Maintaining a Constant Competitive Steiner Tree Online

Xu Shunyi

Syllabus

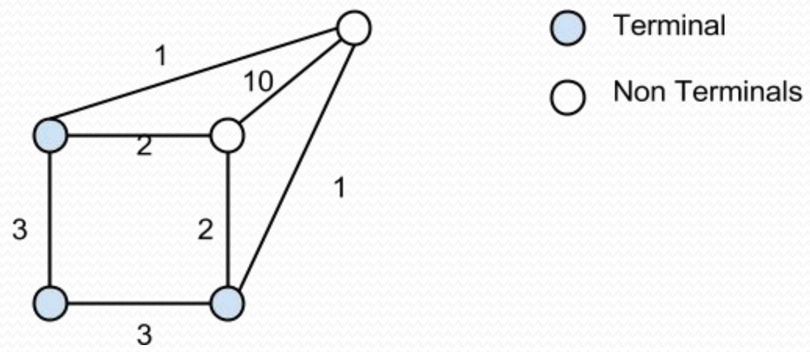
- Steiner Tree Problem(STP)
- Primal Dual of STP
- GGK Algorithm(STOC 2013)

Syllabus

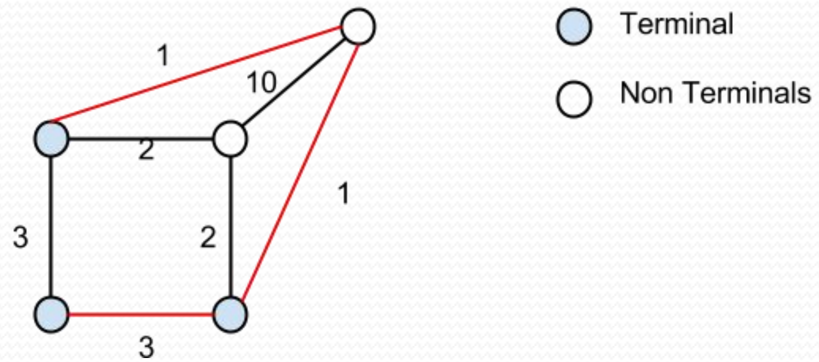
- Steiner Tree Problem(STP)
- Primal Dual of STP
- GGK Algorithm(STOC 2013)

Steiner Tree Problem

- Input:
  - undirected graph $G = (V, E)$;
  - non-negative edge costs $c : E \to R^+$;
  - terminal-set $R = \{s_1, \ldots, s_k\} \subseteq V$



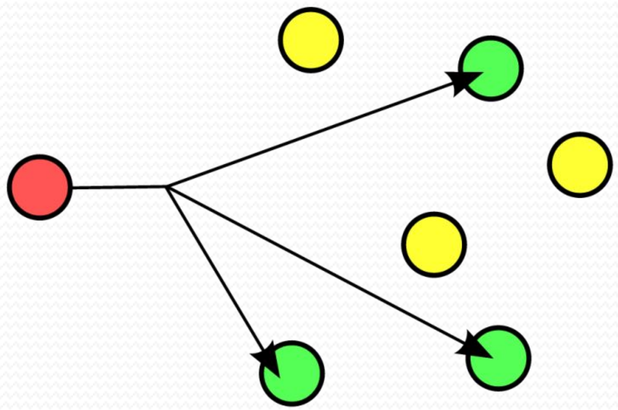
- Output
  - Compute a Spanning Tree on R that's of minimum total edge cost



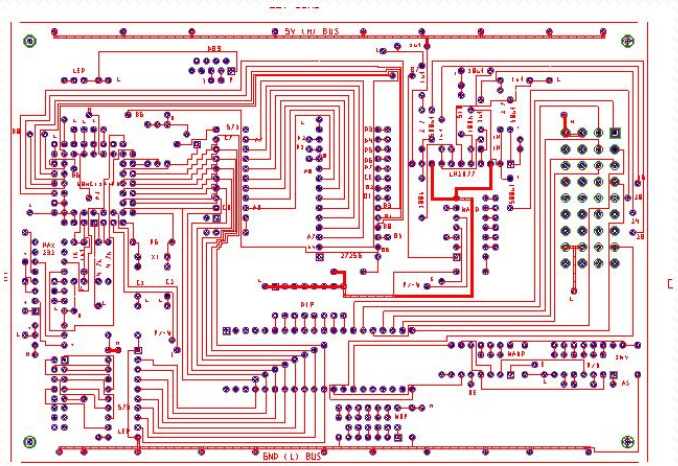
$C(T) = 5$

Steiner Tree Problem --Application

- Multicast Network Design



- Circuit Layout Design



Syllabus

- Steiner Tree Problem(STP)
- Primal Dual of STP
- GGK Algorithm(STOC 2013)

