The maximal dense tree problem on a graph with dynamic edge weights

Tripurari Singh
Dept. of Computer Science
Johns Hopkins University
tsingh@cs.jhu.edu

December 1999

Abstract

In this paper we provide a polylogarithmic competitive online algorithm for the maximal dense tree problem on a graph with dynamic edge weights. The previous work on the topic, [AS97], only handled static edge weights.

We expect that the inclusion of dynamic edge weights will significantly improve the utility of our algorithm. Specifically we expect simpler and better solutions to the important throughput competitive online multicast problem.

1 Introduction

1.1 Statement of the Maximal Dense Tree problem and the results

The traditional *static* edge weight version of the maximal dense tree is formulated as follows. We are given a directed graph with non-negative edge weights and a specified root node. The *input* of the problem are *online requests* from nodes $v \in V$. The *output* of the algorithm is either *rejection*, or *acceptance*, and in case of acceptance, connection via an undirected path to a previously connected node, or to the root. The union of all connecting paths is a tree, rooted at the source.

An interesting generalization of the maximal dense tree problem allows edge weights to change over time. This unusual model is motivated by the interleaved multicast problem. In this dynamic edge-weight setting we consider two graphs, G and G_{max}. G and G_{max} are identical except that the edge weights of G are dynamic while those of G_{max} are static and are upperbounds of edge weights of G. The online algorithm works on the graph G and tries to compete with an optimal offline algorithm working on the graph G_{max}. The weight of the online tree is computed as the sum of the weights of its edges at the time they were taken by the online. The weight of the offline tree, and the revenues of the online and the offline trees are computed as before.

Four parameters of interest are the revenues and weights of the optimal offline and our online algorithms. Let $r^*, \tilde{r}$ be the revenues of the offline and the online algorithms respectively, and let $w^*, \tilde{w}$ be the weights of their trees. Based on these parameters we define the *density* of the offline and the online algorithms as follows:

Definition 1.1 We define the density d^* of the offline algorithm as:

$$d^* = \frac{r^*}{w^*} \tag{1}$$

For the online algorithm, we employ a (weaker) definition of density. We say that the online has an expected normalized density $\tilde{d}$ with error ψ if,

$$E(\tilde{r}) \geq \tilde{d} \cdot E(\tilde{w}) - \psi \tag{2}$$

where $E(\tilde{r})$, $E(\tilde{w})$ are its expected revenue and weight respectively.

Note that no probabilistic assumptions about the input are made, and the expectations are with respect to the random bits generated by the online algorithm. We measure the performance of the online algorithm *relative* to the offline algorithm by the following competitive ratios:

Definition 1.2 We say that the online algorithm is μ-maximal if

$$\frac{\tilde{r}}{r^*} = \mu \tag{3}$$

We say that the online algorithm is Δ dense with error ψ if

$$\frac{\tilde{d}}{d^*} = \Delta \tag{4}$$

where ψ is the error term associated with the online density $\tilde{d}$.

In what follows we use the term density to mean *expected density with error* when referring to the online, but retain the standard definition for the offline.

In the present framework, our goal is to *maximize* μ and Δ given the offline density d^*. Let n be the number of vertices, and let R be the total input revenue. Let $\epsilon > 0$ and $\gamma > 0$ be two constants. The main contribution of this paper is:

Fact 1.3 For every $\epsilon > 0, \gamma > 0$, the algorithm in Figure 4 guarantees the following,

$$\begin{aligned} \mu &\geq 1 - \epsilon \\ \Delta &\geq \frac{\epsilon}{\log(\frac{1}{2\gamma}) \cdot (1 + \log n)} \\ \psi &\leq \gamma \cdot R \end{aligned}$$

1.2 Relationship to other work

The maximal dense tree problem was first formulated in 1992 in an attempt to extend the admission control algorithms in [AAP93] to multicast connections, such as pay-per-view video on demand.

In the special case of a star graph (i.e. clients connected by their individual links to the video server) the online selective multicast problem is essentially the 'winner picking" problem introduced by Awerbuch, Azar, Fiat, and Leighton [AAFL96].

Probabilistic assumptions were used in in [AAG93] to handle the online problem on a general network. Subsequently, [AS97] developed the first polylogarithmically competitive online algorithm for the problem, obviating the need for probabilistic assumption of [AAG93].

The present paper is an extension of [AS97] to a graph with dynamic edge weights.

