IOE 691: Approximation & Online Algorithms

Lecture Notes: Online Dual Fitting

Instructor: Viswanath Nagarajan

Here we discuss another approach for analyzing online algorithms. A key aspect in proving performance guarantees is in obtaining a lower-bound (for a minimization problem) to the optimal value. For some problems, there are natural problem-specific lower bounds that can be used. However, a much more general method to get lower-bounds is through linear-programming (or convex programming) duality. Often, the online algorithms analyzed through this approach are very natural (e.g. greedy) and the key point lies in identifying and using a suitable lower-bound.

1 Load Balancing on Identical Machines

This is one of the earliest online problems to have been studied. There are m identical machines and n jobs with processing times $\{p_j\}_{j=1}^n$ that arrive over time. The algorithm needs to assign each job to a machine immediately upon arrival; this assignment cannot be changed later. The goal is to minimize the maximum load (total processing time) on any machine. This objective is also known as makespan.

Consider the natural greedy algorithm that assigns each arriving job to the machine that has the current smallest load. This algorithm is clearly online. We now analyze it using the following obvious lower-bounds:

$$\text{Average load } A := \frac{1}{m}\sum_{j=1}^{n} p_j, \quad \text{and} \quad \text{Maximum job } P := \max_{j=1}^{n} p_j.$$

Consider the job ℓ that finishes last in the greedy online schedule. Let T be the time at which ℓ starts processing. When job ℓ arrived, it was placed on the least loaded machine that had load of T. So all m machines must have load at least T, i.e., the total load on the machines is at least mT. On the other hand, the total load cannot be more than the total processing times of all jobs. So we have $T \leq \frac{1}{m}\sum_{j=1}^{n} p_j = A$. Finally, the makespan of the online schedule is the finish time of job ℓ, which is $T + p_\ell \leq A + \max_j p_j = A + P$. Thus, we obtain:

Theorem 1.1 *There is a 2-competitive algorithm for minimizing makespan on identical machines.*

2 Matching

In the online (bipartite) matching problem, we have a set L of "offline" nodes and a set R of "online" nodes. The offline nodes are known up-front and the online nodes arrive over time. When node $j \in R$ arrives, it reveals all edges from j to L. The goal is to match some of the online nodes to offline nodes so as to maximize the size of the matching. Crucially, the decision to match (or not) some online node j must be made immediately upon j's arrival. As an example, one can

view the offline nodes as different room-types in a hotel and the online nodes as customers who are interested in particular room-types.

We consider the obvious greedy algorithm. When $j \in R$ arrives, if there is any *unmatched* neighbor i of j then we match j to i. We will show that this has competitive ratio 2.

It is easy to see that the algorithm has competitive ratio at least 2. There are 2 offline nodes $\{i, i'\}$. The first online node j_1 has edges to both i and i'. If the online algorithm matches j_1 to i then the second online node j_2 has an edge to i alone. Whereas, if j_1 is matched to i', the second online node has an edge to i' alone. The optimal value is 2 and online only gets 1.

For the analysis we use the natural LP and its dual:

$$\begin{aligned} \max \sum_{(ij)\in E} x_{ij} \qquad & \qquad \min \sum_{i\in L} \alpha_i + \sum_{j\in R} \beta_j \\ \sum_{j\in R} x_{ij} \le 1, \quad \forall i \in L, \qquad & \qquad \alpha_i + \beta_j \ge 1, \quad \forall (i,j) \in E \\ \sum_{i\in L} x_{ij} \le 1, \quad \forall j \in R, \qquad & \qquad \alpha, \beta \ge 0. \\ x \ge 0. \qquad & \end{aligned} \tag{1}$$

Consider the following dual solution. Set $\alpha_i = 1$ (resp. $\beta_j = 1$) if node i (resp. j) is matched by the algorithm. Note that this is a feasible dual solution. Indeed, suppose (for contradiction) there is an edge $(i, j) \in E$ with $\alpha_i = \beta_j = 0$. In this situation, the greedy algorithm would have matched online node j to offline node i, contradicting our definition of dual values.

