Oblivious Routing

Input:
(wireless) network \( G(V,E) \)
- source target pairs \((s_i, t_i)\) of \( G \)
- for every source target pair \((s_i, t_i)\)
  a demand \( d_i \)

Goal: The fixed routing rule should obtain close to optimum congestion on any set of demands, i.e.

\[
\text{Minimize} \quad \max_{\text{demand-matrix } D} \left[ \frac{\text{congestion}}{\text{congestion opt}(D)} \right] \\
\text{For undirected graphs there are oblivious routing algorithms with competitive ratio of } O(\sqrt{n}) \text{ [Riecke, FOCS'02], } O(\sqrt{\frac{\ln n}{\ln \ln n}}) \text{ [Harren, Hildrum, Rad, SODA'05] and finally optimum solution in } O(n) \text{ by [Riecke, STOC'08].}
\]

For directed or even node weighted graphs, there are directed graphs in which every oblivious routing algorithm has competitive ratio in \( O(\sqrt{n}) \) [ACFKR, STOC'03] [HKRL, SODA'05]

Any oblivious routing scheme defines a unit flow from each node \( a \) which hence there is a node \( b \) which receives at least \( \frac{1}{k} \) units of flow from its direct neighbors on the first level (i.e., nodes \( a \) for \( i \in E(a) \)).

Suppose all neighbors of \( b \) send a demand of one and all other demands are zero.

Then the oblivious algorithm has congestion at least \( \frac{k}{2} \) at node \( b \).

However, optimum can route this demand with congestion 1 by using the paths \( a \rightarrow b \rightarrow \) for demands from node \( a \) and the paths \( a \rightarrow \) \( b \rightarrow t \) for demands to \( t \).

The proof follows immediately since \( k = \Omega(\sqrt{n}) \).
Exactly the same construction works for node-weighted graphs (all capacities are one except with an additional term)

- Though the ratio for directed graphs are $O(1/n)$ and $O(1/n)$ for general undirected graphs, but there is a polynomial time construction that gives the true optimal ratio for any (directed or undirected) network [ACEKR103]

- If demands are random from known distributions, the competitive ratio is in $O(lg^2 n)$ with high probability in directed graphs.

- Randomness does not help for undirected graphs (we have lower bound $O(1/n)$ in this case.

- Other goals such as maximizing throughput instead of minimizing congestion has also been considered [Awerbuch H., Karger, Kleinberg, Leighton, SODA 2005] or oblivious network design with general edge-functions [Gupta H., Rastek; SODA 2005] as we discussed before.

- R. Rask (FOCS 02) introduced a tree decomposition that aims at constructing a tree that does not approximate point-to-point distances in the input graph (like Bahl or FRSl's technique) but instead approximates the cut structure of the graph in the following sense: Given a concurrent multi-commodity flow problem (CMCF problem in which demands $u$ from $s$ to $t$ should be shipped simultaneously with minimum congestion (the maximum over all edges of the flow divided by the capacity of the edge) in graph $G$, the optimal solution of the corresponding flow problem in a decomposition tree has a lower congestion, and conversely, given a CMCF problem in a decomposition tree, the corresponding problem in $G$ can be solved with a congestion that is only a factor of larger (and it is constructible). These kinds of decomposition is called "Cut-decomposition."

- The model: $G$ with nodes $v$ where $|V| = n$ is given, we have a capacity/weight function $c$ on the edges, where $c(u,v) = c(v,u)$ since the graph is undirected. Assume $c(u,v) > 0$ (if there is no edge from $u$ to $v$) and $c(u,v) = 0$ if $(v,u) \in E$. 