While this paper is *not* concerned with computational complexity and focuses on online decision making alone, the offline polynomial-time approximation problem is of independent interest. It is a variation of the prize-collecting salesman and the k-MST problems [Bal89, GH94, GLV87, RSM$^+$94]. The first polynomial-time approximation for these problems, due to Awerbuch, Azar, Blum and Vempala [AABV95], relied on an *offline* polynomial approximation of the maximal dense tree problem.

1.3 Structure of this paper

- In Section 2 we discuss the "resurrection model" and present the GREEDY RESURRECTION algorithm for graphs with *static* edge weights. This algorithm possesses some crucial properties, whose proof turns out to be non-trivial. This section essentially re-works the algebra of [AS97].
- In section 3 we extend the operation of GREEDY RESURRECTION to graphs with dynamic edge weights. We continue to work in the resurrection model.
- In section 4 we revert to the conventional online model: i.e. resurrections are disallowed. In this model, we present the Maximal Dense Tree algorithm that takes its decision by simulating a collection of of greedy resurrecting algorithms. Dynamic edge weights are supported.

2 The Resurrection Model and the GREEDY RESURRECTION Procedure

In this section we define the Resurrection Model by relaxing the Online model. The Resurrection Model allows us to initially reject requests, but later accept them and still earn their revenue. We refer to the revenue earned by connecting requests as soon as they are made as *live* revenue, and the revenue earned by resurrecting requests as *resurrected* revenue.

In the Resurrection Model, the Maximal d-Dense Tree Problem, on a graph with *static* edge weights, can be defined as follows: given a directed graph $G = (V, E, c)$ with $|V| = n$ and a weight function $c : E \to R^+$, and a special *source* node $s \in V$; a sequence of connection requests arrive online at various nodes, each of which can either be rejected or accepted. If a request is accepted, it must be connected to the source s or to a node previously connected to it. Furthermore the tree spanning the source and the set of connected nodes should maintain at least the specified density d, and be the maximal such tree. The crucial difference from the Online Dense Tree problem is that the density is defined as the ratio of the total revenue accepted - *live or resurrected* - to the weight of the spanning tree.

The obvious solution of the Maximal Dense tree in the Resurrection model is to wait until all requests have arrived and then run an Offline Maximal d-Dense tree algorithm. Note that we are only concerned with the online decisions of the problem and not with computational issues. In particular, the Offline Maximal d-Dense Tree problem is NP-hard, and has been approximated by Awerbuch et. al. in [AAG93].

We now turn our attention to the GREEDY RESURRECTION algorithm. While this algorithm is sub-optimal in the Resurrection model, a significant amount of its earned revenue is *live*. Unlike a Maximal d-Dense Tree Algorithm, however, it only maintains d density in the resurrection metric. Note that if this density criteria were absent, the optimal strategy would be to accept all requests. Briefly, the GREEDY RESURRECTION algorithm constructs its tree by incrementally adding subtrees as soon as they become dense enough. The GREEDY RESURRECTION procedure is formally presented in figure 1.

GREEDY RESURRECTION maintains a Steiner tree T spanning all connected requests. At each node, GREEDY RESURRECTION maintains two values - `DeadRevenue`and `ResurrectedRevenue`. Initially, the spanning tree T is set to be the source s, and both `DeadRevenue`and `ResurrectedRevenue`of each node are set to 0. When a request with revenue π arrives at a node v, a test is made to see if this revenue can be combined with the existing `DeadRevenue`s of v and nearby nodes to form a dense sub-tree. Such a dense sub-tree is referred to as a *fragment*. If a fragment is not formed as a result of the new request, its revenue π is added to the `DeadRevenue`of v. Otherwise this fragment is appended to the tree, its `DeadRevenue`is resurrected and the "unused" portion of π is output as *live* revenue.