Now, consider the dual objective. We have $\sum_{i\in L} \alpha_i = \mathsf{ALG}$ the total number of matched edges. Similarly, $\sum_{j\in R} \beta_j = \mathsf{ALG}$. So the dual objective is $2\mathsf{ALG}$. By weak duality, it follows that the optimal LP value (which is an upper bound on the optimal matching) is at most $2\mathsf{ALG}$.

Theorem 2.1 *The greedy algorithm is 2-competitive for bipartite matching.*

3 Randomized Matching

We now revisit the online (bipartite) matching problem. Recall there there is a bipartite graph G with "offline" nodes L (known upfront) and "online" nodes R that arrive over time. For each $j \in R$, let $N(j) \subseteq L$ denote its neighbors. We consider the following simple randomized algorithm called *ranking*.

Choose a uniformly random permutation to index the offline nodes L. When node $j \in R$ arrives online, match j to its lower indexed unmatched neighbor in L (if any).

We will show that this algorithm has a expected competitive ratio $1 - \frac{1}{e}$. For the analysis, it will be convenient to use the following equivalent view of the algorithm. Each offline node $i \in L$ chooses a value $Y_i \in [0, 1)$ uniformly.[1] When node $j \in R$ arrives, it is matched to the unmatched node $i \in N(j)$ with minimum Y_i. We will use the standard primal and dual LPs (1). We will construct a *randomized* dual solution that is not always feasible, but feasible in expectation. In particular,

[1] We will assume throughout that the Y_i values are distinct, which happens with probability one.

we will show that (1) the expected dual objective is at most $\frac{e}{e-1}$ times the algorithm's value, and (2) using the expected value of each dual variable leads to a feasible dual solution.

Constructing the random dual solution. We set the dual variables as follows:

$$\alpha_i = \begin{cases} \frac{e^{Y_i}}{e-1} & \text{if } i \text{ matched} \\ 0 & \text{if } i \text{ unmatched} \end{cases}, \qquad \forall i \in L. \tag{2}$$

$$\beta_j = \begin{cases} \frac{e-e^{Y_i}}{e-1} & \text{if } j \text{ matched to } i \\ 0 & \text{if } j \text{ unmatched.} \end{cases}, \qquad \forall j \in R. \tag{3}$$

Note that β_j and α_i are random variables. Let ALG denote the size of the algorithm's matching; note that this is also random. We first show that ALG is at least $1-\frac{1}{e}$ time the dual objective.

Lemma 3.1 $\mathsf{ALG} \geq (1-\frac{1}{e}) \cdot \left(\sum_{i\in L} \alpha_i + \sum_{j\in R} \beta_j\right)$.

Proof: Let M denote the edges used the in the algorithm's matching. By definition of the dual variables,

$$\sum_{i\in L} \alpha_i + \sum_{j\in R} \beta_j = \sum_{(i,j)\in M} (\alpha_i + \beta_j) = \sum_{(i,j)\in M} \left(\frac{e^{Y_i}}{e-1} + \frac{e-e^{Y_i}}{e-1}\right) = \frac{e}{e-1}|M|.$$

This completes the proof as $\mathsf{ALG} = |M|$. ∎

Dominance and monotonicity properties. We now prove two important properties of the ranking algorithm that will be used in showing dual feasibility (in expectation). Fix any edge (i,j) where $j \in R$ and $i \in N(j) \subseteq L$. Fix the $Y_{i'}$ values for all $i' \in L \setminus i$. Consider the run of the algorithm on graph $G \setminus i$, i.e., with offline nodes $L \setminus i$ and the same sequence of online arrivals until node j. Note that this is a deterministic run because all the Y-values are fixed. Define

$$K = \begin{cases} Y_k & \text{if } j \text{ matched to } k \in L \setminus i \\ 1 & \text{if } j \text{ unmatched} \end{cases}.$$

Lemma 3.2 *Consider the run of the algorithm on graph G with the fixed $Y_{i'}$ values for $i' \in L \setminus i$. If $Y_i \leq K$ then node i gets matched.*