---

Society for Industrial and Applied Mathematics

3600 University City Science Center · Philadelphia, PA USA 19104-2688 · +1-215-382-9800 · Fax +1-215-386-7999 · meetings@siam.org · www.siam.org
Decomposition Trees. A decomposition tree for the graph $G$ is a rooted tree $T = (V_T, E_T)$ whose leaf nodes correspond to nodes in $G$, i.e., there is a one-to-one relation between nodes in $G$ and leaf nodes in $T$. Whenever we use the concept of a decomposition tree for a graph $G$, we implicitly assume that we are also given an embedding of $T$ into $G$. This means there is a node mapping function $m : V_T \to V$ that maps tree nodes to nodes in the graph and because of the property of a decomposition tree this mapping function induces a bijection between leaf nodes of $T$ and nodes in $G$. We are also given a function $m_E : E_T \to E^*$ that maps an edge $e_T = (u_T, v_T)$ of $T$ to a path $P_{uv}$ between the corresponding end points $u = m(v_T)$ and $v = m(v_T)$. We also introduce functions $m^V : V_T \to V$ and $m_E : E_T \to E^*$, responsible for mapping from $G$ to $T$. $m^V$ outputs for a node $v \in V$ the leaf node in $T$ corresponding to $v$, and the function $m_E$ gives for an edge $e \in E(\{u,v\}) \subset E^*$ the (unique) shortest path in $T$ between $m^V(u)$ and $m^V(v)$.

For a multicommodity flow $f$ on a decomposition tree we use $m(f_T)$ to denote the multicommodity flow $f$ on a decomposition tree we use $m(f_T)$ to denote the multicommodity flow that is obtained by mapping $f_T$ to $G$ via the edge-mapping function $m_E$. Similarly, we define for a flow $f$ in $G$, $f(T)$ as the flow in $T$ obtained by mapping $f$ to $T$.

Note that we can add flows and scale them e.g., $m(a'(f_T + b'f_T)) = m(a'f_T) + m(b'f_T)$.

Given a decomposition tree $T$ for $G$ we define the capacity $c(u_T, v_T)$ of an edge $e_T = (u_T, v_T)$ as $c(u_T, v_T) = \sum_{w \in \text{path}(u_T, v_T)} c(w)$, where $V_{u_T}$ and $V_{v_T}$ denote the two partitions of $V$ induced by the cut corresponding to edge $e_T$.

Given a multicommodity flow in $G$, we want to compose its congestion in $G$ to its congestion in $T$.

Thm 4: Suppose you are given a multicommodity flow $f$ in $G$ with congestion $C_f$. Then the flow $m(f)$ obtained by mapping $f$ to some decomposition tree $T$ results in a flow in $T$ that has congestion $C_T \leq C_f$.

pf: Suppose an edge $e_T = (u_T, v_T)$ in the tree has congestion $C_T$. All traffic that traverse the cut in $G$ between $V_{u_T}$ and $V_{v_T}$. The total capacity of all edges over this cut is exactly $c(e_T)$. Hence by a simple averaging argument, one of these edges must have relative load at least $1/m$. This gives $C_T \leq C_f$. 
Given a decomposition tree with an embedding of this tree into graph $G$, we can ask for the load that is induced on a graph edge $e$ by this embedding, let

$$\text{load}(e) := \sum_{\text{absolute}} c(\text{let})$$

and

$$\text{load}_r(e) := \frac{\text{load}(e)}{\text{load}(\text{root})}$$

We are looking for a convex combination of decomposition trees such that for every edge the expected relative load is small, i.e.

$$\min B = \max \left\{ \sum_i \lambda_i \text{load}_{r i}(e) \mid \lambda_i \geq 0 \text{ and } \sum_i \lambda_i = 1 \right\}$$

**Thm 2** suppose we are given a convex combination of decomposition trees with maximum expected relative load $B$ and suppose that we are given for each tree $T_i$, a multi-commodity flow $f_i$ that has congestion $C$ in $T_i$. Then the multi-commodity flow $\sum_i \lambda_i f_i$ has congestion at most $B$ when mapped to $G$. (This is like FRT or Betke.)

**Pf:** Fix a tree $T_i$. Routing the flow $f_i$ in the tree generates congestion at most $C$ which means that the amount of traffic that is sent along an edge $e_i$ in $(u_i, v_i)$ is at most $C(e_i)$. Hence the total traffic that is induced on a graph edge $e$ when mapping $\lambda_i f_i$ to $G$ is at most $C(e) \text{load}_{r i}(e)$. Therefore the relative load induced on $e$ when mapping all flows $\lambda_i f_i$ is at most $\left( \sum_i \lambda_i \text{load}_{r i}(e) \right) \leq B$.

