

An optimal rebuilding strategy for a decremental tree problem

Nicolas Thibault, Christian Laforest

Tour Evry 2, LaMI/IBISC, Université d'Evry, 523 place des terrasses, 91000 EVRY
France {nthibaul, laforest}@lami.univ-evry.fr

Abstract. This paper is devoted to the following *decremental* problem. Initially, a graph and a distinguished subset of vertices, called *initial group*, are given. This group is connected by an initial tree. The decremental part of the input is given by an *on-line* sequence of withdrawals of vertices of the initial group, removed on-line one after one. The goal is to keep connected each successive group by a tree, satisfying a *quality* constraint: The maximum distance (called *diameter*) in each constructed tree must be kept in a given range compared to the best possible one. Under this quality constraint, our objective is to minimize the *number of critical stages* of the sequence of constructed trees. We call "critical" a stage where the current tree is rebuilt. We propose a strategy leading to at most $O(\log i)$ critical stages (i is the number of removed members). We also prove that there exist situations where $\Omega(\log i)$ critical stages are necessary to *any algorithm* to maintain the quality constraint. Our strategy is then worst case optimal in order of magnitude.

A lot of works have been devoted to the construction of trees spanning a given set of vertices in a graph. For example the *Steiner tree* problem, where the goal is to span a set (called *group*) of distinguished vertices (called *members*) with a minimum weight tree, has been extensively studied. As the problem is NP-complete, numerous *approximation algorithms* have been designed (see [1, 3] for example). In [8], Waxman was the first author to present the on-line version of this problem in which vertices to add in, or to remove from, the current group *revealed* one by one (see [2] references on on-line problems). In this first paper, he divides the problem into two categories: A model in which "heavy" changes of the current tree are not allowed and a model in which changes are allowed. Then, Imase and Waxman proposed in [4] two different strategies corresponding to the two models above. In the first one the tree is just incremented or decremented and the degradation of weight is evaluated, whereas in the second one they allow changes in the current tree to maintain a guaranty on the weight. At each stage, they prove that they construct with the first strategy a tree whose weight is at a logarithmic ratio compared to the optimal one (i.e. the weight of a Steiner tree of the current group), and that they construct with the second strategy a tree whose weight is at a constant ratio compared to the optimal one. They give for the second strategy an upper bound of $O(\sqrt{i})$ on the number of elementary changes per stage (where i is the number of new members). However, the tree can

potentially be changed at each stage; this means that each stage is potentially what we call later a *critical stage*.

In [6], a very similar on-line Steiner tree problem with a delay constraint from one node to the others is studied. But the authors only evaluate their method with simulations, and they give no upper bound for the different competitive ratios. Note that in [4, 6], only the number of elementary changes is taken into account to measure the level of damage due to the allowed changes in the current tree.

In this paper we are concerned with a decremental group problem where the members to remove are revealed on-line one by one. However, we do not focus on the same objective function (the weight of the tree) but on a different measure: The *diameter* of the current group induced by the current tree. Note that we consider here a model in which changes are allowed because it can easily be shown that *any* on-line algorithm without critical stage *cannot* guarantee a constant competitive ratio (for the diameter objective function we consider in this paper). That is why we fix here a “relative budget”, called *quality constraint*, on the diameter and we propose an algorithm minimizing the number of critical stages necessary to guarantee this budget constraint at each stage. Note also that we have proved that our algorithm leads to a constant number of elementary changes per stage in average (but we do not give in this paper the definitions and the proof associated to this problem because of the limitation on the number of pages).

A motivation for such model and objective function is the construction of connection structures for groups of members in networks. An important QoS parameter is *latency* that is expressed here in terms of maximum distance between users. This maximum distance must be guaranteed (our *quality constraint*). However, this must be done by minimizing the number of critical stages since they induce perturbations in communications in the current group (implying many re-routing operations between members in the current tree).

In Section 1 we describe more formally our problem. More precisely, in Subsection 1.1 we describe and motivate the constraints (namely the *tree* and *quality* constraints) that must be satisfied at each stage. In Subsection 1.2 we give the definitions of a *critical stage*. In Subsection 2.1 we propose an algorithm satisfying the construction constraints (in Section 2.2). In Subsection 2.3 we prove that our algorithm leads to at most $O(\log i)$ critical stages (where i is the number of removed members). We prove in Section 3 that our strategy is worst case optimal in order of magnitude for the number of critical stages criterion by constructing a scenario in which at least $\Omega(\log i)$ critical stages are necessary for *any* algorithm to satisfy the *quality* constraint. These results show that our algorithm is worst case optimal for the number of critical stages.

1 Definitions and notations

Let $G = (V, E, w)$ be any connected weighted graph representing a network. V is the set of vertices (modeling the nodes of the network), E the set of edges

(modeling the set of physical links) and w a positive weight function of the edges (modeling the length of the edges). We denote by $d_G(u, v)$ the *distance between u and v in G* , i.e. the sum of the weights of the edges of a minimum weight path between u and v in G .

Definition 1 (Diameter of a group M). Let $G = (V, E, w)$ be a graph and let $M \subseteq V$ be a group. We denote the diameter of M in G by

$$D_G(M) = \max\{d_G(u, v) : u, v \in M\}.$$

1.1 Construction constraints

In our problem, the graph $G = (V, E, w)$ and an *initial group* $M_0 \subseteq V$ are given (with $M_0 \neq \emptyset$). For example, in a meeting on network (called net-meeting) this initial group M_0 represents the set of members present at the beginning of the meeting. A structure, noted $T_0 = (V_0, E_0)$, must be created to connect the *members* of M_0 (T_0 spans M_0 in $G : M_0 \subseteq V_0 \subseteq V, E_0 \subseteq E$).