Proof: Consider the situation when node j arrives. If node i is already matched, the lemma is clearly true. If i is unmatched then the run of the algorithm (on graph G) so far must be *identical* to the run on graph $G \setminus i$. In particular, if $N \subseteq L \setminus i$ denotes the unmatched neighbors of j in the run on $G \setminus i$ then the unmatched neighbors of j in the run on G must be $N \cup i$. Hence, node j must get matched in graph G. We now consider two cases depending on the value of K:

- If $K = 1$ then j was unmatched in the $G \setminus i$ run, i.e., $N = \emptyset$. This implies that, for the run on graph G, node i is the only unmatched neighbor of j. So j will get matched to i.
- If $K < 1$ then j was matched to some node $k \in L \setminus i$ in the run on $G \setminus i$. By definition, $K = Y_k = \min_{i'\in N} Y_{i'}$. As $Y_i < K$ (and i is unmatched) it follows that $Y_i = \min_{i'\in N\cup i} Y_{i'}$. So j will get matched to i in the run on G.

In all cases, we see that i gets matched. ■

Lemma 3.3 *Consider the run of the algorithm on graph G with the fixed $Y_{i'}$ values for $i' \in L \setminus i$. We have $\beta_j \geq \frac{e-e^K}{e-1}$.*

Proof: Consider the runs of the algorithm on $G \setminus i$ and G in parallel. Note that $Y_{i'}$ is fixed for all $i' \in L \setminus i$; whereas $Y_i \in [0,1)$ can take any value. We first claim:

> At any point in the algorithm, the set of unmatched offline nodes in G is a superset of that in $G \setminus i$.

Suppose (for a contradiction) that j' is the first online arrival after which this statement is not true. Let $N' \subseteq L \setminus i$ denote the unmatched offline nodes (in the run on $G \setminus i$) just before j' arrives. Note that all nodes in N' are also unmatched in G just before j' arrives. After j' arrives, the unmatched nodes in G is *not* a superset of that in $G \setminus i$. For this to happen, we must have:

C1. in the run on G, node j' must be matched to some node $k' \in N'$.

C2. in the run on $G \setminus i$, node j' must *not* be matched to k'.

By C1 and the matching rule, we have $Y_{k'} \leq \min_{i' \in N' \cap N(j')} Y_{i'}$ as all nodes in N are unmatched. By C2, we have $Y_{k'} > \min_{i' \in N' \cap N(j')} Y_{i'}$, which is a contradiction.

Now consider the arrival of node j. Let $N \subseteq L \setminus i$ be the unmatched offline nodes in the run on $G \setminus i$. We now consider two cases depending on the value of K:

- If $K = 1$ then the lemma is trivially true as $\beta_j \geq 0$.
- If $K < 1$ then j was matched to some node $k \in N$ in the run on $G \setminus i$. By definition, $K = Y_k = \min_{i' \in N} Y_{i'}$. As all nodes in N are also unmatched in the run on G (by the above claim), j must get matched to some node $i' \in L$ with $Y_{i'} \leq Y_k = K$. Hence,
$$\beta_j = \frac{e - e^{Y_{i'}}}{e-1} \geq \frac{e-e^K}{e-1}.$$

■

Theorem 3.1 *The ranking algorithm is $1 - \frac{1}{e}$ competitive (in expectation) for bipartite matching.*

Proof: The key observation is that the dual solution ρ given by $\{\mathbb{E}[\alpha_i]\}_{i \in L}$ and $\{\mathbb{E}[\beta_j]\}_{j \in R}$ is feasible.