Ricke '08 shows that we can indeed obtain a convex combination of decompositions for which $B = O(\log n)$. The proof is not hard but involves use FRT trees for which $B = O(\log n)$, the proof is not hard but involves use FRT trees for which $B = O(\log n)$. The proof is not hard but involves use FRT trees for which $B = O(\log n)$. The proof is not hard but involves use FRT trees for which $B = O(\log n)$.
New Oblivious Routing Algorithm: The convex combination of decomposition trees defines a unit flow for every source-target pair, by combining for a pair \((u, v)\) the paths between \(u\) and \(v\) in trees \(T_i\), where the path from \(T_i\) is weighted with \(x_i\). This is the oblivious routing.

\[\textbf{pf:}\] Now given a demand vector that can be routed with congestion \(C\) in \(G\), routing it in a decomposition tree creates congestion less than \(C\) in any tree. Now mapping the flows from all decomposition trees back and thus scaling it by a factor \(x_i\), the thing that indeed we have done in our oblivious routing gives a solution in \(G\) with congestion at most \(B \cdot \max_i \{\text{const}(T_i)\} \leq B \cdot \text{const}(G)\) due to Thm 2. (Note that because of uniqueness of paths in trees all scaling by vectors \(x_i\) has been done automatically.)

Hence the oblivious routing scheme has competitive ratio \(O(\log n)\) as desired.

This convex combination of trees has several other applications like Barycentric result, e.g., for min bisection, sparsest cut, multi-cost routing, online multicast, etc. gives \(O(\log n)\) approximation.

Universal solutions: given a metric space, it is a total ordering of points of the space such that for any finite subset, the tour which visits these points in the given order is as close as possible to the optimal tour.

This is Universal TSP.

Universal solutions are motivated by oblivious routing through they were before.

Universal TSP (first) considered by Platzman and Bartholdi in SACTM 89.

For planar graphs and for plane graphs there is a lower bound \(\log n \cdot k\) for general graphs the lower bound is \(2(\log n)\) for expander graphs (GKSS '10).

**Upper bound is \(O(\log n)\)** [CHR '06]
Consider multi-commodity (uniform) buy-at-bulk in which there is a non-decreasing, monotone subadditive function \( f : B \rightarrow \mathbb{R}^+ \) for an edge \( e \) where \( f(b) \) is the minimum cost of cable installation with bandwidth \( b \) for an edge \( e \). Given a set of bandwidth demand pairs, install sufficient capacities with minimum total cost.

The uniform case has \( O(\log n) \) approximation by Barak (see the notes) (non-uniform was \( O(\log^2 n) \)). The oblivious version in which we decide the paths beforehand has \( O(\log n) \) algorithm. [GHR06]

Incremental solutions: given a notion of ordering on solutions of different cardinality, we give solutions for all values of \( K \) such that the solutions respect the ordering and such that for any \( K \), our solution is close in value to the value of an optimal solution of cardinality \( K \). For example, for \( K \)-median, \( K \)-MST (the ordering of \( n \) edges \( e_1, e_2, \ldots, e_{n-1} \) such that for any \( K \in \{1, \ldots, n\} \), the first \( K \) edges of the sequence span \( K \) vertices including \( 1 \) and \( n \)), \( K \)-set cover, which asks for a min-cost subcollection of \( C \) that covers at least \( K \)-elements. (outputs an ordering of sets such that for any \( K \in \{1, \ldots, n\} \), the minimal prefix of the sequence that covers \( K \) elements is a good approximation to \( K \)-set cover problem). There is \( O(1) \)-incremental solutions for \( K \)-median and \( K \)-MST and \( O(\log n) \) for \( K \)-set cover. [LNRW06]

Also, if people consider the version that we are given a set of important vertices called terminal and we want to find a solution universal whose competitive ratio only depends on \( K \), e.g., poly \( \log(K) \) for problems such as \( K \)-cut, \( K \)-multi-cut, etc. [Möllenkamp09]. He considers mainly the cut problems [what about the connectivity problems?]}