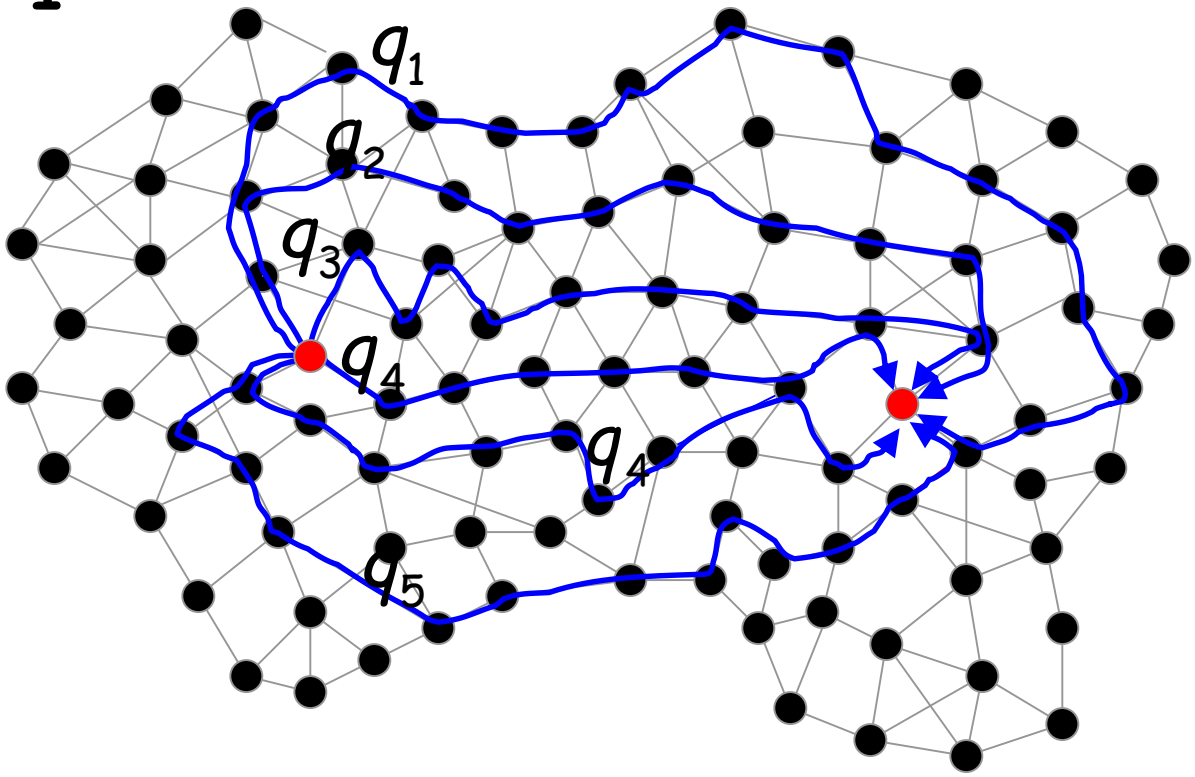
# Oblivious Routing – Sample Graph Demo

Probability of choosing a path: $\Pr[q_i]$

$$\sum_{i=1}^{k} \Pr[q_i] = 1$$

Path choices:

$q_1, \ldots, q_k$



# MODELLING AS MULTI COMMODITY FLOW AND SOLVING USING ELLIPSOID ALGORITHM

## "OPTIMAL OBLIVIOUS ROUTING IN POLYNOMIAL TIME" AZAR ET AL (STOC 2003)

# Formulation as a MCFP: Constraints & Definitions

- A routing from i to j = unit flow from i to j, say $f_{ij}$ (e)
- Routing subject to linear constraints:

$$\forall e \in E \; \forall i \forall j \neq i: \quad f_{ij}(e) \geq 0$$

$$\forall i \forall j \neq i: \quad \sum_{e \in \mathrm{OUT}(i)} f_{ij}(e) - \sum_{e \in \mathrm{IN}(i)} f_{ij}(e) = 1$$

$$\forall k \forall i \neq k \forall j \neq k, i: \quad \sum_{e \in \mathrm{OUT}(k)} f_{ij}(e) - \sum_{e \in \mathrm{IN}(k)} f_{ij}(e) = 0$$