However, members may leave the meeting. These members must be removed from the current group (we underline that they are **not** removed from the underlying graph G). Let $m_0 = |M_0|$ be the size of the initial group. Let $u_1, u_2, \dots, u_i, \dots$ ($i \leq m_0 - 1$) be the sequence of members to remove. For every i , $1 \leq i \leq m_0 - 1$, we denote by $M_i = M_{i-1} \setminus \{u_i\}$ the i^{th} group, and by $m_i = |M_i|$ its size. Thus, starting from the initial connection structure T_0 for M_0 , at each *stage* of withdrawal i , the member u_i is removed by updating the current structure T_{i-1} (spanning M_{i-1}) to obtain T_i spanning M_i .

Note that as the members to remove are revealed one by one, we are in an *on-line* model. It means that we do not know the future: Neither in which order the members are removed, nor what is the set of members to remove. Hence, each stage can potentially be the last one; this explains why we are interested by giving guarantees *at each stage*.

We need the following definition that presents the best possible connection tree for the group M_k , minimizing the diameter parameter.

Definition 2 (Optimal tree). Let $G = (V, E, w)$ be a graph. For every i , $0 \leq i \leq m_0 - 1$, we denote by T_i^* a tree satisfying

$$D_{T_i^*}(M_i) = \min\{D_T(M_i) : T \text{ tree spanning } M_i\}.$$

We are now ready to give the two constraints that each current structure T_i must satisfy.

- **The tree constraint:** For every i , $0 \leq i \leq m_0 - 1$, T_i must be a *tree*, spanning M_i , in which all leaves are in M_i (we call that a *pruned tree*).
- **The quality constraint:** Let $c \geq 1$ be any fixed constant representing the *required level of quality*. For every i , $0 \leq i \leq m_0 - 1$, we must have $D_{T_i}(M_i) \leq c \cdot D_{T_i^*}(M_i)$.

As in a net-meeting the current structure T_i is used to support the communications between members of M_i , the *tree constraint* is set in order to simplify the mechanisms of routing and duplication of information in T_i . Indeed, there is only one route between any pair of members in a tree; moreover as there is no cycle, a simple flooding process can be used to broadcast information from any member. This flooding naturally ends at the leaves that are members (because trees are pruned); there is no need of costly process to control it.

The *quality constraint* of level c is set to guarantee that the induced diameter of the current group in T_i is not too large compared to the best possible diameter in T_i^* (at most c times the best possible diameter).

In the rest of the paper we say that an algorithm solves our problem if, for any on-line sequence of successive groups $M_0 \supset \dots \supset M_i$, it returns a sequence of trees T_0, \dots, T_i (T_i spanning M_i) satisfying the *tree* and *quality* constraints.

1.2 The criterion to minimize

In this subsection we present the cost associated with any algorithm satisfying the tree and quality constraints. We first need the following definitions.

Definition 3 (Critical stage). *Let \mathcal{A} be an algorithm solving our problem. At stage i , $1 \leq i \leq m_0 - 1$, Algorithm \mathcal{A} builds $T_i = (V_i, E_i)$ from $T_{i-1} = (V_{i-1}, E_{i-1})$. Stage i is a critical stage if $E_i \not\subseteq E_{i-1}$.*

We distinguish critical stages from other stages since they generate a lot of perturbations. Indeed, if i is a critical stage, the communication routes in T_{i-1} between members already in the current group M_{i-1} have to be changed in T_i . Potentially all the routing tables of the connecting nodes must be modified. This generates a heavy traffic to update them. Moreover the current communications between members of M_{i-1} initiated before the changes may be interrupted. That is why the number of critical stages must be minimized.

On the other hand, the withdrawal of a member by just removing useless branches in the tree generates only local changes and is not considered as a critical stage (since in this case $E_i \subseteq E_{i-1}$). The update of the routing can just be done by broadcasting the information of the departure of the leaving member in the new tree T_i . This does not create any re-routing between the other members. The aim of this paper is to minimize the total number $\#CS(T_0, \dots, T_i)$ of critical stages while respecting the *tree* and the *quality* constraints.

2 Our algorithm CS

2.1 Definition of Algorithm CS (Critical Stages)

To define Algorithm CS, we need the following algorithm, called MD for Minimum Distance. We denote by $MD(M)$ Algorithm MD applied to group M of size m to find a particular group $M(r^*)$ of size $\lfloor \frac{m}{2} \rfloor + 1$ and what we call its associated *root*.

Algorithm MD(M)

1. For each $r \in M$, sort the m vertices of M by non decreasing value of their distance to r :
 $r, u_1^r, \dots, u_{m-1}^r$ ($d_G(r, u_1^r) \leq d_G(r, u_2^r) \leq \dots \leq d_G(r, u_{m-1}^r)$).
 Let $M(r) = \left\{ r, u_1^r, \dots, u_{\lfloor \frac{m}{2} \rfloor}^r \right\}$.
2. Return r^* and its associated group $M(r^*)$ such that

$$d_G\left(r^*, u_{\lfloor \frac{m}{2} \rfloor}^{r^*}\right) = \min \left\{ d_G\left(r, u_{\lfloor \frac{m}{2} \rfloor}^r\right) : r \in M \right\}$$

Note that for all $r \in M$, the vertices u_1^r, \dots, u_{m-1}^r can be sorted by non decreasing value of $d_G(r, u_k^r)$ and the associated group $M(r)$ can be constructed in polynomial time by using Dijkstra's algorithm. Thus, Algorithm MD(M) finds $M(r^*)$ and its associated root r^* in polynomial time.