Primal Dual of STP

- Primal

$$\begin{aligned}\text{min} \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(U)} x_e \geq 1 \quad \forall \text{ Steiner cut } U \\ & x_e \geq 0 \quad \forall e \in E\end{aligned}$$

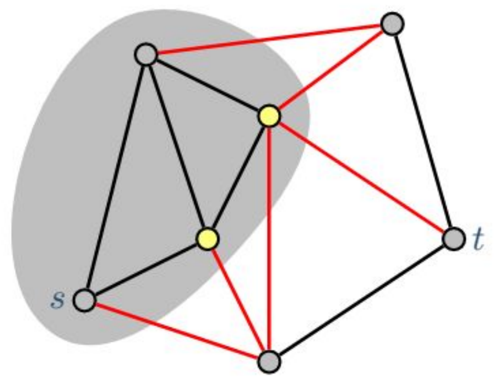
$\delta(U)$: Edges with exactly one endpoint in U.

- Dual

$$\begin{aligned}\text{max} \quad & \sum_{U} y_U \\ \text{s.t.} \quad & \sum_{U:\, e \in \delta(U)} y_U \leq c_e \quad \forall e \in E \\ & y_U \geq 0 \quad \forall U\end{aligned}$$

$\delta(U)$: Edges with exactly one endpoint in U.

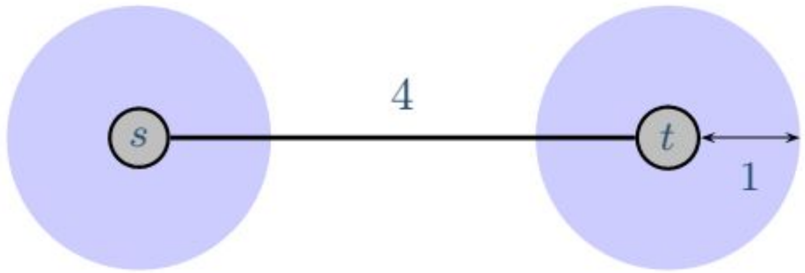
- Steiner Cut: Subset of nodes that separates at least one terminal pair (s, t) ∈ R



Can visualize y_U as disks around U with radius y_U.
Example: Terminal pair $(s,t) \in R$, edge (s,t) with cost 4

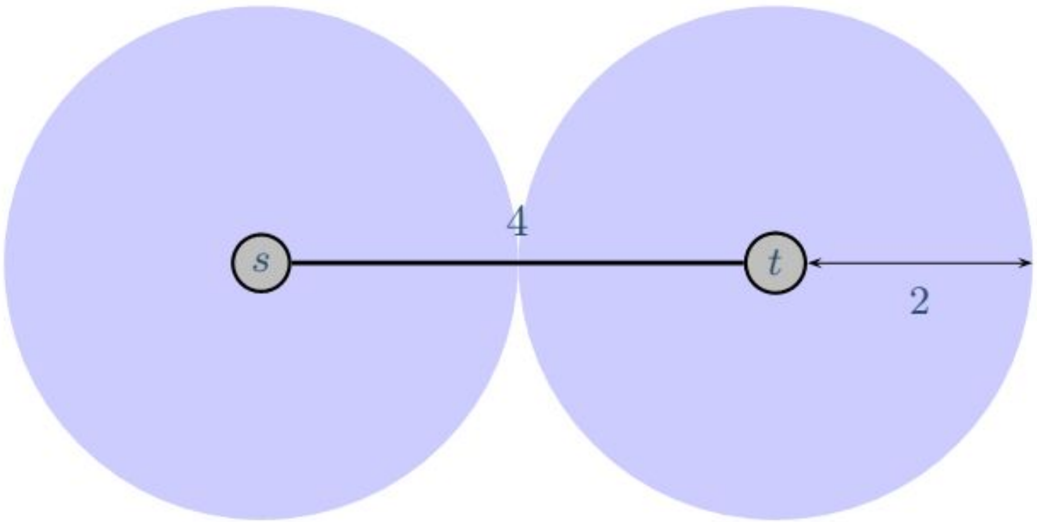
$y_s = y_t = 0$

Can visualize y_U as disks around U with radius y_U.
Example: Terminal pair $(s,t) \in R$, edge (s,t) with cost 4