Consider the constraint for any edge (i,j), i.e., $\alpha_i + \beta_j \geq 1$. We condition on the $Y_{i'}$ values for all $i' \in L \setminus i$. Recall the definition of K from above. We will show that the constraint for (i,j) is satisfied in expectation, where the only randomness is in Y_i. Finally, taking an expectation also over $\{Y_{i'} : i' \in L \setminus i\}$ this would imply that ρ satisfies the constraint for (i,j). By Lemma 3.2, we have $\alpha_i = \frac{e^{Y_i}}{e-1}$ whenever $Y_i < K$. As $\alpha_i \geq 0$ always, taking expectation over Y_i,

$$\mathbb{E}[\alpha_i] \geq \int_{y=0}^{K} \frac{e^y}{e-1} dy = \frac{e^K - 1}{e-1}.$$

By Lemma 3.3, we have $\mathbb{E}[\beta_j] \geq \frac{e-e^K}{e-1}$. So we obtain $\mathbb{E}[\alpha_i] + \mathbb{E}[\beta_j] \geq 1$ as desired.

Finally, by Lemma 3.1, $\mathbb{E}[\mathsf{ALG}] \geq (1 - \frac{1}{e}) \cdot \mathbb{E}[\sum_{i \in L} \alpha_i + \sum_{j \in R} \beta_j]$. The theorem now follows by weak duality. ■

3.1 Randomized Lower Bound

Here, we show that no randomized online algorithm for matching can achieve a better competitive ratio that $1 - \frac{1}{e}$.

The input distribution We set $L = R = [n]$. The edges are defined randomly as follows. Let π be a uniformly random permutation on $[n]$. For each online node $j \in [n]$, its neighbors are

$$N(j) = \{(\pi(i), j) : j \leq i \leq n\}.$$

Note that the optimal value is n for every such instance. Consider any deterministic online algorithm on this input distribution. We will show that the expected size of the matching is at most $(1-\frac{1}{e})\cdot n$. This would imply the desired lower bound by Yao's lemma.

Bounding the algorithm's expected value For any position $i \in [n]$ and $j \in R$, let $x_{i,j} = \Pr[(\pi(i), j) \text{ matched}]$ where the probability is taken over the random permutation π. As there are no edges from $j \in R$ to $\pi(i)$ for $i < j$, we have $x_{i,j} = 0$ for all $1 \leq i < j \leq n$. We claim that:

$$x_{i,j} \leq \frac{1}{n-j+1}, \qquad \forall i \geq j, \forall j \in [n]. \tag{4}$$

To see this, fix any $j \in [n]$ and condition on $\pi(1), \cdots \pi(j-1)$. The algorithm needs to decide on the matching edge (if any) out of $j \in R$ immediately after it arrives. For each $\ell \in L$, let $M_{\ell,j}$ be the indicator random variable that is one if j gets matched to offline node ℓ. Note that $\sum_{\ell \in L} M_{\ell,j} \leq 1$. Moreover, for every $i \geq j$, $\pi(i)$ is uniformly distributed on $[n] \setminus \{\pi(1), \cdots \pi(j-1)\}$. So,

$$\Pr[(\pi(i), j) \text{ matched} \,|\, \pi(1), \cdots \pi(j-1)] \leq \frac{1}{n-j+1}, \quad \forall i \geq j.$$

Taking expectations over $\pi(1), \cdots \pi(j-1)$, we obtain (4).

Let $k \in [n]$ be some parameter that will be fixed later. The expected size of the algorithm's matching is

$$\mathbb{E}[\mathsf{ALG}] \;\leq\; \sum_{i=1}^{k} \sum_{j \in R} x_{i,j} \;+\; n - k.$$

Using (4) we can bound:

$$\sum_{i=1}^{k} \sum_{j \in R} x_{i,j} = \sum_{j \in R} \sum_{i=j}^{k} x_{i,j} \leq \sum_{j=1}^{k} \frac{k-j+1}{n-j+1} = k - \sum_{j=1}^{k} \frac{n-k}{n-j+1}.$$

So, we have

$$\mathbb{E}[\mathsf{ALG}] \;\leq\; n - (n-k) \cdot \sum_{j=1}^{k} \frac{1}{n-j+1}.$$

We now use

$$\sum_{j=1}^{k} \frac{1}{n-j+1} = \frac{1}{n-k+1} + \cdots \frac{1}{n} \geq \int_{n-k+1}^{n+1} \frac{1}{y} dy = \ln\left(\frac{n+1}{n+1-k}\right),$$

which implies

$$\mathbb{E}[\mathsf{ALG}] \;\leq\; n - (n-k)\cdot \ln\left(\frac{n+1}{n+1-k}\right). \tag{5}$$

Setting $k = n+1-\left\lfloor \frac{n+1}{e} \right\rfloor$, it follows that $\mathbb{E}[\mathsf{ALG}] \leq k \leq (1-\frac{1}{e})n + 2$. Thus, we obtain:

Theorem 3.2 *No randomized online algorithm for matching has competitive ratio better than* $1-\frac{1}{e}$.