It is important to note that a fragment contains exactly one live request, and that immediately after its resurrection the fragment's `ResurrectedRevenue`is reduced to zero (see Figure 2). Also note that only the *smallest* portion, π', of the revenue π is used to form a fragment. The d-dense sub-tree formed due

The d-Dense RESURRECTION Procedure

Initialization

$T \leftarrow s$

for all u

`DeadRevenue`$(u) \leftarrow 0$

`ResurrectedRevenue`$(u) \leftarrow 0$

Invoked on receiving π revenue at node v

Contract T into a single vertex s

Find smallest $\pi' \leq \pi$ such that $\exists$ a tree S with the following property

$v \in S, s \in S$ and $Cost(S) \leq (\sum_{u \in S, u \neq s} \texttt{DeadRevenue}(u) + \pi')/d$

If such π' found

for all $u \in S, u \neq s$

`ResurrectedRevenue`$(u) \leftarrow$ `DeadRevenue`(u)

`DeadRevenue`$(u) \leftarrow 0$

Uncontract T, $T \leftarrow T \cup S$

Output $\pi - \pi'$ as *live* revenue

else

Uncontract T

Figure 1: The d-Dense GREEDY RESURRECTION procedure.

to this smallest π' revenue may not be unique, in which case we define the fragment to be the maximal such d-dense sub-tree.

2.1 Analysis of the GREEDY RESURRECTION Procedure

In this section, we compare the *live* revenue earned by GREEDY RESURRECTION with that earned by the optimal Offline operating at a higher density.

Let $\tilde{T}, \tilde{w}, \tilde{r}$ be the tree constructed by the d-Dense GREEDY RESURRECTION , its weight and revenue respectively. Let T^* be *any* tree, and let w^*, r^* be its weight and revenue respectively. Let $\tilde{r}_l, \tilde{r}_d, \tilde{r}_r$ be the live, dead and resurrected revenues of the GREEDY RESURRECTION procedure. Let r_l^*, r_d^*, r_r^*be the components of $\tilde{r}_l, \tilde{r}_d, \tilde{r}_r$, respectively, that lie on the Offline tree T^*.

Since all revenue on T^* is either live, dead or resurrected by GREEDY RESURRECTION ,

$$r^* = r_l^* + r_d^* + r_r^* \tag{5}$$

In the remaining part of this section, we lower bound r_l^*, and hence $\tilde{r}_l$, by upperbounding r_d^* and r_r^*. In the following lemma we bound r_d^*.

Lemma 2.1 Let r_d^*be the dead revenue of d-Dense GREEDY RESURRECTION that intersects with the given tree T^*. Let w^* be the weight of T^*. Then $r_d^* < dw^*$.

Proof:

Assume to the contrary that $r_d^* \geq dw^*$.

Since r_d^*lies on T^* all its nodes can be spanned using w^* weight. This forms a d-dense fragment that should have been resurrected by GREEDY RESURRECTION - a contradiction since $r_d^*$is the *final* dead revenue. ■

Next, we turn our attention to r_r^*. Recall that r_r^*, along with $r_l^*$is used to "pay" for the tree $\tilde{T}$. In the following lemma, we compare $r_r^*$with the weight w^* of T^* times the density d of GREEDY RESURRECTION .