$y_s = y_t = 1$

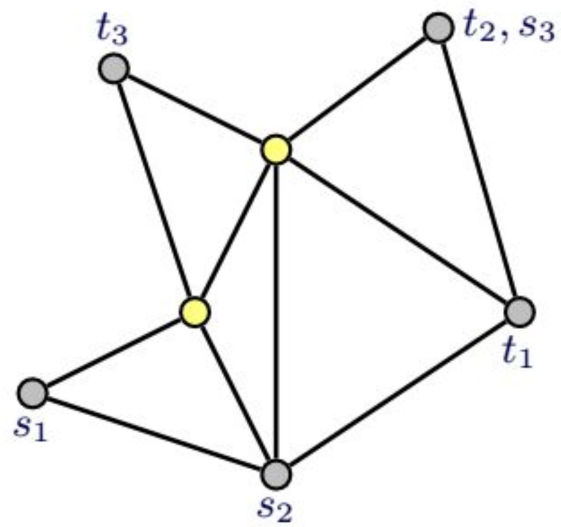
Can visualize y_U as disks around U with radius y_U.
Example: Terminal pair $(s,t) \in R$, edge (s,t) with cost 4



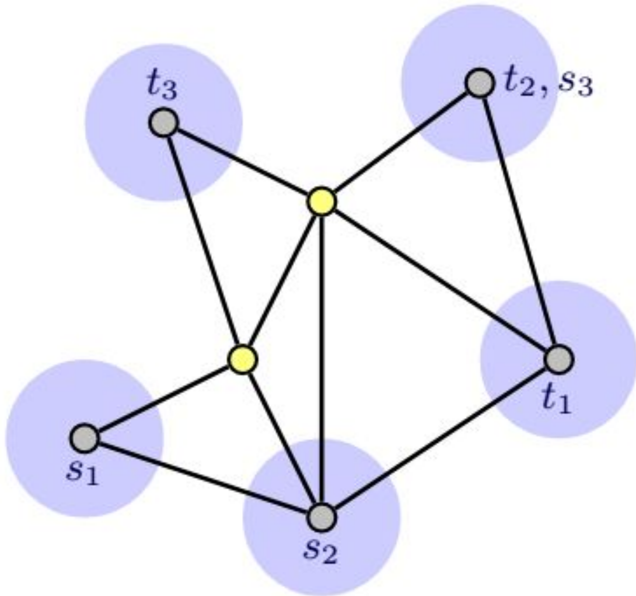
$y_s = y_t = 2$

Have: $y_s + y_t = 4 = c_{st}$. Edge (s,t) is tight.

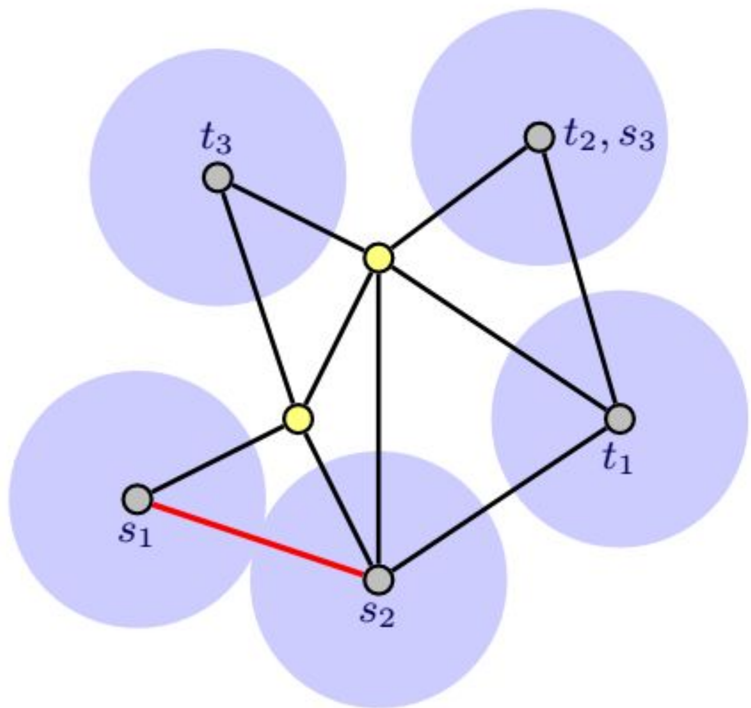
Algorithm grows duals of connected components.



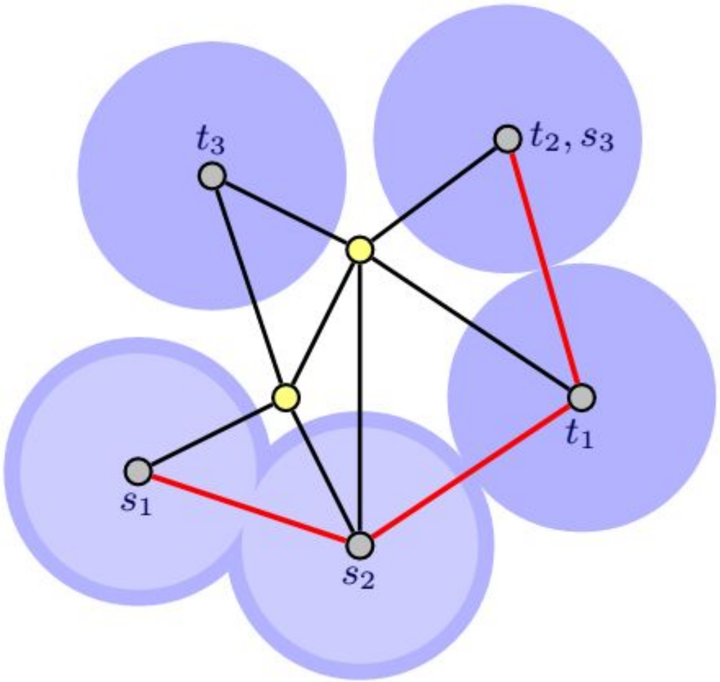
Algorithm grows duals of connected components.