The main idea of Algorithm CS is to define particular stages numbers, called *rebuilding stages* during which we (totally) reconstruct the current tree (to match the quality constraint). Between two successive *rebuilding stages*, a member is leaving by just removing the dead branches of the current tree (in order to maintain at each stage a pruned tree to satisfy the *tree* constraint).

The following sequence (a_k) defines the rebuilding stages of our algorithm: $m_{a_0} = m_0$ is the size of the initial group M_0 and for every a_k ($k \geq 1$), $m_{a_k} = \lfloor \frac{m_{a_{k-1}}}{2} \rfloor$ is the size of the group M_{a_k} .

Algorithm CS

- Initially, at stage $a_0 = 0$:
 CS builds a shortest path tree spanning the first group M_0 , rooted in $r_0 \in M_0$, where $r_0 \in M_0$ is the root found by MD(M_0).
- After the last rebuilding stage a_k :
 Let M_{a_k+j} be the current group and let u_{a_k+j} be the j^{th} member revealed to be removed since the last rebuilding stage a_k .
 - If $m_{a_k+j} > \lfloor \frac{m_{a_k}}{2} \rfloor$ (corresponding to $j < m_{a_k} - \lfloor \frac{m_{a_k}}{2} \rfloor$):
 Update the tree $T_{a_k+j-1} = (V_{a_k+j-1}, E_{a_k+j-1})$ by pruning potential useless branches.
 We obtain the pruned tree $T_{a_k+j} = (V_{a_k+j}, E_{a_k+j})$ spanning M_{a_k+j} satisfying $E_{a_k+j} \subseteq E_{a_k+j-1}$.
 - Otherwise, we have $m_{a_k+j} = \lfloor \frac{m_{a_k}}{2} \rfloor$
 (corresponding to $j = m_{a_k} - \lfloor \frac{m_{a_k}}{2} \rfloor$):
 This is a rebuilding stage and we have $m_{a_k+j} = \lfloor \frac{m_{a_k}}{2} \rfloor = m_{a_{k+1}}$.
 Break the current tree and construct $T_{a_{k+1}}$, a shortest path tree spanning $M_{a_{k+1}}$, rooted in $r_{a_{k+1}}$ (where $r_{a_{k+1}} \in M_{a_{k+1}}$ is the root found by MD($M_{a_{k+1}}$)). Thus, a_{k+1} is the new last rebuilding stage.

The rebuilding stages of CS can be critical stages (because the current tree is broken and rebuilt). The other stages are non critical because the algorithm only removes from the current tree useless branches to obtain the new tree.

Note that this algorithm is polynomial because it uses Algorithm MD (MD is polynomial) and because updating a tree by removing useless branches can be done in polynomial time.

Note also that by construction, at each stage, the *tree* constraint is satisfied. Section 2.2 shows that it also respects the *quality* constraint for a level of quality $c = 4$.

2.2 CS respects the *quality* constraint

Theorem 1 shows that CS respects the *quality* constraint with a level of quality $c = 4$.

Theorem 1. *Let $G = (V, E, w)$ be a graph. For any sequence of withdrawals, at every stage i , $0 \leq i \leq m_0 - 1$ (i is the number of removed members), let T_i^* be an optimal (off-line) tree spanning M_i for the diameter. CS respects the quality constraint with a level of quality $c = 4$, i.e. for every i , $0 \leq i \leq m_0 - 1$, we have*

$$D_{T_i}(M_i) \leq 4D_{T_i^*}(M_i).$$

Proof.

- If i is a stage of rebuilding. In this case, $i = a_k$. Let $u_0, v_0 \in M_{a_k}$ be such that $d_{T_{a_k}}(u_0, v_0) = D_{T_{a_k}}(M_{a_k})$ (where T_{a_k} is the tree spanning M_{a_k} rooted in r^* built by CS at stage a_k). We have

$$\begin{aligned} D_{T_{a_k}}(M_{a_k}) &= d_{T_{a_k}}(u_0, v_0) \leq d_{T_{a_k}}(u_0, r^*) + d_{T_{a_k}}(r^*, v_0) \\ &\quad \text{(by triangular inequality)} \\ &= d_G(u_0, r^*) + d_G(r^*, v_0) \leq 2D_G(M_{a_k}) \\ &\quad \text{(because } T_{a_k} \text{ is a shortest path tree rooted} \\ &\quad \text{in } r^* \text{ and because } u_0, v_0, r^* \in M_{a_k}) \\ &\leq 2D_{T_{a_k}^*}(M_{a_k}) \leq 4D_{T_{a_k}^*}(M_{a_k}) \\ &\quad \text{(because for every tree } T \text{ spanning} \\ &\quad \text{a group } M, D_G(M) \leq D_T(M)) \end{aligned}$$