- Flow on an edge given a demand matrix 'D':

$$\mathrm{FLOW}(e, \mathbf{f}, \mathbf{D}) = \sum_{ij} D_{ij} * f_{ij}(e).$$

- Congestion on an edge, given a demand matrix 'D':

$$\mathrm{EDGE\text{-}CONG}(e, \mathbf{f}, \mathbf{D}) = \frac{\mathrm{FLOW}(e, \mathbf{f}, \mathbf{D})}{c(e)}$$

- Overall Congestion:

$$\mathrm{CONGESTION}(\mathbf{f}, \mathbf{D}) = \max_{e \in E} \mathrm{EDGE\text{-}CONG}(e, \mathbf{f}, \mathbf{D})$$

# Formulation as a MCFP: Constraints & Definitions

- Given D, the minimum congestion possible is "opt(D)", which is:

$$\text{minimize } Z \text{ such that}$$

$$\mathbf{f} \text{ is a routing}$$

$$\forall e \in E,\ \text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq Z.$$

$$\forall e \in E$$
$$\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq z\text{OPT}(\mathbf{D})$$

- Performance Ratio for a given 'D': congestion(f, D)/opt(D)
- Oblivious Performance Ratio of a routing 'f':

$$\text{OBLIV-PERF-RATIO}(\mathbf{f}) = \sup_{\mathbf{D}} \frac{\text{CONGESTION}(\mathbf{f}, \mathbf{D})}{\text{OPT}(\mathbf{D})}$$

- Optimal Oblivious Performance Ratio:

$$\text{OBLIV-OPT}(G) = \min_{\mathbf{f}} \text{OBLIV-PERF-RATIO}(\mathbf{f})$$

- For General Graphs: OBLIV-OPT(G) ϵ $O(n^2)$
- For Undirected Graphs: OBLIV-OPT(G) ϵ O(polylog(n))

# LEMMA: RESTRICTION OF DEMAND MATRIX

**LEMMA:** There exists an (exponential size) set $V(H_1)$ of demand matrices such that:

- $\forall \mathbf{D} \in V(H_1)$ has **OPT(D) = 1** & coefficients of size polynomial in ***n*** and ***REP(C)***
- New Constraint: $\forall e \in E$, $\forall \mathbf{D} \in V(H_1)$, EDGE-CONG(e, **f**, **D**) ≤ z

**LEMMA PROOF OUTLINE:**

- Scaling 'D' has no effect on: $\frac{\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})}{\text{OPT}(\mathbf{D})}$

- We therefore normalize 'D' to be restricted to: $H_1 = \{\mathbf{D} | \text{OPT}(\mathbf{D}) \leq 1\}$
- $H_1$ is a polyhedron on n(n−1) dimensional space, defined by a polynomial number of inequalities, with coefficients in the set {±1, c(e) e ∈ E}
- $V(H_1)$ contains the vertices of the polyhedron $H_1$. Since the polyhedron has poly(n) constraints and variables, with coefficients of size rep(C), each vertex has representation of size poly(n, rep(C)).
- For any routing f and an edge e, $\arg\max_{\mathbf{D} \in H_1} \text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})$ includes a vertex of H1.

# SEPARATION ORACLE/ELLIPSOID ALGORITHM

- A binary search on 'T', a scalar value in the interval: $T \in [1, n^2]$

**Separation Oracle.**

- **Input:** A network $G$, capacities $c(e)$, and a routing $\mathbf{f}$.
- **Output:**
  - $\text{OBLIV-PERF-RATIO}(\mathbf{f})$
  - A demand matrix $\mathbf{D} \in V(H_1)$ and an edge $e$, such that

$$\text{OBLIV-PERF-RATIO}(\mathbf{f}) = \frac{\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D})}{\text{OPT}(\mathbf{D})}$$

If $\text{OBLIV-PERF-RATIO}(\mathbf{f}) \leq T$, then $\mathbf{f}$ is a feasible point of $\text{LI}(T)$ and the algorithm terminates. Otherwise,

$$\text{EDGE-CONG}(e, \mathbf{f}, \mathbf{D}) \leq T$$

is a constraint of $\text{LI}(T)$ violated by $\mathbf{f}$.