4 Steiner Tree

Recall the Steiner tree problem. Given a metric (V,d) and terminals $R \subseteq V$, we want to find a tree containing R of minimum length. In the online setting, the terminals R arrive incrementally over time. We need to maintain a solution $S \subseteq \binom{V}{2}$ of edges that connects the current set of terminals to each other. Importantly, are only allowed to add edges to the solution S. We will compare the cost of our online algorithm to the optimal offline cost (which knows the terminals R).

Consider the natural greedy algorithm: when a new terminal t arrives, connect it to the current solution S using edges of minimum cost. Note that the solution S is always a spanning tree on the current set of terminals: this follows from triangle inequality.

Lower bound. We use the following "ball packing" lower bound on the optimal cost. For any subset $K \subseteq R$ of terminals, let $d_{min}(K)$ be the minimum pairwise distance between nodes of K. Then $\mathsf{OPT} \geq \frac{1}{2}|K| \cdot d_{min}(K)$. This can be verified directly. In fact, this is a consequence of LP duality as well. Consider the cut-covering LP relaxation and its dual:

$$\begin{aligned} \min \sum_{(uv)\in E} d_{uv}\cdot x_{uv} & \qquad\qquad & \max \sum_{W\in\mathcal{C}} y_W \\ \sum_{u\in W, v\notin W} x_{uv} \geq 1, \quad \forall W\in\mathcal{C}, & & \sum_{W: u\in W, v\notin W} y_W \leq d_{uv}, \quad \forall u,v\in V \\ x \geq 0. & & y \geq 0. \end{aligned}$$

Above, $\mathcal{C} = \{W \subseteq V : 1 \leq |R\cap W| < |R|\}$ denotes all subsets that separate R. Given subset $K \subseteq R$ of terminals, let $r = \frac{1}{2}d_{min}(K)$. Consider growing balls around each $t \in K$ from radius 0 till r, and raising y_W variables accordingly (for $W \cap R = \{t\}$). It is easy to check that the packing constraints in the dual LP are satisfied (by definition of r). Moreover, the dual objective is $r \cdot |K|$, which is a lower bound on OPT.

For the analysis, we will maintain several dual solutions. For each integer j, we will maintain a dual solution corresponding to terminals $N_j \subseteq R$ where $d_{min}(N_j) \geq 2^j$. Initially, all $N_j = \emptyset$. When terminal $t \in R$ arrives, we say that is belongs to *scale* g if the increase in the solution cost is in the interval $[2^g, 2^{g+1})$. Moreover, we add t to N_g, i.e. $N_g \leftarrow N_g \cup \{t\}$. Let q denote the number of non-empty scales at the end of the algorithm. Note that $q \leq \log_2(d_{max}/d_{min})$ where d_{max} (resp. d_{min}) denotes the maximum (resp. minimum) distance in the metric.

Lemma 4.1 *The online cost is at most* $2\sum_j 2^j \cdot |N_j|$.

Proof: By definition of the scale $s(t)$ of each terminal, we get $\mathsf{ALG} \leq \sum_{t\in R} 2^{s(t)+1} = \sum_j 2^{j+1}|N_j|$, as desired. ∎

Lemma 4.2 *For any scale j, there is a dual LP solution of objective $2^{j-1}|N_j|$.*

Proof: Fix any scale j. We will show that $d_{min}(N_j) \geq 2^j$. This would imply the lemma using the lower bound applied to $K = N_j$.