- Otherwise a_k is not a stage of rebuilding. Let j , $1 \leq j < m_{a_k} - \lfloor \frac{m_{a_k}}{2} \rfloor$ be the number of removed vertices after the last rebuilding, happening at stage a_k (i.e. j is such that $m_{a_k+j} \geq \lfloor \frac{m_{a_k}}{2} \rfloor + 1$). Let $M(r^*) = \{r^*, u_1^{r^*}, \dots, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^*}\}$ be the set returned by MD(M_{a_k}). As $M_{a_k+j} \subset M_{a_k}$ (by definition of the sequence of withdrawals) and $M(r^*) \subseteq M_{a_k}$ with $m_{a_k+j} \geq \lfloor \frac{m_{a_k}}{2} \rfloor + 1$ and $|M(r^*)| = \lfloor \frac{m_{a_k}}{2} \rfloor + 1$, we have $M_{a_k+j} \cap M(r^*) \neq \emptyset$. Thus, there exists $v \in M_{a_k+j} \cap M(r^*)$. As $v \in M(r^*)$, $v = r^*$ or $v = u_l^{r^*}$, with $l \leq \lfloor \frac{m_{a_k}}{2} \rfloor$. As $r^*, u_1^{r^*}, \dots, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^*}$ are sorted by non decreasing value of their distance to

r^* (see definition of Algorithm MD), we have

$$d_G(r^*, v) \leq d_G\left(r^*, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^*}\right) \quad (1)$$

Moreover, as Algorithm MD(M_{a_k}) finds r^* and $M(r^*)$ such that $d_G\left(r^*, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^*}\right) = \min\left\{d_G\left(r, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^r\right) : r \in M_{a_k}\right\}$, for every $r^0 \in M_{a_k+j} \subset M_{a_k}$ we have

$$d_G\left(r^*, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^*}\right) \leq d_G\left(r^0, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^0}\right) \quad (2)$$

As $m_{a_k+j} \geq \lfloor \frac{m_{a_k}}{2} \rfloor + 1$ and as $r^0, u_1^{r^0}, \dots, u_{m_{a_k}-1}^{r^0}$ are sorted by non decreasing value of their distance to r^0 , there exists $u_l^{r^0} \in M_{a_k+j}$ with $\lfloor \frac{m_{a_k}}{2} \rfloor \leq l \leq m_{a_k+j} - 1$ such that

$$d_G\left(r^0, u_{\lfloor \frac{m_{a_k}}{2} \rfloor}^{r^0}\right) \leq d_G\left(r^0, u_l^{r^0}\right) \quad (3)$$

By (1), (2), (3) and as r^0 and $u_l^{r^0}$ are in M_{a_k+j} , by definition of the diameter, we obtain

$$\exists v \in M_{a_k+j} \cap M(r^*) : d_G(r^*, v) \leq D_G(M_{a_k+j}) \quad (4)$$

Let $u^0 \in M_{a_k+j}$ and $v^0 \in M_{a_k+j}$ be such that $d_{T_{a_k+j}}(u^0, v^0) = D_{T_{a_k+j}}(M_{a_k+j})$ (where T_{a_k+j} is the tree spanning M_{a_k+j} built by CS at stage $a_k + j$). We have

$$\begin{aligned} D_{T_{a_k+j}}(M_{a_k+j}) &= d_{T_{a_k+j}}(u^0, v^0) = d_{T_{a_k}}(u^0, v^0) \\ &\quad \text{(since, by definition of Algorithm CS,} \\ &\quad \text{we have } T_{a_k+j} \subseteq T_{a_k}) \\ &\leq d_{T_{a_k}}(u^0, r^*) + d_{T_{a_k}}(r^*, v^0) \\ &\quad \text{(by triangular inequality)} \\ &= d_G(u^0, r^*) + d_G(r^*, v^0) \\ &\quad \text{(because } T_{a_k} \text{ is a shortest path tree rooted in } r^*) \\ &\leq d_G(u^0, v) + d_G(v, r^*) + d_G(r^*, v) + d_G(v, v^0) \\ &\quad \text{(by triangular inequality, using vertex } v \text{ of (4))} \\ &\leq 4D_G(M_{a_k+j}) \\ &\quad \text{(because } v \in M_{a_k+j}, u^0 \in M_{a_k+j}, \\ &\quad v^0 \in M_{a_k+j} \text{ and by (4))} \\ &\leq 4D_{T_{a_k+j}^*}(M_{a_k+j}) \\ &\quad \text{(because for every tree } T \text{ spanning} \\ &\quad \text{a group } M, D_G(M) \leq D_T(M)) \end{aligned}$$

In conclusion, for every i , $0 \leq i \leq m_0 - 1$, we obtain $D_{T_i}(M_i) \leq 4D_{T_i^*}(M_i)$. \square

2.3 CS leads to $O(\log i)$ critical stages

Theorem 2. *Let $G = (V, E, w)$ be a graph. For any sequence of withdrawals, let T_0, \dots, T_i ($0 \leq i \leq m_0 - 1$) be the sequence of trees constructed by CS. We have*

$$\#CS(T_0, \dots, T_i) \leq \lfloor \log_2(2i) \rfloor = O(\log i)$$

Proof. Two cases may occur:

- If $i < m_0 - \lfloor \frac{m_0}{2} \rfloor$, by definition of CS, there is no rebuilding stage. Thus, $CS(T_0, \dots, T_i) = 0$.
- Otherwise, $i \geq m_0 - \lfloor \frac{m_0}{2} \rfloor \geq \frac{m_0}{2}$ and we obtain

$$m_0 \leq 2i \tag{5}$$

Moreover, by definition of the sequence (a_k) and CS, if there are p rebuildings (that are critical stages), then p is such that

$$\begin{aligned} m_{a_{p+1}} < m_0 - i \leq m_{a_p} &\Rightarrow m_0 - i \leq \frac{m_0}{2^p} \quad (\text{by definition of sequence } (a_k), \\ &\quad \forall k, m_{a_k} \leq \frac{m_0}{2^k}) \\ &\Rightarrow m_0 - i \leq \frac{2i}{2^p} \quad (\text{by (5)}) \\ &\Rightarrow 1 \leq \frac{2i}{2^p} \quad (\text{by definition, } i \leq m_0 - 1) \\ &\Rightarrow p \leq \lfloor \log_2(2i) \rfloor \quad (\text{because } p \text{ is an integer}) \\ &\Rightarrow p \leq O(\log i) \end{aligned}$$