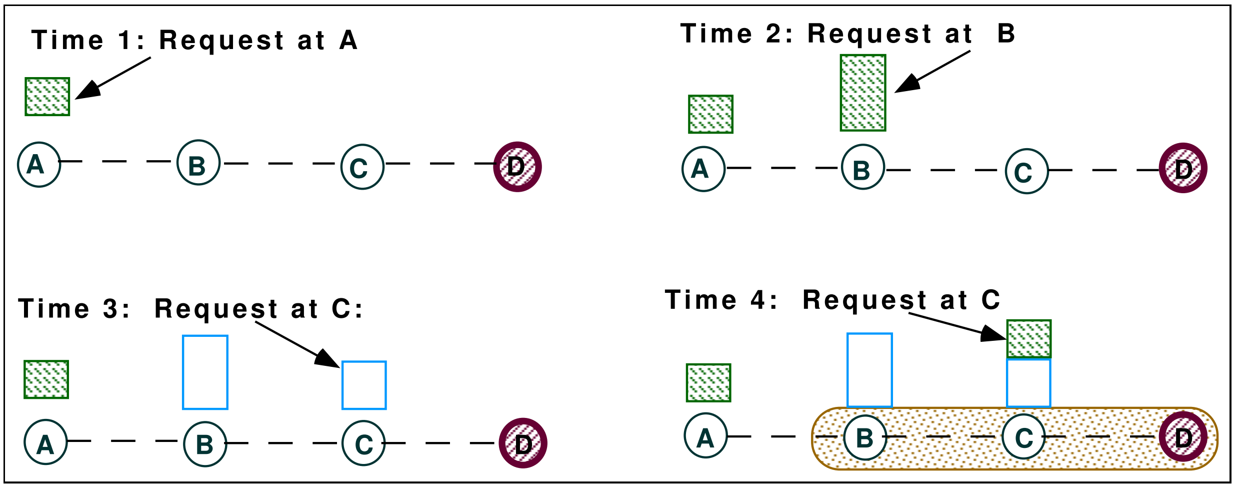

Figure 2: The execution of the algorithm on a weighted graph represented by a line consisting of node A, B, C and the source at node D. We illustrate 4 consecutive events in the course of the algorithm, which are arrivals of request at nodes A, B, C and then again at C at times 1,2,3 and 4, respectively. The request at time 3 at C creates dense subset $\{C\}$ and a dense subset $\{C, B\}$. The second is maximal dense subset and is resurrected by the algorithm, zeroing residual revenue at B, C. Residual revenue is shaded; resurrected revenue is unshaded.

r_r^*will, in general be larger than dw^* because Greedy Resurrection may use an "expensive" steiner tree to span the nodes in T^*, and also because some of the revenue on T^* may be used by Greedy Resurrection to connect nodes not in T^*. To see this recall that $d < d^*$, and hence Greedy Resurrection may connect more nodes than the optimal offline.

Lemma 2.2 Let T^* be *any* tree on the graph G, and let w^* be its weight. Also let r_r^*be the revenue resurrected by the d-Dense Greedy Resurrection from tree T^*. Then, $r_r^* \leq 2w^* d \log n$.

Proof:

Consider the nodes of T^*. Recall that a fragment is a maximal dense set created as a result of a request at a node (see Figure 3). Our main result will be to show the existence of a previously resurrected node "near" every resurrected node.

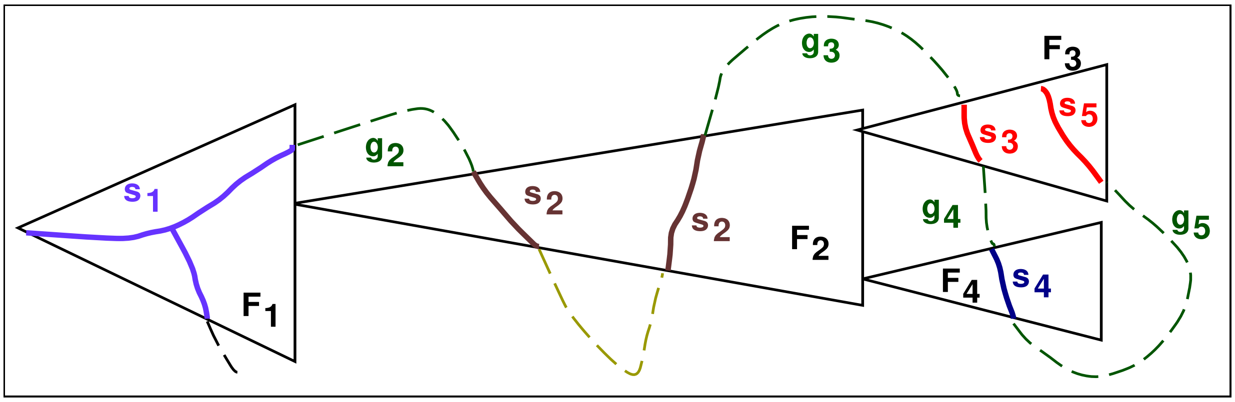

Figure 3: Proof of lemma lem:Steiner1. In the depth first search we encounter 4 segments s_1,s_2,s_3,s_4 (in that order) belonging to fragments F_1, F_2, F_3, F_4. The order of creation of these fragments is F_1, F_2, F_4, F_3. Thus s_1, s_4 are early segments and s_2, s_3, s_5 are late segments.

Conduct a depth first tour of T^*. Note that this tour traverses each edge twice and "opens out" T^* into a line graph. Define a *segment* as a contiguous section of the tour which does not pass through more than one fragment. Note that a segment can enter and leave a fragment any number of times, as long as it does not enter any other fragment in between.

Consider a linear ordering of a subset of segments. Classify a segment *early* if its adjacent segment(s) were resurrected after it; *late* otherwise. Note that if j is the number of early segments in a given sequence, the number of late segments is at least $j-1$.