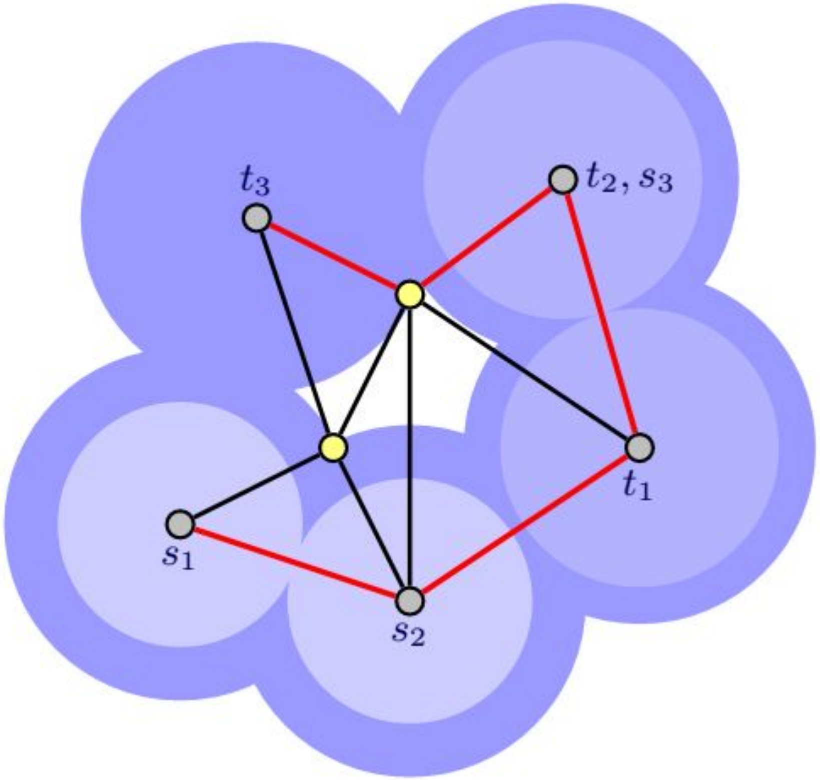
Algorithm grows duals of connected components.



Algorithm grows duals of connected components.



Algorithm grows duals of connected components.



Syllabus

- Steiner Tree Problem(STP)
- Primal Dual of STP
- GGK Algorithm(STOC 2013)

GGK Algorithm(STOC 2013)

- Related work:
  - [IW91]--O(log n) competive ratio, no swaps
  - [MSVW12, Ver12] –$O(1+\varepsilon)$ completive ratio and $O(1/\varepsilon \log 1/\varepsilon)$ swaps in amortized sense per vertex arrival.

- GGK claims:
  - constant competitive with ratio $C = 2\alpha^5/(\alpha-1)^2$ and K=2 α swaps per vertex arrival
  - differs from previous work in 1) establishing a constant competitive ratio and 2) make a constant swaps per vertex arrival instead of in an amortized sense

Algorithm Overview

For $i \geq 1$, when the i-th vertex arrives, we begin round i.

Round i involves three steps:

(a) running a clustering algorithm on the vertex set $[i]$ (all the terminals that have arrived by round i) which defines the rank function $\rho_i: [i] \to \mathbb{Z}_{\geq 0}$

(b) getting a virtual rank function ν_i from the actual rank function ρ_i

(c) constructing the tree T_i given the virtual rank function

Algorithm Overview

For $i \geq 1$, when the i-th vertex arrives, we begin round i.

Round i involves three steps:

(a) running a clustering algorithm on the vertex set $[i]$ (all the terminals that have arrived by round i) which defines the rank function $\rho_i: [i] \rightarrow \mathbb{Z}_{\geq 0}$

(b) getting a virtual rank function ν_i from the actual rank function ρ_i

(c) constructing the tree T_i given the virtual rank function

rank function ρ_i: [i] → Z≥0

- ξ_i(t): clusters in the i-th round, after phase t finishes.