- Algorithm works by solving |E| instances of LP. For each edges, it solves: $\text{OLI}(e, G, \mathbf{f})$ whose objective is to locate $\mathbf{D} \in H_1$ and $\text{Maximize EDGE-CONG}(e, \mathbf{f}, \mathbf{D})$
- It is evident: $\text{OBLIV-PERF-RATIO}(\mathbf{f}) = \max_{e \in E} \text{OLI}(e, G, \mathbf{f})$

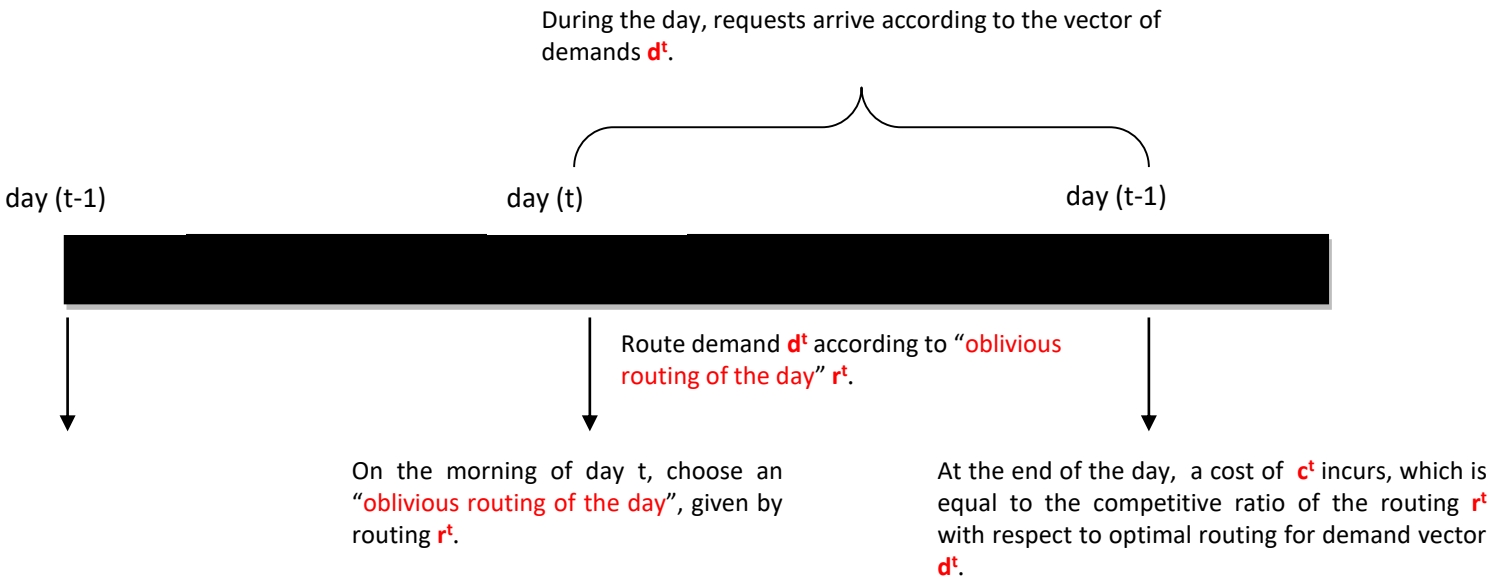
- It is well known that (at least one) of the maxima of a linear objective function over a polyhedron are obtained on a vertex of the polyhedron and that polynomial time LP algorithms can obtain such a vertex maximum.

# CONSIDERING ONLINE VERSION OF THE PROBLEM AND SOLVING USING CONVEX OPTIMIZATION

## "ONLINE OBLIVIOUS ROUTING " BANSAL ET AL (SPAA 2003)

# Formulation as an Online Game

- Network where requests are to be routed obliviously.
- The routing can be changed from day to day.




- **Goal**: minimize the average competitive ratio over all days.