□

3 Lower bound for the number of critical stages of any algorithm

In this section, we prove that for *any* algorithm respecting the *tree* constraint and the *quality* constraint, for any sufficiently large i , there exists a particular sequence of withdrawals leading to at least $\Omega(\log i)$ critical stages. To prove that, we describe the graph G in Section 3.1. Then, we define the particular on-line sequence of withdrawals in Section 3.2 and prove the main result in Section 3.3.

3.1 Description of the graph G

Let $k, d, 0 \leq k \leq d$ and $3 \leq p$ be any integer. We define graphs $G_k^p = (V_k^p, E_k^p, w_k^p)$ recursively on k as follows:

- $G_0^p = (V_0^p, E_0^p, w_0^p)$ is the cycle of length p such that $\forall e \in E_0^p, w_0^p(e) = 2^d$.
- $\forall k, 1 \leq k \leq d$, we define $G_k^p = (V_k^p, E_k^p, w_k^p)$ as follows. $\forall v \in V_{k-1}^p$, let $C_v = (V_v^C, E_v^C, w_v^C)$ be a cycle of length p such that $v \in V_v^C, (V_v^C \setminus \{v\}) \cap V_{k-1}^p = \emptyset$ and $w_v^C(e) = \frac{2^{d-k}}{p^k}$.

$G_k^p = (V_k^p, E_k^p, w_k^p)$ is the graph such that:

- $V_k^p = V_{k-1}^p \cup \bigcup_{v \in V_{k-1}^p} V_v^C$
- $E_k^p = E_{k-1}^p \cup \bigcup_{v \in V_{k-1}^p} E_v^C$
- $\forall e \in E_{k-1}^p, w_k^p(e) = w_{k-1}^p(e)$ and $\forall e \in \bigcup_{v \in V_{k-1}^p} E_v^C, w_k^p(e) = w_v^C(e)$.

See Figure 1 for an illustration of G_k^p (with $k = 2$ and $p = 4$). We can now



Fig. 1. The graph G_2^4

define the graph $G = (V, E, w)$. Let $c \geq 1$ be the constant corresponding to the required level of quality and let d be a positive integer sufficiently large such that $i \leq \left\lfloor V_d^{\lceil 6c+2 \rceil} \right\rfloor - 1$ (where i is the number of removed vertices and $V_d^{\lceil 6c+2 \rceil} = M_0$ is the initial group). We set $G = G_d^{\lceil 6c+2 \rceil}$.

Definition of a cycle of level k .

We say that a cycle $C = (V^C, E^C, w)$ (subgraph of G) is of level k ($0 \leq k \leq d$) if each edge $e \in E^C$ has weight $w(e) = \frac{2^{d-k}}{p^k}$. See Figure 1 for an illustration of such cycle (Note that G_2^4 is too small to be a possible graph of the form $G_d^{\lceil 6c+2 \rceil}$, but this is just an illustration).

Definition of the subgraphs $G_k(v)$.

Let v be any vertex of the graph G ($v \in V$). Let k be the smallest index such that $C_k = (V_k^C, E_k^C, w)$ is the cycle of level k containing v ($v \in V_k^C$). We define $G_k(v) = (V_k(v), E_k(v), w)$ the subgraph induced by every vertices and edges which can be reached from vertex v by going through edges of weight strictly less than $\frac{2^{d-k}}{p^k}$ (i.e. by going through edges of cycles of level strictly more than k). See Figure 1 for an illustration of such subgraph.

3.2 Definition of the sequence of withdrawals $M_0 \supset \dots \supset M_i$

Let A be *any* online algorithm respecting the tree and quality of level c constraints. We use an *adaptive adversary* to define the sequence of withdrawals in the graph $G = (V, E, w)$ defined above.

We first define a generic sequence of withdrawals of vertices. Note that we do not specify each elementary stage of withdrawal, but only the “main” stages interesting for our analysis (stages of the form $i = \alpha(k, b)$). For every $k \geq 0$, for every $b \in \{0, 1\}$, for every $i = \alpha(k, b)$ ($0 \leq \alpha(k, b) \leq m_0 - 1$), let T_i be the tree spanning M_i constructed by Algorithm A at stage i . Note that at each stage, we have $\alpha(k, b) = |M_0| - |M_{\alpha(k, b)}|$ ($0 \leq \alpha(k, b) \leq m_0 - 1$). The sequence of withdrawals is defined as follows. We set $p = \lceil 6c + 2 \rceil$.

Basic Cases:

- At stage $\alpha(0, 0) = 0$, we have

$$M_{\alpha(0,0)} = V$$

As $T_{\alpha(0,0)}$ is a tree spanning $M_{\alpha(0,0)}$, it is necessarily made up of, amongst other things, all the edges of the cycle $C_0 = (V_0^C, E_0^C, w)$, except one edge e_0 . Let v_0^1 and v_0^2 be the two vertices connected by e_0 . The adaptive adversary now removes (one by one) from $M_{\alpha(0,0)}$ all the vertices in $\bigcup_{v \in V_0^C \setminus \{v_0^1, v_0^2\}} V_1(v)$ in order to obtain $M_{\alpha(0,1)}$.