Define *combing* as a procedure that takes as its input a sequence of segments and removes its late segments. Starting with the sequence of segments generated by our depth first search tour of T^*, iteratively comb it until only *one* segment - that containing the source - is left. Let h be the total number of combings needed. Let $\mathcal{A}_i$ be the segment sequence input to the i^{th} combing, and let $\mathcal{L}_i$ be the set of segments dropped by the i^{th} combing. Clearly, $h \leq 2\log n$ and $\cup_{i=1}^{h}\mathcal{L}_i$ is the initial set of segments.

We now show that the cost of resurrecting all nodes in $\mathcal{L}_i$ is bounded above by wd for $\forall i$. Consider a segment $s \in \mathcal{L}_i$. Note that s is a late segment in $\mathcal{A}_i$, and hence one of its adjacent segments t in $\mathcal{A}_i$ is resurrected before s. Clearly, at the time (the fragment containing) the segment s is resurrected, it was possible to connect the nodes in s to the closest node v in t along the path taken by the depth first search tour. This connection could be made by a line graph spanning the nodes in $s \cup v$. We refer to this line graph as a *connector*. Observe that all *connectors* are disjoint. Also, each *connector* is contained in the depth first search tour of T^*. Since our depth first search tour traverses each edge twice, and the threshold density of resurrection is d,

$$\texttt{ResurrectedRevenue}(\mathcal{L}_i) \leq 2wd, 1 \leq i \leq h$$

Substituting $2\log n$ for h, and correcting for the double traversal of each edge by the tour, we get,

$$r \leq 2w^* d \log n$$

■

We now lower bound r_l^*.

Theorem 2.3 Let r^*, w^* be the revenue and weight respectively of any d^*-Dense Tree T^*, and let r_l^* be the live revenue earned by the d-Dense GREEDY RESURRECTION procedure lying on T^*. Then $r_l^* = r^* - w^* d(1+\log n)$. Also, if $d = d^*\delta/(1+\log n)$, then $r_l^* = (1-\delta)r^*$.

Proof:

From equation 5 and lemmas 2.1 and 2.2 we have,

$$\begin{aligned} r_l^* &= r^* - r_d^* - r_r^* && (6) \\ &\geq r^* - (w^* d + w^* d \log n) && (7) \\ &= r^* - w^* d(1+\log n) && (8) \end{aligned}$$

Furthermore, if $d = \frac{d^*\delta}{1+\log n}$ we have,

$$r_l^* \geq r^* - w^* d^* \delta \quad (9)$$

But $w^* d^* \leq r^*$. Therefore,

$$r_l^* \geq r^*(1-\delta) \quad (10)$$

■

3 GREEDY RESURRECTION on graphs with dynamic edge weights

In this section we analyze the performance of the GREEDY RESURRECTION procedure on a graph with dynamic edge weights. That is, we work with a graph $G = (V, E, c)$ whose weight function $c : (E \times t) \rightarrow$

R^+, is a function of time t. We neglect the processing time and thus assume that the edge weights of the graph G do not change while requests are being processed. We compare the performance of the d-Dense GREEDY RESURRECTION procedure with the offline that works on a "static" graph $G_{max} = (V, E, c_{max})$ whose edge weights $c_{max}(e) = max_t(c(t, e)), \forall e \in E$ are upperbounds on the edge weights of G.

Let T^*, w^*, r^* be a given tree on G_{max}, its weight and revenue respectively, and let $\tilde{T}, \tilde{w}, \tilde{r}$ be the tree constructed by the d-Dense GREEDY RESURRECTION , its weight and revenue respectively. Note that when calculating the weight of a tree, we use the weight of every edge *at the time that edge is added.* This is significant when working with the graph G. Also note that some sub-trees that were not viable fragments when their requests arrived can become viable once their edge weights decrease. However, since GREEDY RESURRECTION is invoked only when new requests arrive, such viable sub-trees may not be examined, and consequently may not become fragments.

As before, let $\tilde{r}_l, \tilde{r}_d, \tilde{r}_r$ be the live, dead and resurrected revenues of the GREEDY RESURRECTION procedure. Let r_l^*, r_d^*, r_r^*be the components of $\tilde{r}_l, \tilde{r}_d, \tilde{r}_r$, respectively, that lie on the Offline tree T^*. We first bound $r_d^*$from below.