# Framework and Notation

$(r_{i,j})$: *a route for commodity* $(i,j) \in V^2$ *is a unit flow from vertex i to vertex j*

$r = (r_{i,j})_{(i,j)\in V^2}$ *is a routing*

$d(i,j)$: *demand vector for* $(i.j) \in V^2$

$f_{d,r}(e) = \sum_{(i,j)\in V^2} d(i,j) r_{i,j}(e)$ *flow on an edge* $e \in E$

$C_d(r) = \max_{e\in E} f_{d,r}(e)$ *congestion of a routing r, given a demand vector d*

$COST_d(r) = \frac{C_d(r)}{\min_{r'} C_d(r')}$ *cost incurred by a routing r on demand vector d*

$\rho = argmin_r \max_d COST_d(r)$ *minimax optimal routing*

*The oblivious routing problem is to find a routing ρ that achieves the minimum possible cost over all demand vectors. Azar et al developed a polynomial time algorithm to find a routing for general networks on a given time t.*

*Online oblivious routing problem minimizes* $\sum_t COST_{d^t}(r^t)$ *where t dentoes each time step t*

$D = \{d^t\}$, *let* $r_D{}^*$ *(static optimal routing for demand sequence D ) minimize* $\sum_t COST_{d^t}(r^t)$.

# Online Convex Programming Notation

*The greedy projection algorithm:*

*Start with an arbitrary vector* $x_0 \in F$ *(feasible region).*

*At each step* $t+1$*, the algorithm first perfoms a gradient descent.*

*If point lies outside* $F$*, project it back to* $F$*, by picking a point closest to this point.*

$$y^t = x^t - \eta_t \nabla g^t(x^t)$$

$$x^{t+1} = \operatorname*{argmin}_{x \in F} \delta(x, y^t)$$

Let $\kappa(d^t) = \min_r C_{d^t}(r)$. *Consider the demand vector* $\overline{d^t} = \dfrac{d^t}{\kappa(d^t)}$.

*Note that optimal congestion on the demand vector* $\overline{d^t}$ *is exactly* 1.

*Flow is reduced by* $\kappa(d^t)$*, the cost remains same* $COST_{d^t}(r^t) = COST_{\overline{d^t}}(r^t)$.

$e^t = \operatorname{argmax}_{e \in E} f_{\overline{d^t}, r}(e)$ *maximum congested edge at time* $t$

$$c^t{}_{e,i,j} = \begin{cases} \overline{d^t}(i,j), & \text{if } e^t = e \\ 0, & \text{otherwise} \end{cases}$$

$COST_{d^t}(r^t) = c^t \cdot r^t$ *cost of algorithm at step* $t$

# Greedy Projection Algorithm

- Apply greedy projection algorithm to the problem (F, c).

1. *Pick* $r_0 \in F$ *arbitrarily*

2. *At step* $t+1$

(a) *Compute* $c^t$ *from* $d^t$ *using* $c^t{}_{e,i,j}$ *equation.*

(b) *Let* $y^t = r^t - \eta_t c^t$ *with* $\eta_t = \frac{2}{\sqrt{t}}$.

(c) *Pick* $r^{t+1} \in F$ *to be a* $(1+\epsilon_t)$ *approximation to* $\log_{x \in F} \|x - y^t\|^2$

*using the Projection algorithm defined below, where* $\epsilon_t = \frac{1}{t}$.

*Theorem: After* $\frac{n^6}{\epsilon^2}$ *steps, the Greedy projection algorithm is* $(1+4\epsilon)-$ *competitive with respect to the optimal static routing.*

# Semi-definite formulation and Ellipsoid Algorithm

***Semi − definite program***

$\min t \; subject\ to\ x \in F$

$$\begin{bmatrix} I & (x-y) \\ (x-y)^T & t \end{bmatrix} \geq 0$$

***Theorem***: *Let $K$ be a convex body in $R^n$ with $B(r) \subseteq K \subseteq B(R)$, for some $0 \leq r \leq R$, where $B(r)$ is a ball of radius $r$ around the origin.*