- At stage $\alpha(0, 1)$, we have

$$M_{\alpha(0,1)} = V_1(v_0^1) \cup V_1(v_0^2)$$

The adaptive adversary now removes (one by one) from $M_{\alpha(0,1)}$ all the vertices in $V_1(v_0^1)$ in order to obtain $M_{\alpha(1,0)}$ (note that the adversary chooses arbitrarily to remove all the vertices in $V_1(v_0^1)$ rather than in $V_1(v_0^2)$).

Main Cases:

- At stage $\alpha(k, 0)$. Let $C_k = (V_k^C, E_k^C, w)$ be the cycle of level k such that $V_k^C \subset M_{\alpha(k-1,1)}$. We have

$$M_{\alpha(k,0)} = \bigcup_{v \in V_k^C} V_{k+1}(v)$$

As $T_{\alpha(k,0)}$ is a tree spanning $M_{\alpha(k,0)}$, it is necessarily made up of, amongst other things, all the edges of the cycle C_k , except one edge e_k . Let v_k^1 and v_k^2 be the two vertices connected by e_k . The adaptive adversary now removes (one by one) from $M_{\alpha(k,0)}$ all the vertices in $\bigcup_{v \in V_k^C \setminus \{v_k^1, v_k^2\}} V_{k+1}(v)$ in order to obtain $M_{\alpha(k,1)}$.

– At stage $\alpha(k, 1)$, we have

$$M_{\alpha(k,1)} = V_{k+1}(v_k^1) \cup V_{k+1}(v_k^2)$$

The adaptive adversary now removes (one by one) from $M_{\alpha(k,1)}$ all the vertices in $V_{k+1}(v_k^1)$ in order to obtain $M_{\alpha(k+1,0)}$ (note that the adversary chooses arbitrarily to remove all the vertices in $V_{k+1}(v_k^1)$ rather than in $V_{k+1}(v_k^2)$).

We specify with $\alpha(k, b)$ only the “main” stages of the sequence of withdrawals, corresponding to the stages where the adaptive adversary has to make a choice. Indeed, between two successive “main” stages $\alpha(k, 0)$ and $\alpha(k, 1)$ (resp. $\alpha(k, 1)$ and $\alpha(k + 1, 0)$), the vertices are removed one by one in any order. Note that we stop removing vertices after the last “main” stage, when exactly i vertices have been removed. See Figure 2 for an illustration of the six first “main” stages $\alpha(0, 0)$, $\alpha(0, 1)$, $\alpha(1, 0)$, $\alpha(1, 1)$, $\alpha(2, 0)$ and $\alpha(2, 1)$, where the successive trees are built by an arbitrary algorithm (Note that G_2^4 is too small to be a possible graph of the form $G_d^{\lceil 6c+2 \rceil}$, but this figure is just an illustration of a sequence of withdrawals).

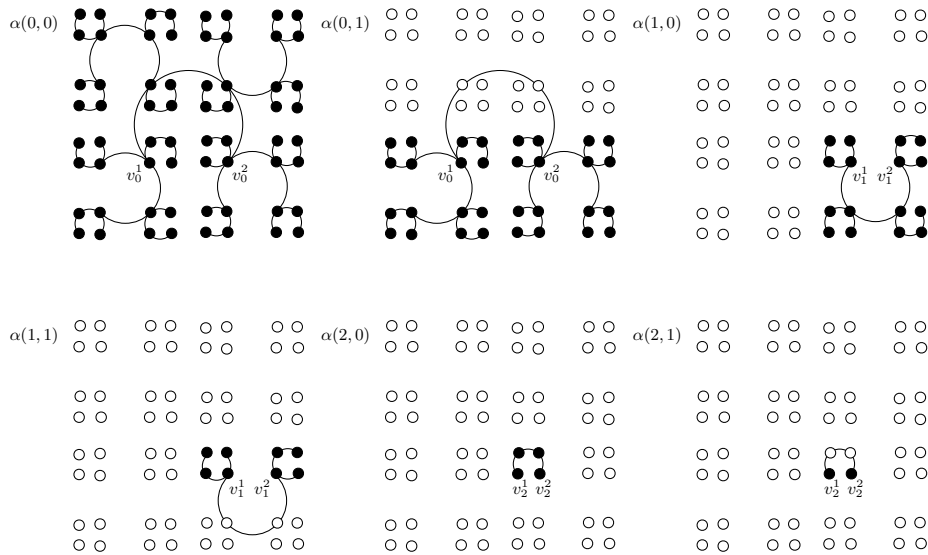


Fig. 2. Illustration of the sequence of withdrawals on graph G_2^4

3.3 Any algorithm leads to $\Omega(\log i)$ critical stages

Lemmas 1 and 2 are preliminary technical results (Lemma 1 is trivial. A proof can be found in [5]).

Lemma 1 Let $G = (V, E, w)$ be any graph. For every $M \subseteq V$, there exists a tree T^{off} spanning M such that

$$D_{T^{\text{off}}}(M) \leq 2D_G(M)$$

The following Lemma is central in our analysis. It describes sub-sequences of withdrawals where *at least* one rebuilding/critical stage occurs.