- We start with ξ_i(0), which consists of trivial clusters {{0}, {1}, {2},…{i}}

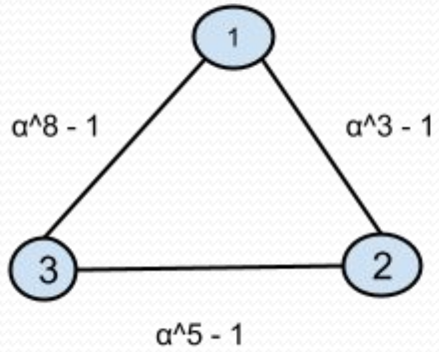
- At beginning of any t, if two clusters from ξ_i (t-1) satisfy d(C,C') < $2\alpha^{t+1}$, We merge them into a new Cluster

- We maintain the invariant: at end of t that two distinct cluster C, C' $\in$ ξ_i (t) satisfy d(C,C') >= $2\alpha^{t+1}$

rank function ρ_i: [i] → Z≥0

- For each cluster $C \in \xi_i(t)$, define C's leader as the vertex with least index.

- Define $\rho_i(x)$ of vertex x the largest t of which x is still the leader of its cluster.



time	Cluster
0	{1}, {2}, {3},
1	threshold = α^2, {1}, {2}, {3}
2	threshold = α^3, {1,2}, {3}
3	threshold = α^4, {1,2}, {3},
4	threshold = α^5, {1,2 ,3},

hence p(1) = ∞, p(2) = 1, p(3) = 3

Analysis

- Why do we want a rank function?
  - Rank is defined on vertex while cost is defined on edges.
  - If we can convert edge cost to vertex rank, we can analyze the SPT using its dual form.

- Define $Wt_j(f) = \sum_{L=1\ldots j} \alpha^{f(L)}$

- We claim that $Wt_i(p_i) <= OPT([i])/(\alpha-1)$

Proof

- Remember the dual of a Steiner Tree Problem

$$
\begin{aligned}
&\max \sum_{S} y_S \\
&\sum_{S:|S\cap\{j,l\}|=1} y_S \le d(j,l) \qquad \forall j,l \in [i] \\
&y_S \ge 0.
\end{aligned}
$$

- if we can find a feasible solution y such that $\sum_s y_s >= Wt_i(p_i)(\alpha\text{-}1)$, by weak duality, this will imply the claim.
- In fact we can find such a feasible set defined as: For each cluster $C \in \xi_i(t)$ which doesn't contain vertex 0 (a Steiner cut by nature), let $y_C = \alpha^t(\alpha - 1)$ for other steiner cut S, let $y_s = 0$;

$$\max \sum_{S} y_S$$

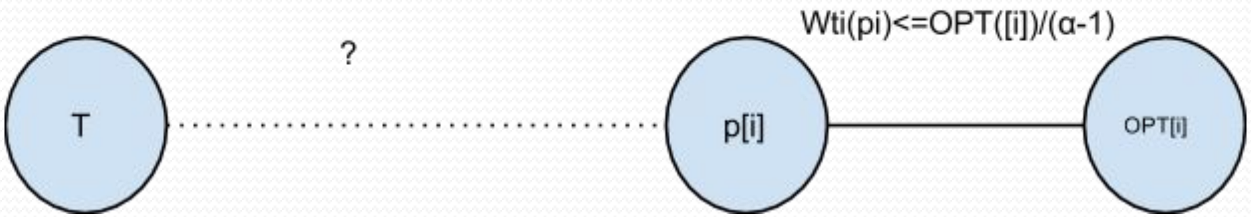
$$\sum_{S:|S\cap\{j,l\}|=1} y_S \leq d(j,l) \qquad \forall j,l \in [i]$$

$$y_S \geq 0.$$

- Check feasibility:
  - For any edge (j, l), let t be the last phase such that j and l lie in different clusters in $\xi_i(t)$
  - For all phases t'<=t, we contribute $2\alpha^{t'}(\alpha - 1)$ to LHS
  - We have LHS = $\sum_{t'=0}^{t} 2\alpha^{t'}(\alpha-1) = 2(\alpha^{t+1}-1) \leq d(j,l)$
- objective function:

$$\sum_{C} y_C \geq \sum_{j=1}^{i} \alpha^{\rho_i(j)}(\alpha-1) = \mathsf{Wt}_i(\boldsymbol{\rho}_i)\cdot(\alpha-1).$$

- Therefore we proved $\sum_s y_s \geq \mathrm{Wt}_i(p_i)(\alpha-1)$



Wti(pi)<=OPT([i])/(α-1)
?
T
p[i]
OPT[i]

Algorithm Overview

For $i \geq 1$, when the i-th vertex arrives, we begin round i.