*Assume that we have a separation oracle for $K$.*

*Let $c \in R^n$ and $\varepsilon > 0$ be given. Then, we can find $x \in K$ with $c^T x \geq c^T z - \varepsilon, \forall\, z \in K$, using the ellipsoid algorithm, with the number of calls to the oracle computation time bounded by a polynomial in $\log\left(\frac{R}{r}\right), \log\left(\frac{1}{\varepsilon}\right)$, and $n$.*

# MODELLED AS MULTI COMMODITY FLOW & HIERARCHICAL DECOMPOSITION INTO TREE

## "MINIMIZING CONGESTION IN GENERAL NETWORKS" RACKE (FOCS 2002)

# Basic Idea Concept

- Given Network – Weighted Graph: $G = (V, E)$ with $|V| = n$ nodes and $|E|$ edges
- Each e in E has an associated bandwidth capacity function $b : E \rightarrow \Re^+$
- **Centralized** Routing Algorithm – Global Authority
- Initial Phase – Construction of Decomposition Tree: $T_G = (V_t, E_t)$ with $b_t : E_t \rightarrow \mathbb{R}^+$
- Each Routing Request σ processed in two steps (Bi-Simulation):
  - **Step 1: σ in G is converted to σ in $T_G$**
    - **Obvious** Routing used to obtain a path in $T_G$ (routing in a tree is trivial)
    - σ in G processed with congestion 'C' ⇒ σ in $T_G$ processed with congestion 'C'
  - **Step 2: Path in $T_G$ converted to Path in G**
    - Achieved using "simulation" – tweaked to obtain good competitive ratio
    - σ in $T_G$ processed with congestion '$C_t$' ⇒ σ in G processed w.h.p. congestion 'c' such that: **congestion in G < c $C_t$ + K** (K being a constant)
    - Each v in V may require simulation of multiple nodes in $T_G$ ( Randomized Mapping)
  - Let $\pi_1 : V \rightarrow V_t$ & $\pi_2 : V_t \rightarrow V$ be mapping in Steps 1 & 2, then we must ensure: $\pi_2(\pi_1(v)) = v \ \forall \ v \in V$

# Competitive Ratio & Laminar Set Systems

**Competitive Ratio:**

- Let us assume the following notations and/or situations:
  - there exists $c_t$-competitive online algorithm for routing in $T_G$
  - $C_{opt}(G)$ be the Optimal Congestion in G
  - $C_{opt}(T_G)$ be the Optimal Congestion in $T_G$
  - $C_{onl}(T_G)$ be the Congestion using Online Algorithm in $T_G$
  - $C_{\text{onl}}(G)$ be the Congestion using Online Algorithm in G
- Competitive Ratio Summary for Step 1:

$$C_{opt}(G) \geq C_{opt}(T_G) \geq \frac{1}{c_t} C_{onl}(T_G) - K_t$$

- Competitive Ratio Summary for Step 2:

$$C_{onl}(G) \leq c \cdot C_{onl}(T_G) + K \leq c \cdot (c_t \cdot C_{opt}(G) + K_t) + K$$

**Decomposition Tree – Laminar Set System:**

- Given G = (V , E) & each $v \in V_t$ of $T_G = (V_t, E_t)$ ⟺ a subset of V
- **Subsets of V form a Laminar Set System,** which is defined as:
  *"Two subsets X & Y are called intersecting if neither X\Y, Y\X & XY are empty. A set system is called laminar if it contains no intersecting pair. A Laminar Set System of subsets of V is 'complete' if it contains the set V and all sets {v}, $v \in V$"*

# Decomposition Tree & Laminar Set Systems

- Let ***S*** be a Laminar Set System for the set V. Then:

1. Each set $S \in \boldsymbol{S}$ corresponds to a node $v_t$ in $V_t$; such a set $S$ is called the cluster corresponding to $v_t$ and to identify $S$ using $v_t$, we use the notation $S_{v_t}$.

2. Two nodes $u_t$ and $v_t$ are connected in $T_G$ if $S_{u_t} \subset S_{v_t}$ or $S_{v_t} \subset S_{u_t}$. To satisfy the tree property, we also require that there exists no $S \in \boldsymbol{S}$ such that $S_{u_t} \subset S \subset S_{v_t}$ or $S_{v_t} \subset S \subset S_{u_t}$.