Lemma 2 Let $c \geq 1$ be any constant (representing the required level of quality). For every $k \geq 0$, let $T_{\alpha(k,0)}^*, T_{\alpha(k,0)+1}^*, \dots, T_{\alpha(k,1)}^*$ be the trees respectively spanning $M_{\alpha(k,0)}, M_{\alpha(k,0)+1}, \dots, M_{\alpha(k,1)}$ optimal for the diameter and let $T_{\alpha(k,0)}, T_{\alpha(k,0)+1}, \dots, T_{\alpha(k,1)}$ be any trees respectively spanning $M_{\alpha(k,0)}, M_{\alpha(k,0)+1}, \dots, M_{\alpha(k,1)}$. If for every i , $\alpha(k,0) \leq i \leq \alpha(k,1)$, we have $D_{T_i}(M_i) \leq c \cdot D_{T_i^*}(M_i)$, then

$$\#CS(T_{\alpha(k,0)}, T_{\alpha(k,0)+1}, \dots, T_{\alpha(k,1)}) \geq 1.$$

Proof. We prove Lemma 2 by contradiction. Suppose that there exists $k \geq 0$ such that for every i , $\alpha(k,0) \leq i \leq \alpha(k,1)$, the *quality* constraint is satisfied and there is no critical stage, i.e. there exists $k \geq 0$ such that for every i , $\alpha(k,0) \leq i \leq \alpha(k,1)$, we have $D_{T_i}(M_i) \leq c \cdot D_{T_i^*}(M_i)$ and $T_{\alpha(k,0)} \supseteq T_{\alpha(k,0)+1} \supseteq \dots \supseteq T_{\alpha(k,1)}$.

These trees are made up of, amongst other things, all edges of the cycle $C_k \subset G$, except one edge, noted e_k . We insist on the fact that, because there is no critical stage, this edge e_k is always the same in all trees $T_{\alpha(k,0)}, T_{\alpha(k,0)+1}, \dots, T_{\alpha(k,1)}$.

Let us focus now on stage $\alpha(k,1)$, where $M_{\alpha(k,1)} = V_{k+1}(v_k^1) \cup V_{k+1}(v_k^2)$. We lower bound $D_{T_{\alpha(k,1)}}(M_{\alpha(k,1)})$ and upper bound $D_{T_{\alpha(k,1)}^*}(M_{\alpha(k,1)})$ to show that at this particular stage, the quality constraint is not satisfied. This leads to the wanted contradiction and proves the Lemma.

– Lower bound of $D_{T_{\alpha(k,1)}}(M_{\alpha(k,1)})$.

As the two extremities v_k^1 and v_k^2 of the edge e_k are separated by a path made of $p-1 = \lceil 6c+1 \rceil$ edges of weight $\frac{2^{d-k}}{p^k}$ in $T_{\alpha(k,1)}$ we have

$$D_{T_{\alpha(k,1)}}(M_{\alpha(k,1)}) \geq (p-1) \frac{2^{d-k}}{p^k} \geq (6c+1) \frac{2^{d-k}}{p^k} \quad (6)$$

– Upper bound of $D_{T_{\alpha(k,1)}^*}(M_{\alpha(k,1)})$.

In order to upper bound $D_{T_{\alpha(k,1)}^*}(M_{\alpha(k,1)})$, we first upper bound $D_G(M_{\alpha(k,1)})$. By construction of the graph G , two cases may occur:

1. If $k = d$, there is no cycle of level $k+1$ in G . Thus, we have

$$D_G(M_{\alpha(k,1)}) = w(e_k) = \frac{2^{d-k}}{p^k} \leq 3 \frac{2^{d-k}}{p^k}$$

2. If $k \leq d - 1$, we have

$$\begin{aligned}
D_G(M_{\alpha(k,1)}) &\leq D_G(V_{k+1}(v_k^1)) + d_G(v_k^1, v_k^2) + D_G(V_{k+1}(v_k^2)) \\
&\leq \sum_{e \in E_{k+1}(v_k^1)} w(e) + w(e_k) + \sum_{e \in E_{k+1}(v_k^2)} w(e) \\
&\quad (\text{because for every graph or subgraph } \\
&\quad G = (V, E, w), D_G(V) \leq \sum_{e \in E} w(e)) \\
&= \sum_{l=k+1}^d \frac{2^{d-l}}{p^l} p^{l-k} + \frac{2^{d-k}}{p^k} + \sum_{l=k+1}^d \frac{2^{d-l}}{p^l} p^{l-k} \\
&\leq \frac{2}{p^k} \sum_{l=k+1}^d 2^{d-l} + \frac{2^{d-k}}{p^k} \leq 2 \frac{2^{d-k}}{p^k} + \frac{2^{d-k}}{p^k} = 3 \frac{2^{d-k}}{p^k}
\end{aligned}$$

Moreover, by Lemma 1, there exists a tree $T_{\alpha(k,1)}^{\text{off}}$ spanning $M_{\alpha(k,1)}$ such that $D_{T_{\alpha(k,1)}^{\text{off}}}(M_{\alpha(k,1)}) \leq 2D_G(M_{\alpha(k,1)})$. Thus, as $T_{\alpha(k,1)}^*$ is a tree spanning $M_{\alpha(k,1)}$ optimal for the diameter, we have

$$D_{T_{\alpha(k,1)}^*}(M_{\alpha(k,1)}) \leq D_{T_{\alpha(k,1)}^{\text{off}}}(M_{\alpha(k,1)}) \leq 2D_G(M_{\alpha(k,1)}) \leq 6 \frac{2^{d-k}}{p^k} \quad (7)$$

By (6) and (7), we obtain

$$\frac{D_{T_{\alpha(k,1)}}(M_{\alpha(k,1)})}{D_{T_{\alpha(k,1)}^*}(M_{\alpha(k,1)})} \geq \frac{(6c+1) \frac{2^{d-k}}{p^k}}{6 \frac{2^{d-k}}{p^k}} \geq c + \frac{1}{6} > c$$

This result contradicts the assumption that the *quality* constraint is satisfied. Thus, Lemma 2 is proved by contradiction. \square

The following Theorem shows that if the *tree* constraint and the *quality* constraint are satisfied, *any* algorithm leads to $\Omega(\log i)$ critical stages, where i is the number of removed vertices.