Lemma 3.1 Let r_d^* be the dead revenue of d-Dense GREEDY RESURRECTION that intersects with *any* tree T^*. Let w^* be the weight of T in G_{max}. Then, $r_d^* < dw^*$.

Proof:

Assume to the contrary that $r_d^* \geq dw^*$.

Since r_d^* lies on T^*, all its nodes can be spanned using w^* weight G_{max}. At the end of the GREEDY RESURRECTION procedure, this leaves us with at least one d-dense fragment that can be resurrected by GREEDY RESURRECTION - a contradiction since r_d^* is the *final* dead revenue.

■

We now turn our attention to r_r^*.

Lemma 3.2 Let T^* be *any* tree on the graph G_{max}, and let w^* be its weight. Also let r_r^* be the revenue resurrected by the d-Dense GREEDY RESURRECTION from tree T^*. Then, $r_r^* \leq 2w^* d \log n$.

Proof:

As in lemma 2.2 define segments using a depth first tour of T^* and define the sets $\mathcal{L}_i, \mathcal{A}_i, 1 \leq i \leq h, h \leq 2 \log n$ using *combing*.

We now show that the cost of resurrecting in G, the set of nodes in $\mathcal{L}_i$ is bounded above by $w^* d$ for $\forall i$, where w^* is measured in G_{max}. Consider a segment $s \in \mathcal{L}_i$. Note that s is a late segment in $\mathcal{A}_i$, and hence one of its adjacent segments t in $\mathcal{A}_i$ is resurrected before s. Clearly, at the time (the fragment containing) the segment s is resurrected, it was possible to connect the nodes in s to the closest node v in t along the path taken by the depth first search tour. This connection could be made by a line graph spanning the nodes in $s \cup v$. We refer to this line graph as a *connector*. Observe that all *connectors* are disjoint. Also, each *connector* is contained in the depth first search tour of T^*. Since our depth first search tour traverses each edge twice, the cost of each edge in G is upperbounded by their cost in $G_{max}$and the threshold density of resurrection is d,

$$\texttt{ResurrectedRevenue}(\mathcal{L}_i) \leq 2w^* d, 1 \leq i \leq h$$

Substituting $2 \log n$ for h, and correcting for the double traversal of each edge by the tour, we get,

$$r \leq 2w^* d \log n$$

■

We now lower bound r_l^*.

Theorem 3.3 Let r^*, w^* be the revenue and weight respectively of any d^*-Dense Tree T^* in G_{max}, and let r_l^* be the live revenue earned by the d-Dense Greedy Resurrection procedure lying on T^*. Then $r_l^* = r^* - w^* d(1+\log n)$. Also, if $d = d^*\delta/(1+\log n)$, then $r_l^* = (1-\delta)r^*$.

Proof:

From equation 5 and lemmas 3.1 and 3.2 we have,

$$\begin{aligned} r_l^* &= r^* - r_d^* - r_r^* && (11)\\ &\geq r^* - (w^* d + w^* d \log n) && (12)\\ &= r^* - w^* d(1+\log n) && (13) \end{aligned}$$

Since $d = \frac{d^*\delta}{1+\log n}$ we have,

$$r_l^* \geq r^* - w^* d^* \delta \tag{14}$$

Furthermore, if $w^* d^* \leq r^*$,

$$r_l^* \geq r^*(1-\delta) \tag{15}$$

■

4 The Maximal Dense Tree Algorithm

In this section we solve the Maximal Dense tree problem in the true (i.e. non-resurrection) online model. Formally, we are given a graph $G = (V, E, c)$ whose weight function $c : (E \times t) \rightarrow R^+$ is a function of time t, a source vertex $s \in V$ and a set of online requests for connection. A request can either be accepted or rejected. If it is accepted, it should be connected to the source s, or to a node previously connected to s. Also given is a graph $G_{max} = (V, E, c_{max})$ which is identical to G except that $c_{max}(e) = max_t(c(t,e)), \forall e \in E$.

We track the working of the online algorithm on the graph G (with time-varying edge costs), and compare it with an offline working on "static" graph G_{max}. As before, we consider the weight of an edge *at the time it is taken* when calculating the weight of a tree.

In this section we develop an online algorithm that maintains density d with error $\psi = \gamma R$, where R is the total input revenue, and earns revenue that is within a constant factor of that earned by the offline Maximal d^*-Dense Tree algorithm on graph G_{max}, where d^* is a polylogarithmic factor larger than d.