3. The bandwidth of the edge $(u_t, v_t)$ is the total bandwidth of the edges connecting the nodes of $S_{v_t}$ to $V \backslash S_{v_t}$.

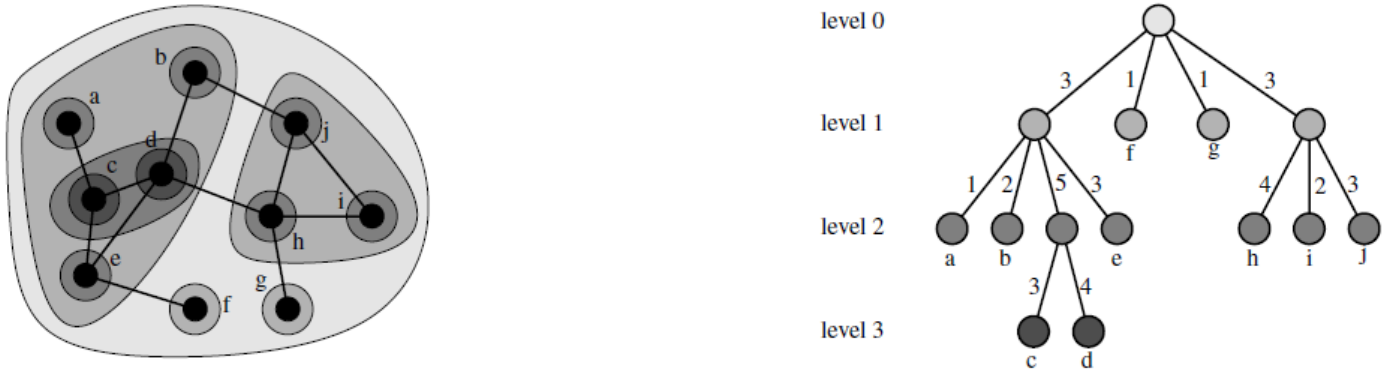



**Figure 1. A set system for a graph and the associated decomposition tree. Edge labels in the right figure indicate the bandwidth of the respective edge. (Edges in the left figure have bandwidth 1.)**

- **One to One Relation between Nodes in 'G' and leaf nodes in $T_G$**

# Quantitative Parametric Definitions

***Basic Parameters:***

- Total Bandwidth Available between nodes of X, Y subsets of V:
$$cap(X,Y) := \sum_{x \in X, y \in Y} b((x,y))$$
- Total bandwidth of edges leaving 'X' (subset of V): $\text{out}(X) = \text{cap}(X, \overline{X})$
- Bandwidth of edge $(u_t, v_t) \in E_t$ defined as: $b_t((u_t, v_t)) = \text{out}(S_{v_t})$

**Quality of Bi-Simulation Parameters:**

- **Height** of Tree $T_G$ denoted as: $h(T_G)$, a level 'l' ∈ {0,1,2,...., $h(T_G)$}
- The subset $V_t^l \in V_t$ denotes the set of level 'l' nodes in $V_t$
- **Weight Function** $w_l(X)$ evaluates cumulative bandwidth of all nodes adjacent to nodes in subset X but which are not connecting nodes of the same cluster of any $V_t^l$:
$$w_l(X) := cap(X,V) - \sum_{v_t \in V_t^l} cap(X \cap S_{v_t}, S_{v_t})$$
- **Bandwidth Ratio** of $v_t \in V_T$ defined as:
$$\lambda_{v_t} := max_{U \subset S_{v_t}, |U| \leq \frac{|S_{v_t}|}{2}} \{ \frac{out(U)}{cap(U, S_{v_t} \backslash U)} \}$$

Ratio between capacity of edges that leave a set U to those that connect U with the rest of the cluster.