Round i involves three steps:

(a) running a clustering algorithm on the vertex set [i] (all the terminals that have arrived by round i) which defines the rank function ρ_i: $[i] \rightarrow Z_{\geq 0}$

(b) getting a virtual rank function ν_i from the actual rank function ρ_i

(c) constructing the tree Ti given the virtual rank function

Algorithm Overview

For $i \geq 1$, when the i-th vertex arrives, we begin round i.

Round i involves three steps:

(a) running a clustering algorithm on the vertex set $[i]$ (all the terminals that have arrived by round i) which defines the rank function $\rho_i: [i] \to Z_{\geq 0}$

(b) getting a virtual rank function ν_i from the actual rank function ρ_i

(c) constructing the tree T_i given the virtual rank function

Algorithm Overview

For $i \geq 1$, when the i-th vertex arrives, we begin round i.

Round i involves three steps:

(a) running a clustering algorithm on the vertex set [i] (all the terminals that have arrived by round i) which defines the rank function ρ_i: $[i] \rightarrow Z_{\geq 0}$

(b) getting a virtual rank function ν_i from the actual rank function ρ_i

(c) constructing the tree T_i given the virtual rank function given an admissible function

Admissible function β

- Define $Init(j) = p_j(j)$, (i.e., the rank of j at round j)

- $\beta: [i] \rightarrow Z_{>=0}$ is admissible if $\beta(j) \in [p_i(j)...Init(j)]$ for all $j \in [i]$

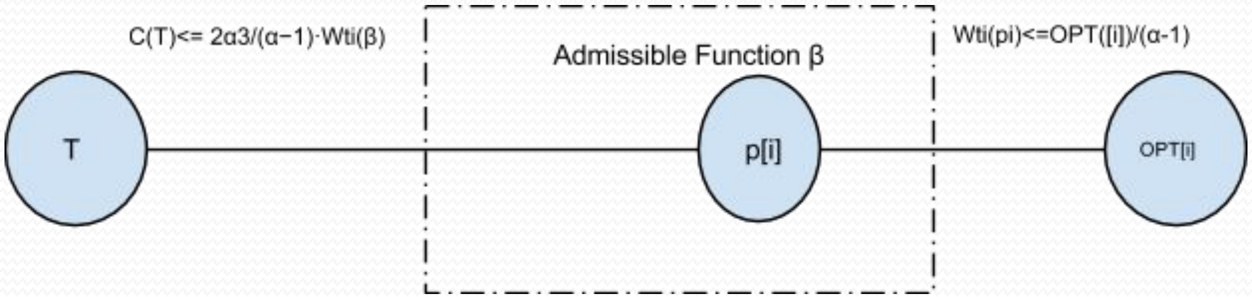
- Let's also define a tree valid with respect to β if we can partition the edge set E_T into sets E_T^1 , E_T^2,..., E_T^s, such that the following hold
  - Let $E_T^{<=L}$ denote E_T^1 U... U E_T^L. For any connected component of $E_T^{<=L}$, let j be the head of this component. Then we require $\beta(j) >= L$ (head means the vertex has biggest β)
  - Each edge in E_T^L has length at most $2\alpha^{L+1}$

Why need a generalized admissible function?

- Because we can show that the total cost of any tree T valid respect to β is at most **2 α3/(α-1) Wt$_i$(β)** based on the definition of admissible and valid tree T.

- Proof:
  - The cost of each $E_T^{\leq(l-1)}$ in E_T^l can be charged to the heads of components of
  - Any vertex j! E_T^l is charged $2\alpha^{l+1}$ me E_T^l only if $l \leq \beta(j) + 1,$
  - Each edge in is at most
  - Total cha

$$\sum_{l=1}^{\beta(j)+1} 2\alpha^{l+1} \leq 2\frac{\alpha^3}{\alpha - 1} \cdot \alpha^{\beta(j)}.$$



$$C(T) \le 2\alpha^3/(\alpha-1)^2 OPT([i])$$

Steps to build the tree

- Lemma in constructing valid tree in respect to β and β'
  - If T = ([i], E_T)is valid to β already, and β' differs from β only at vertex j*, then there's a tree T' = ([i], $E_{T'}$) that valid with respects to β' and $|E_{T'} \triangle E_T| = 2$ (△ means symmetric difference)
- Proof:
  - Let L* = β'(j*)+1. hence **β' (j*) < L***
  - Let edges sets union $\mathbf{E_{T'}^{<=L^*-1}} = \mathbf{E_T^{<=L^*-1}}$. Nothing changes.
  - For edge set $\mathbf{E_{T'}L^*}$, if j* is the head of some connected component C in $E_{T'}^{<=L^*}$, we may find **β' (j*) < L*,** a contradiction to the tree property β(j) >=L for all L