We now prove $d_{min}(N_j) \geq 2^j$ by induction. This is clearly true when $|N_j| \leq 1$. Consider a new terminal t that gets assigned to N_j and let $N'_j = N_j \cup \{t\}$. By induction we have $d_{min}(N_j) \geq 2^j$. We will show that $d_{min}(N'_j) \geq 2^j$, which would complete the proof. To see this, note that $\min_{u \in N_j} d_{t,u} \geq 2^j$ as the minimum cost to connect t to the current solution (that contains all of N_j) is at least 2^j. So $d_{min}(N'_j) = \min\{d_{min}(N_j), \min_{u \in N_j} d_{t,u}\} \geq 2^j$. ■

Using these two lemmas,

$$\mathsf{ALG} \leq 4 \sum_j 2^{j-1}|N_j| \leq 4q \cdot \mathsf{OPT},$$

where $q = O(\log(d_{max}/d_{min}))$ is the number of non-empty scales. We can further improve this bound to obtain:

Theorem 4.1 *The greedy algorithm is $O(\log k)$-competitive for Steiner forest.*

Proof: Let $T = \frac{1}{2}\sum_j 2^j|N_j|$ be the total objective from all dual solutions. Let m denote the largest scale j with $|N_j| \geq 2$, so the dual contribution from all higher scales is zero. Note that $T \geq 2^{m-1}$. We will ignore all dual solutions from scales $j < m - 2\log_2(k)$. Note that $\max |N_j| \leq k$, which means the dual value in T due to any scale j is at most $k2^{j-1}$. So the total dual value in scales $j < m - 2\log k$ is at most

$$k \cdot \sum_{j < m - 2\log k} 2^{j-1} = k \cdot 2^{m - 2\log k} \leq \frac{1}{2k} \cdot 2^{m-1} \leq \frac{T}{2k}.$$

So the total dual value from scales $m - 2\log k \leq j \leq m$ is $T' \geq (1 - \frac{1}{2k})T$. By Lemma 4.2 and weak duality, we get $T' \leq (2\log k) \cdot \mathsf{OPT}$. Hence, $T \leq ((2 + o(1))\log k) \cdot \mathsf{OPT}$. Finally, using Lemma 4.1, we get $\mathsf{ALG} \leq 4T \leq ((8 + o(1))\log k) \cdot \mathsf{OPT}$. ■

It is also known that no (deterministic) online algorithm for Steiner tree can achieve a ratio better than $\approx \frac{1}{2}\log_2 |R|$; see [2]. So this algorithm is essentially the best possible.

5 Steiner Forest

We now consider the more general Steiner forest problem. There is a metric (V, d) and k terminal-pairs $\{s_i, t_i\}_{i=1}^k$ arrive online. We need to maintain a solution F (subset of edges) that connects each terminal-pair, i.e., each s_i must be connected to t_i (using edges of F) immediately upon its arrival. We are only allowed to add edges to F. The objective is to minimize the total cost $\sum_{e \in F} d_e$. Let $R = \{s_i, t_i : i \in [k]\}$ denote all the terminals.

We will analyze the natural greedy algorithm that maintains solution F, initially empty. When pair (s, t) arrives, consider the shortest $s - t$ path P after contracting all edges in F (the current solution), and update $F \leftarrow F \cup P$.

We will show that this is an $O(\log^2 k)$ competitive algorithm.

Lower bound. We again use a "ball packing" lower bound, as for Steiner tree. For any subset $K \subseteq R$ of terminal-pairs[2], let $d_{min}(K)$ be the minimum pairwise distance between nodes of K. Then $\mathsf{OPT} \geq \frac{1}{2}|K| \cdot d_{min}(K)$. This can be verified directly. In fact, this is a consequence of LP duality as well. Consider the cut-covering LP relaxation and its dual:

$$\min \sum_{(uv)\in E} d_{uv} \cdot x_{uv} \qquad\qquad \max \sum_{W\in\mathcal{C}'} y_W$$

$$\sum_{u\in W, v\notin W} x_{uv} \geq 1, \quad \forall W \in \mathcal{C}' \qquad\qquad \sum_{W: u\in W, v\notin W} y_W \leq d_{uv}, \quad \forall u,v \in V$$