# Parametric Definitions (contd.) & Important Theorem

- For some level 'l' node $v_t \in V_T$, Weight Ratio $\delta_{v_t}$ defined as:

$$\delta_{v_t} := max_{U \subset S_{v_t}, |U| \leq \frac{|S_{v_t}|}{2}} \{ \frac{w_{l+1}(U)}{cap(U, S_{v_t} \backslash U)} \}$$

- Bandwidth Ratio & Weight Ratio for the Tree TG defined as:

$$\lambda(T_G) = \max_{v_t \in V_t} \{\lambda_{v_t}\} \quad \& \quad \delta(T_G) = \max_{v_t \in V_t} \{\delta_{v_t}\}$$

**IMPORTANT THEOREM/RESULT:**

**Theorem 9** *For any graph $G = (V, E)$ with $n = |V|$ nodes there exists a decomposition tree $T_G$ that has height $h(T_G) = O(\log n)$, maximum bandwidth-ratio $\lambda(T_G) = O(\log n)$ and maximum weight-ratio $\delta(T_G) = O(\log n)$.*

**Proof.** For the following proof we define $\lambda := 20 \cdot \log n + 1$ and $\delta := 24 \cdot \lambda$. We construct a decomposition tree $T_G$ with $\lambda(T_G) \leq \lambda$ and $\delta(T_G) \leq \delta$. We say that a subset $S \subset V$ fulfills the bandwidth-property and the weight-property iff

$$\lambda \geq \max_{\substack{U \subset S \\ |U| \leq \frac{3}{4}|S|}} \left\{ \frac{\text{out}(U)}{\text{cap}(U, S \setminus U)} \right\} \quad \text{and} \quad \delta \geq \max_{\substack{U \subset S \\ |U| \leq |S|/2}} \left\{ \frac{\text{w}_{\ell+1}(U)}{\text{cap}(U, S \setminus U)} \right\},$$

**NOTE – Minor Result:**

- If $v_i, i \in \{1, ..., d\}$ are children of level 'l' node $v_t$, then:

$$U \subset S_{v_t},\ w_{l+1}(U) = \sum_{i=1}^{d} cap(U \cap S_{v_i}, \bar{S_{v_i}})$$

# Important Result (contd.) & Lemmas

**Important Lemmas:**

**Lemma 10** *Given a level $\ell$ cluster $S \subset V$ that fulfills the bandwidth property, it is possible to partition $S$ into sub-clusters $S_i$ with the following characteristics.*

1. *The sets $S_i$ are disjoint and form a partitioning of $S$, i.e., $\biguplus_i S_i = S$.*
2. *Each sub-cluster $S_i$ fulfills the bandwidth-property.*
3. *For each sub-cluster $S_i$ we have $|S_i| \leq \frac{5}{6} \cdot |S|$.*
4. *The cluster $S$ fulfills the weight-property, i.e., for each subset $U \subset S$ with $|U| \leq |S|/2$ we have $\sum_i \text{cap}(U \cap S_i, \overline{S_i}) \leq \delta \cdot \text{cap}(U, S \setminus U)$.*

**Lemma 11** *For any subset $R \subset V$ it is possible to partition $R$ into disjoint subsets $R_i \subset R$, such that each $R_i$ fulfills the bandwidth-property and $\sum_i \text{out}(R_i) \leq 2 \cdot \text{out}(R)$.*

# Algorithm Pseudo-codes: Bandwidth Partition

BANDWIDTHPARTITION($R$)

$\mathcal{P}_R := \{R\}$

**while** $\exists R_i \in \mathcal{P}_R$ not fulfilling the bandwidth-property **do**

find $U \subset R_i$ with $|U| \leq \frac{3}{4}|R_i|$
and $\lambda \cdot \text{cap}(U, R_i \setminus U) < \text{out}(U)$

$\mathcal{P}_R := \mathcal{P}_R \setminus \{R_i\}$

$\mathcal{P}_R := \mathcal{P}_R \cup \{U, R_i \setminus U\}$

**end**

**return** $\mathcal{P}_R$

**Figure 2. The algorithm BANDWIDTHPARTITION**

# Algorithm Pseudo-codes: Partition/Decomposition

PARTITION($S$)

compute $\frac{1}{6}$-balanced mincut $(A, B)$ of $S$

$\mathcal{P}_S := \{\{v\} \mid v \in S\}$

**while** $\exists U$ violating the sharpened weight condition **do**

/* i.e. $|U| \leq \frac{2}{3}|S|$ and $\exists Q \subset \mathcal{P}_S$ with $U = \biguplus_{S_i \in Q} S_i$ */