    Then we can find a vertex j in C' such that d(j, C)<= $2\alpha^{(L^*+1)}$ let $E_{T'}L^* = E_{T'}L^*$ U (j, c) where c is the vertex in C closest to j. Hence C and C' is connected, and the head of C' is the head of the new component which satisfy β'(j) = β(j) >=L
  - For edges $E_{T'}^L$ where L>L*. Build $E_{T'}^L$ to be same as E_T^L except for edges that connect two vertices in the same component, hence we will drop only the edge that connect C and C' in E_T edge
  - This is an edge swap because we replace the old edge connecting C and C' with a new edge (j, c)

Lemma →

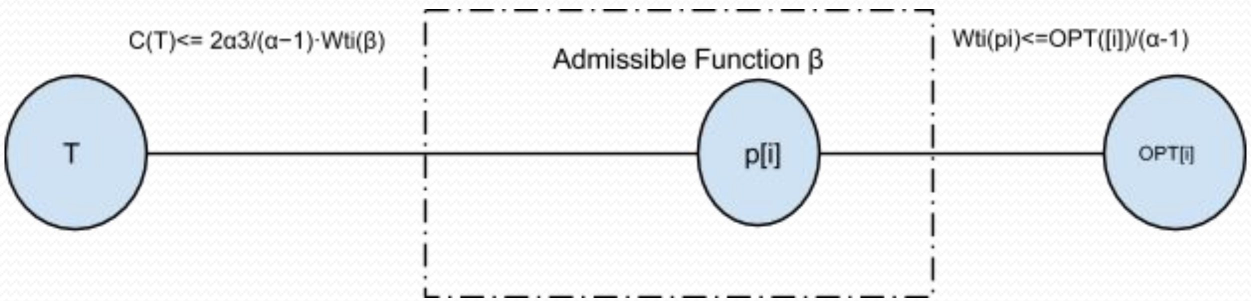
- Since rank function $\{p_i\}_i$ is inherently admissible, if use p_i as the algorithm function. We can do the following to make sure the built tree is valid.

  - At new arrival of vertex i, we first add a single edge from i to [i-1](a shortest path to closest vertex in [i-1])

  - then there are at most $|| p_i - p_{i-1}||_H$ different value in the rank between round i and round i-1. Using last lemma, we can aggregate at most $|| p_i - p_{i-1}||_H$ edge swaps, each for one pair of difference.

Analysis

- Using p_i directly as the admissible function, achieves constant competitive ratio.



$$C(T) <= 2\alpha^3/(\alpha-1)^2 OPT([i])$$

Analysis

- but… it incurs non-constant swaps per vertex arrival.

- Reason being that at each round i, there's $||p_i - p_{i-1}||$ swaps. Therefore, we need to restrict the number of swaps tolerable to a constant.

Algorithm Overview

For $i \geq 1$, when the i-th vertex arrives, we begin round i.

Round i involves three steps:

(a) running a clustering algorithm on the vertex set $[i]$ (all the terminals that have arrived by round i) which defines the rank function $\rho i: [i] \to Z \geq 0$

(b) getting a virtual rank function ν_i from the actual rank function ρ_i

(c) constructing the tree Ti given the virtual rank function

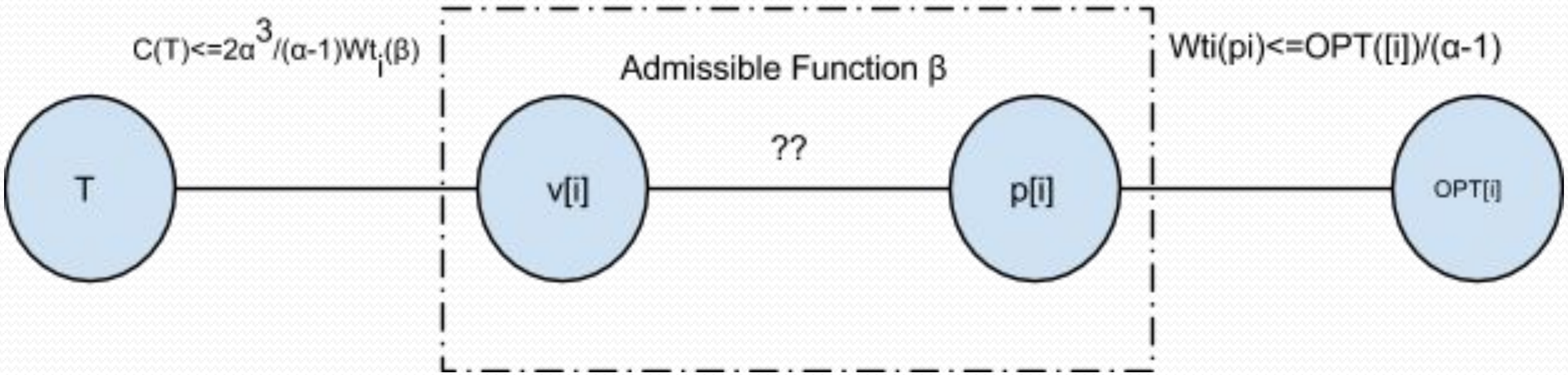
Virtual Rank Function

K is used as a bottleneck to limit swaps in a round. Besides, we know that the Cost of $T_i \approx \sum_{j>0} \alpha^{vi(j)}$ .We wanted to keep this as small as possible. A natural way to get v_i from v_{i-1} is to decrement those K vertices with highest $v_{i-1}(j)$