$$x \geq 0. \qquad\qquad y \geq 0.$$

Above, $\mathcal{C}' = \{W \subseteq V : \exists i \text{ with } |\{s_i, t_i\} \cap W| = 1\}$ denotes all subsets that separate *some* terminal-pair. Given subset $K \subseteq R$ of terminals, let $r = \frac{1}{2} d_{min}(K)$. Consider growing balls around each $t \in K$ from radius 0 till r, and raising y_W variables accordingly (for $W \cap R = \{t\}$). Note that each of these balls is an "active" subset that separates some pair. It is easy to check that the packing constraints in the dual LP are satisfied (by definition of r). Moreover, the dual objective is $r \cdot |K|$, which is a lower bound on OPT.

For the analysis, we will maintain several dual solutions in a careful manner. Let $\delta \approx \frac{1}{\log(2k)}$ be such that $\frac{1}{\delta}$ is an even integer. For each integer j, we will maintain a graph (N_j, E_j) with nodes $N_j \subseteq R$ and edges E_j where $d_{min}(N_j) \geq \delta \cdot 2^{j-1}$. Initially, all these graphs are empty.

When pair (s,t) arrives, recall that P is the shortest $s-t$ path in the metric d_F where edges in F are contracted. Let $g \in \mathbb{Z}$ be such that $2^g \leq d(P) < 2^{g+1}$. For each $j \leq g$ do:

1. If $d(s, N_j) \geq \delta \cdot 2^{j-1}$ then $N_j \leftarrow N_j \cup \{s\}$ and $\pi_j(s) \leftarrow s$.
2. If $d(s, N_j) < \delta \cdot 2^{j-1}$ then set $\pi_j(s) \in N_j$ to be the node closest to s.
3. Repeat steps 1-2 with s replaced by t.
4. Update $E_j \leftarrow E_j \cup \{(\pi_j(s), \pi_j(t))\}$.

Lemma 5.1 *For any scale j and edge $(u,v) \in E_j$, the distance $d_F(u,v) < \delta \cdot 2^j$.*

Proof: Consider the point in the algorithm when edge (u,v) was added to E_j. Let (s,t) be the terminal-pair that caused it. Then, we must have $u = \pi(s)$ and $v = \pi(t)$; we drop the subscript j as it is fixed. Note that

$$d_F(\pi(s), \pi(t)) \leq d_F(\pi(s), s) + d_F(s,t) + d_F(\pi(s), s) < \delta 2^{j-1} + 0 + \delta 2^{j-1} = \delta 2^j.$$

The second inequality uses that s is connected to t in F and the definition of $\pi(s)$. ■

Lemma 5.2 *Consider any scale j. The girth of graph (N_j, E_j) is more than $\frac{1}{\delta}$. Hence, the number of edges $|E_j| \leq 5 \cdot |N_j|$.*

Proof: Again, we drop subscript j. Suppose there is some cycle of length ℓ. Let (u,v) be the last added edge in this cycle and (s,t) the terminal-pair that added it. Note that $u = \pi(s)$ and $v = \pi(t)$. Consider the situation just before the arrival of pair (s,t) and let F denote the current solution. The graph (N, E) already has a $u-v$ path of length at most $\ell - 1$. By Lemma 5.1 and triangle inequality,

[2]We assume that the terminals in K are always paired, i.e., $s_i \in K$ iff $t_i \in K$.

it follows that $d_F(u,v) < \delta 2^j(\ell-1)$. Moreover, we know that $d(s,\pi(s)), d(t,\pi(t)) < \delta 2^{j-1}$. So, $d_F(s,t) < \delta 2^j \ell$. Combined with the fact that $d_F(s,t) \geq 2^g \geq 2^j$, we get $\ell \geq 1 + \frac{1}{\delta}$.