**for each** $S_i \in Q$ **do** $\mathcal{P}_S := \mathcal{P}_S \setminus \{S_i\}$

$\mathcal{P}_S := \mathcal{P}_S \cup$ BANDWIDTHPARTITION$(U \cap A)$

$\mathcal{P}_S := \mathcal{P}_S \cup$ BANDWIDTHPARTITION$(U \cap B)$

**end**

**return** $\mathcal{P}_S$

**Figure 3. The algorithm PARTITION.**

# Algorithm Pseudo-codes

BANDWIDTHPARTITION($R$)

$\mathcal{P}_R := \{R\}$
**while** $\exists R_i \in \mathcal{P}_R$ not fulfilling the bandwidth-property **do**
  find $U \subset R_i$ with $|U| \leq \frac{3}{4}|R_i|$
  and $\lambda \cdot \text{cap}(U, R_i \setminus U) < \text{out}(U)$
  $\mathcal{P}_R := \mathcal{P}_R \setminus \{R_i\}$
  $\mathcal{P}_R := \mathcal{P}_R \cup \{U, R_i \setminus U\}$
**end**
**return** $\mathcal{P}_R$

**Figure 2. The algorithm BANDWIDTHPARTITION**

- **Non-polynomial Run-time as it solves NP-hard problems.**
- Racke Et Al gave a polynomial time method in the paper ***"Practical Algorithm for constructing Oblivious Routing Schemes"***

- Bandwidth Property Satisfaction implies $\lambda(T_G) = O(log\ n)$
- Weight Property Satisfaction implies $\delta(T_G) = O(log\ n)$
- The Intersection with 1/6 Min-Cut implies size of cluster decreases & hence the bound $h(T_G) = O(\log\ n)$

PARTITION($S$)

compute $\frac{1}{6}$-balanced mincut $(A, B)$ of $S$
$\mathcal{P}_S := \{\{v\} \mid v \in S\}$
**while** $\exists U$ violating the sharpened weight condition **do**
  /* i.e. $|U| \leq \frac{2}{3}|S|$ and $\exists Q \subset \mathcal{P}_S$ with $U = \biguplus_{S_i \in Q} S_i$ */
  **for each** $S_i \in Q$ **do** $\mathcal{P}_S := \mathcal{P}_S \setminus \{S_i\}$
  $\mathcal{P}_S := \mathcal{P}_S \cup$ BANDWIDTHPARTITION$(U \cap A)$
  $\mathcal{P}_S := \mathcal{P}_S \cup$ BANDWIDTHPARTITION$(U \cap B)$
**end**
**return** $\mathcal{P}_S$

**Figure 3. The algorithm PARTITION.**

# Bounds on Competitiveness

- Expectation of Relative Load on an edge $e \in E$ due to simulation of tree strategy on G is:

$$E(L(e)) \leq O(\log n \cdot h \cdot \max\{\delta, \lambda\} \cdot C_t)$$

- Given a Graph G and a decomposition tree $T_G$, there exists an oblivious routing algorithm that is strictly $O(\log n \cdot h \cdot \max\{\delta, \lambda\})$ competitive w.r.t. congestion.
- This gives a bound of $O(\log^3 n)$ for the competitive ratio of the algorithm w.r.t. congestion.

# OBLIVIOUS ROUTING FOR GEOMETRIC NETWORKS (I.E. NETWORKS WITHIN EUCLEDIAN PLANE)

## "OBLIVIOUS ROUTING IN GEOMETRIC NETWORKS" BUSCH ET AL. (SPAA 2005)

# Competitive Ratio Results

- **This paper proposes a method that gives:**

**Constant stretch**

$$stretch = O(1)$$

**Small congestion**

$$C_{node} = O(C^*_{node} \log n)$$

degree

$$C_{edge} = O(\Delta \cdot C^*_{edge} \log n)$$

# REFER BUSCH ET AL PPT 1