We obtain our online algorithm by "cascading" a set of Greedy Resurrection procedures, picking a "winning" Greedy Resurrection procedure using the techniques of [AAFL96] and taking each edge that this Greedy Resurrection procedure takes. We formally present our algorithm in fig 4.

4.1 Analysis

Let T^*, w^*, r^* be any offline d^*-Dense tree on G_{max}, its weight and revenue respectively, and let $\tilde{T}, \tilde{w}, \tilde{r}$ be the tree constructed by the Maximal d-Dense tree algorithm in fig 4, its weight on G and revenue respectively. Also, let $T_i, w_i, r_i, 1 \leq i \leq L$ be the tree constructed by the Greedy Resurrection$_i$, its weight (on G) and *live* revenue respectively.

Observe that the live requests output by Greedy Resurrection$_\eta$ are accepted by our online algorithm and hence the live revenue r_η is earned. In the next lemma we show that the total revenue accepted by our online algorithm $\tilde{r} = r_\eta$ is "comparable" to r^*.

Initialize
Initialize the Maximal Dense Tree $T \leftarrow s$
Initialize $L = \log(\frac{1}{2\gamma})$ copies of GREEDY RESURRECTION(on dynamic edge weights), GREEDY RESURRECTION$_i, 1 \leq i \leq L$ with density $= 2d$
Choose $\eta \in [1 \dots L]$ such that $p_j = \text{Prob}(\eta = j) = p_1 \cdot 2^{j-1}$, where $p_1 = 2\gamma$

Invoke GREEDY RESURRECTION$_1$ with the input request of profit π at node v and GREEDY RESURRECTION$_i, 2 \leq i \leq \eta$ with input node v and profit $=$ live revenue output by GREEDY RESURRECTION $_{i-1}$
Accept all *live* requests accepted by GREEDY RESURRECTION $_\eta$ and connect them using the tree constructed by GREEDY RESURRECTION $_\eta$

Figure 4: **The Online Maximal Dense Tree algorithm.** This algorithm takes as its input two parameters, d and γ, and a sequence of connection requests and outputs a maximal dense tree of expected density d with error γR, where R is the revenue spent on the first greedy resurrection procedure.

Lemma 4.1 The Maximal d-Dense with error γR Tree algorithm of fig 4 earns $\tilde{r} = (1-\epsilon)r^*$ units of revenue, where r^* is the revenue earned by any offline algorithm that maintains density $d^* = \frac{d}{\epsilon} \cdot log(\frac{1}{2\gamma})(1 + \log n)$ on the graph G_{max}, and R is the total input revenue.

Proof: Since the live revenue of GREEDYRESURRECTION$_i$ forms the input of GREEDY RESURRECTION$_{i+1}$, $\forall i < \eta$,

$$\tilde{r} = r_\eta \leq r_i, \forall i < \eta$$

Next, we bound r_η from below. From theorem 3.3 we have,

$$r_{i+1} \geq r_i - w^* d(1 + \log n), \forall i < \eta$$

where w^* is the weight of the offline tree. The above recurrence implies:

$$r_\eta \geq r^* - L \cdot w^* d(1 + \log n)$$

Substituting $L = \log(\frac{1}{2\gamma})$ and $d = \frac{\epsilon d^*}{log(\frac{1}{2\gamma})(1+\log n)}$ we have,

$$\begin{aligned} r_\eta &\geq r^* - log(\frac{1}{2\gamma}) \cdot w^* \cdot \frac{\epsilon d^*}{log(\frac{1}{2\gamma})(1 + \log n)} \cdot (1 + \log n) & (16) \\ &= r^* - \epsilon w^* d^* & (17) \end{aligned}$$

But $r^* \geq w^* d^*$. Hence we have the result,

$$\tilde{r} = r_\eta \geq (1 - \epsilon) r^*$$

■

In the following lemma we prove the key density result: that the expected live revenue $E(\tilde{r})$ of the online algorithm of fig 4 is "comparable" to d times its expected weight $E(\tilde{w})$. Note that $\tilde{w}$ may be much greater than w^*, but in most such cases $\tilde{r}$ will be correspondingly greater than r^*.

Lemma 4.2 The Maximal Dense Tree algorithm in fig 4 maintains an expected density of no less than d with error no greater than γR, where R is the revenue used by the first resurrection procedure.