So the girth of graph (N,E) is at least $L = 1 + \frac{1}{\delta}$. We now use a well-known fact:

Theorem 5.1 ([1]) *The number of nodes in any graph of girth at least L (an odd number) and average degree $\rho \geq 2$ is more than $(\rho-1)^{\frac{L-1}{2}}$.*

Suppose (for a contradiction) that the average degree in graph (N,E) is $\rho \geq 5$. Then, we get $|N| > 4^{\frac{1}{2\delta}} = 2^{1/\delta} = 2k$. This contradicts with the fact that there are at most $2k$ terminals. Hence, $|E| = \rho \cdot |N| \leq 5|N|$, which proves the lemma. ■

Lemma 5.3 *The online cost is at most $\sum_j 2^j \cdot |E_j| \leq 5 \cdot \sum_j 2^j |N_j|$.*

Proof: We will prove this inequality step by step. Consider the arrival of pair (s,t). The increase in online cost is at most 2^{g+1}. Consider the quantity $G = \sum_{j \leq g} 2^j \Delta E_j$ where ΔE_j denotes the increase in the number of edges. For each $j \leq g$ it is clear that $\Delta E_j \geq 1$. So, $G \geq 2\sum_{j \leq g} 2^{j-1} = 2^{g+1}$. Adding over all arrivals, the online cost is at most $\sum_j 2^{j-1} \cdot 2|E_j|$. The last inequality now follows by Lemma 5.2. ■

Lemma 5.4 *For any scale j, there is a dual LP solution of objective $2^{j-2}|N_j| \cdot \delta$.*

Proof: This follows from the LP dual and the fact that $d_{min}(N_j) \geq \delta 2^{j-1}$. ■

Let $T = \frac{1}{4}\sum_j 2^j |N_j|$ be the total objective from all dual solutions. Let m denote the largest scale j with $N_j \neq \emptyset$, so $|N_m| \geq 2$. Note that $T \geq 2^{m-1}$. We will ignore all dual solutions from scales $j < m - 2\log_2(k)$. Note that $\max |N_j| \leq 2k$, which means the dual value in T due to any scale j is at most $k2^{j-1}$. So the total dual value in scales $j < m - 2\log k$ is at most $k \cdot 2^{m-2\log k} \leq \frac{1}{k} \cdot 2^{m-1} \leq T/k$. By averaging, there is some scale $m - 2\log k \leq j \leq m$ with dual objective $\geq \frac{1-1/k}{2\log k} \cdot T = \Omega(\frac{1}{\log k}) \cdot T$. Lemma 5.3 implies that the online cost is $O(1) \cdot T$, and by weak duality, we get:

Theorem 5.2 *The greedy algorithm is $O(\log^2 k)$-competitive for Steiner forest.*

5.1 A tight algorithm

Initially $F = \emptyset$. Upon arrival of terminal-pair s,t, we do the following:

1. Let denote the shortest $s-t$ path after contracting edges in F, and let $d(P)$ be its cost.
2. **Add** all edges in P to solution F.
3. Let $g \in \mathbb{Z}$ be such that $2^g \leq d(P) < 2^{g+1}$.
4. For each $j \leq g$ do:
   (a) If $d(s,N_j) \geq 2^{j-1}$ then $N_j \leftarrow N_j \cup \{s\}$ and $\pi_j(s) \leftarrow s$.
   (b) If $d(s,N_j) < 2^{j-1}$ then:
      - set $\pi_j(s) \in N_j$ to be the node closest to s.
      - **add** edge $(s, \pi_j(s))$ to solution F.
   (c) Repeat steps a-b with s replaced by t.

(d) Update $E_j \leftarrow E_j \cup \{(\pi_j(s), \pi_j(t))\}$.

This algorithm has the following guarantee:

Theorem 5.3 *There is an $O(\log k)$-competitive algorithm for Steiner forest.*

References

[1] Noga Alon, Shlomo Hoory, and Nathan Linial. The moore bound for irregular graphs. *Graphs Comb.*, 18(1):53–57, 2002.

[2] Makoto Imase and Bernard M. Waxman. Dynamic steiner tree problem. *SIAM J. Discret. Math.*, 4(3):369–384, 1991.