Theorem 3. *Let $c \geq 1$ be any constant. For any algorithm A , for every sufficiently large i , there exists a graph G_0 , there exists $M_0 \supset \dots \supset M_i$, such that if Algorithm A returns a sequence of trees T_0, \dots, T_i respectively spanning $M_0 \supset \dots \supset M_i$ respecting the quality constraint of level c , then*

$$\#CS(T_0, \dots, T_i) = \Omega(\log i)$$

Proof. Let $c \geq 1$ be any constant. We set $p = \lceil 6c + 2 \rceil$. Let i be the number of removed vertices. There exists d and G_0 (where G_0 is graph G , defined in Section 3.1), there exists $M_0 \supset \dots \supset M_i$ (the sequence defined in Section 3.2) such that

$$\alpha(d-1, 1) \leq i \leq \alpha(d, 1) \leq |V| = p^d$$

Thus, we have $i \leq p^d \Rightarrow \log_p i \leq d \Rightarrow \log_p i \leq d$. And as $p = \lceil 6c + 2 \rceil$ is a constant, we have $d \geq \Omega(\log i)$. Moreover, by Lemma 2, we have

$$\begin{cases} \#CS(T_{\alpha(0,0)}, T_{\alpha(0,0)+1}, \dots, T_{\alpha(0,1)}) \geq 1 \\ \#CS(T_{\alpha(1,0)}, T_{\alpha(1,0)+1}, \dots, T_{\alpha(1,1)}) \geq 1 \\ \vdots \\ \#CS(T_{\alpha(d-1,0)}, T_{\alpha(d-1,0)+1}, \dots, T_{\alpha(d-1,1)}) \geq 1 \end{cases}$$

$$\begin{aligned} &\Rightarrow \#CS(T_{\alpha(0,0)}, \dots, T_{\alpha(d-1,1)}) \geq d \\ &\Rightarrow \#CS(T_0, \dots, T_i) \geq d \quad (\text{because } i \geq \alpha(d-1, 1)) \\ &\Rightarrow \#CS(T_0, \dots, T_i) \geq \Omega(\log i) \quad (\text{because } d \geq \Omega(\log i)) \end{aligned}$$

□

Theorem 2 and Theorem 3 show that Algorithm CS is worst case optimal in order of magnitude for the number of critical stages criterion.

4 Conclusion

We have proposed an algorithm, called CS, solving an on-line covering problem of members by respecting the following *quality* constraint: For each stage of withdrawal, the diameter between members induced by the built tree is at most a constant time the best possible value. Moreover, our algorithm is easy to use. Indeed, for a stage of withdrawal, either it breaks the tree and rebuilds a new one which is a tree of shortest paths (only $O(\log i)$ times, where i is the number of removed members), or it just updates the current tree by removing useless branches (in all the other cases).

Moreover, our algorithm is worst case optimal in order of magnitude for the number of critical stages: It leads to $O(\log i)$ critical stages and we showed that *any* algorithm leads to $\Omega(\log i)$ critical stages in the worst case. We also have proved that the number of elementary changes per stage (see equivalent definition in [4]) is constant in average. Due to space limitation, we do not include these results. Note that we only consider the decremental problem because the incremental problem (adding new members in the current tree) considering the diameter as quality constraint is trivial. Indeed, plugging each new member with a shortest path to the initial member leads to 0 critical stage with a level of quality $c = 2$. We also have results with another objective function than the diameter. Indeed, concerning the average distance between members of the groups, we proved similar results in [7] for the incremental version of the problem. We are now currently working on mixing additions and withdrawals.

Acknowledgments

The authors wish to thank the anonymous referees for their very useful comments.

References

1. G. AUSIELLO, P. CRESCENZI, G. GAMBOSI, V. KANN, A. MARCHETTI SPACCAMELA, AND M. PROTASI, *Complexity and approximation*, Springer, 1999.
2. A. BORODIN AND R. EL-YANIV, *Online computation and competitive analysis*, Cambridge University press, 1998.
3. D. HOCHBAUM, *Approximation algorithms for NP-hard problems*, PWS publishing compagny, 1997.
4. M. IMASE AND B. WAXMAN, *Dynamic steiner tree problem*, SIAM J. Discr. Math., 4 (1991), pp. 369–384.
5. C. LAFOREST, *A good balance between weight and distances for multipoint trees*, in International Conference On Principles Of DIstributed Systems 2002, pp. 195–204.
6. S. RAGHAVAN, G. MANIMARAN, AND C. S. R. MURTHY, *A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees*, IEEE/ACM (SIGCOMM), ACM Press, 7 (1999).
7. N. THIBAUT AND C. LAFOREST, *An optimal rebuilding strategy for an incremental tree problem*, submitted in 2004 to journal of interconnection networks.
8. B. WAXMAN, *Routing of multipoint connections*, IEEE Journal on Selected Areas in Communications, 6 (1988), pp. 1617–1622.