Proof: By definition, the expected weight $E(\tilde{w})$ of the tree constructed by the algorithm in fig 4 is

$$E(\tilde{w}) = p_1 \cdot w_1 + p_2 \cdot w_2 \dots + p_L \cdot w_L \tag{18}$$

Since only the *live* revenue of GREEDY RESURRECTION$_i$ is used to buy T_{i+1}, and GREEDY RESURRECTION$_i$ is a $2d$-Dense GREEDY RESURRECTION procedure, for all $i, 1 \leq i \leq \eta$

$$r_i \geq 2dw_{i+1}$$

A subtle point to note is that w_{i+1} is not defined when $i = \eta$, and hence the above relationship is not defined to $i = \eta$. At the expense of abusing notation, we "extend" the definition for $i = \eta$ by using w_{i+1} to refer to the weight of T_{i+1} in a *different* execution where $\eta \geq i+1$. This extension still does not define w_{L+1}, and consequently we will not use the above relationship for $i = L$.

Now, the expected revenue, $E(\tilde{r})$, of the tree constructed by the algorithm in fig 4 is

$$\begin{aligned} E(\tilde{r}) &\geq 2d \cdot (p_1 \cdot w_2 + p_2 \cdot w_3 \ldots + p_{L-2} \cdot w_{L-1} + p_{L-1} \cdot w_L) + r_L \\ &\geq 2d \cdot (p_1 \cdot w_2 + p_2 \cdot w_3 \ldots + p_{L-2} \cdot w_{L-1} + p_{L-1} \cdot w_L) \end{aligned}$$

But $p_i = 2p_{i+1}, \forall i \leq L-1$. Therefore,

$$E(\tilde{r}) \geq d \cdot (p_2 \cdot w_2 + p_3 \cdot w_3 \ldots + p_{L-1} \cdot w_{L-1} + p_L \cdot w_L) \tag{19}$$

From equations 18 and 19 we have,

$$E(\tilde{r}) \geq d \cdot E(\tilde{w}) - p_1 \cdot d \cdot w_1$$

But $d \cdot w_1 \leq \frac{r_1}{2} \leq \frac{R}{2}$ where R is the revenue used by the first resurrection procedure. Since $p_1 = 2\gamma$ we have,

$$E(\tilde{r}) \geq d \cdot E(\tilde{w}) - \gamma R$$

Hence the expected revenue of the Maximal d-Dense tree algorithm of fig 4 is d with error γR ■

Corollary 4.3 Given that the optimal offline Maximal d^*-Dense algorithm earns r^* units of revenue, we have an online algorithm that earns at least $r^*(1-\epsilon)$ units of revenue and maintains expected density $= \frac{\epsilon d^*}{log(\frac{1}{2\gamma})(1+\log n)}$ with error $\psi \leq \gamma R$, where R is the total input revenue.

References

[AABV95] Baruch Awerbuch, Yossi Azar, Avrim Blum, and Santosh Vempala. Improved approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. In *Proceedings of the ACM STOC*, May 1995.

[AAFL96] Baruch Awerbuch, Yossi Azar, Amos Fiat, and Tom Leighton. Making commitments in the face of uncertainty: How to pick a winner almost every time. In *Proceedings of the ACM STOC*, 1996.

[AAG93] Baruch Awerbuch, Yossi Azar, and Rainer Gawlick. Dense trees and competitive selective multicast. unpublished manuscript, December 1993.

[AAP93] Baruch Awerbuch, Yossi Azar, and Serge Plotkin. Throughput competitive on-line routing. In 34^{th} *Annual Symposium on Foundations of Computer Science,* Palo Alto, California, pages 32–40. IEEE, November 1993.

[AS97] Baruch Awerbuch and Tripurari Singh. Online algorithms for selective multicast and maximal dense trees. In *Proceedings of the* 29^{th} *Annual ACM Symposium on Theory of Computing, El Paso, Texas*, May 1997.

[Bal89] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19:621–636, 1989.

[GH94] N. Garg and D. Hochbaum. An $o(\log k)$ approximation algorithm for the k minimum spanning tree problem in the plane. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 432–438, 1994.

[GLV87] B.L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

[RSM+94] R. Ravi, R. Sundaram, M.V. Marathe, D.J. Rosenkrantz, and S.S. Ravi. Spanning trees short and small. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1994.