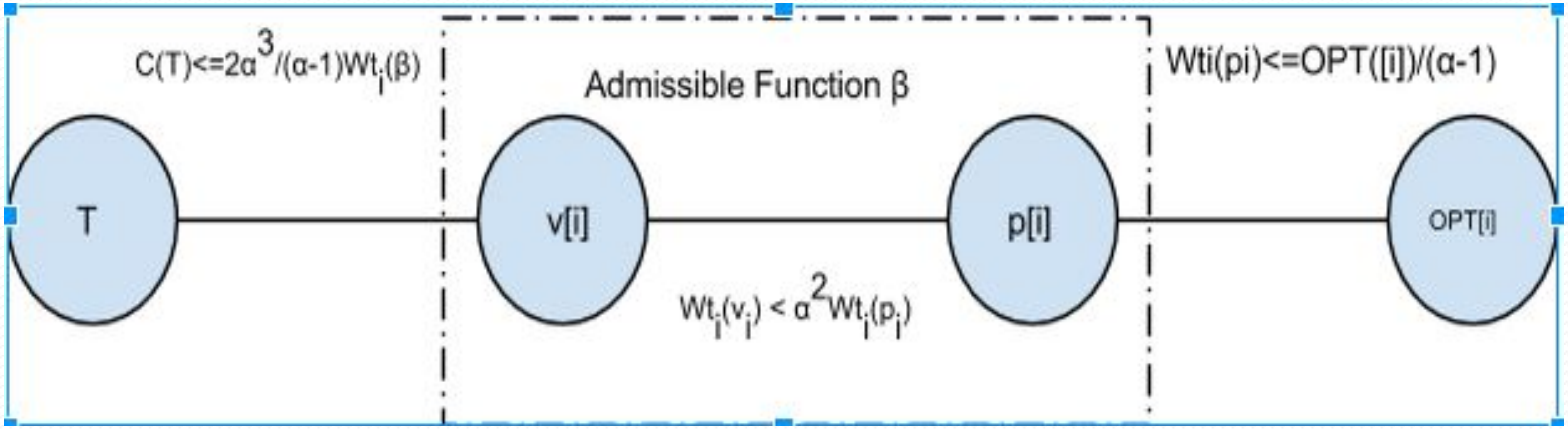
Virtual Ranks :

1. Initially, we just have the root vertex 0. Define $\boldsymbol{\nu}_0(0) = \infty$.
2. For $i = 1, 2, \ldots$
   (i) Run the clustering algorithm $\mathcal{R}_i$ to define the rank function $\boldsymbol{\rho}_i$.
   (ii) Set $\boldsymbol{\nu}_i(i)$ as $\texttt{Init}(i)$.
   (iii) Define $Q(i) = \{(j, k) \mid j \in [i-1], k \in [\boldsymbol{\rho}_i(j) \ldots (\boldsymbol{\nu}_{i-1}(j) - 1)]\}$.
   (iv) Let Q_K be the set of the K highest pairs (w.r.t. $\prec$) from $Q(i)$.
   (v) Define the first $i-1$ coordinates of $\boldsymbol{\nu}_i$ as follows:

$$\boldsymbol{\nu}_i(j) := \begin{cases} \boldsymbol{\nu}_{i-1}(j) & \text{if } (j, \star) \notin Q_K \\ \min\{k \mid (j, k) \in Q_K\} & \text{if } (j, \star) \in Q_K \end{cases}$$



C(T)<=2α3/(α-1)Wti(β)
Admissible Function β
Wti(pi)<=OPT([i])/(α-1)
T
v[i]
??
p[i]
OPT[i]



We can prove that $Wt_i(v_i) < \alpha^2 Wt_i(p_i)$.

And this completes our proof:
for any n:
1. $Cost(T_n) <= 2\alpha^3/\alpha\text{-}1\ Wt_n(v_n) <= 2\alpha^5/\alpha\text{-}1\ Wt_n(p_n) <= 2\alpha^5/(\alpha-1)^2\ opt([n])$

2. This is a **constant swap algorithm for each arrival round i**, since only K swaps are allowed(k coordinates of v_i and v_{i-1} differ by the definition of virtual rank function)


Q&A