Distributed algorithms for weighted problems in sparse graphs

A. Czygrinow[*], M. Hańćkowiak[†]

June 2, 2005

Abstract

We study distributed algorithms for three graph-theoretic problems in weighted trees and weighted planar graphs. For trees, we present an efficient deterministic distributed algorithm which finds an almost exact approximation of a maximum-weight matching. In addition, in the case of trees, we show how to approximately solve the minimum-weight dominating set problem. For planar graphs, we present an almost exact approximation for the maximum-weight independent set problem.

1 Introduction

We consider a distributed model of computations introduced by Linial in [L92]. In this model, a network is represented by an undirected graph, each vertex of the graph corresponds to a processor, and an edge corresponds to a connection between processors. The network is synchronized and computations proceed in rounds. In a single round vertices can send messages to their neighbors, can receive messages from their neighbors, and can perform some local computations. Neither the amount of local computations nor the lengths of messages are restricted in any way. In such a distributed network, objective of processors is to compute some global function (for example a maximum independent set) of the underlying graph. In a connected graph on n vertices any function of the graph can be computed in $O(n)$ rounds and the goal is to do it much faster. Usually a distributed algorithm is considered efficient if its running time is poly-logarithmic in n. Distributed model of computations is entirely different than a massively parallel PRAM model. In the latter one processors have access to a shared memory and can use this means to communicate with each other. As a result communication between processors in the parallel model of computations is not restricted in any way. In contrast, in the distributed model of computations it is the underlying graph which restricts the communication. In this graph vertices are confined to the local subgraphs, as a vertex v can learn only a subgraph of radius $polylog(n)$ around v in a poly-logarithmic number of rounds. Based on that local information a global function of the underlying graph is determined. In addition, we assume that an underlying graph is weighted, with nonnegative weights defined either on the vertex set or the edge set. Although weights impact significantly a strategy taken to design algorithms for our problems, they do not influence the communication model.

In this paper, we will focus on two types of network topologies: trees and planar graphs. Both are classical families of graphs that appear in many different situations. Trees are the most basic topologies in graph theory and understanding distributed complexity for this class of graphs seems to be necessary to hope for a further progress. The class of planar graphs is a natural generalization. In addition trees can appear as spanning subgraphs of networks with arbitrary topology as solving some problems in a spanning tree immediately gives a feasible solution in the network. We will consider three classical problems in graph theory. In the case when a network is a tree we will discuss algorithms for two

[*]Department of Mathematics and Statistics, Arizona State University, Tempe, AZ,85287-1804, USA, andrzej@math.la.asu.edu.

[†]Faculty of Mathematics and Computer Science, Adam Mickiewicz University, Poznań, Poland, mhanckow@main.amu.edu.pl.

problems. For trees with weights on edges we will study the *Maximum-Weight Matching* problem and for trees with weights on vertices we will consider the *Minimum-Weight Dominating Set* problem. In the Maximum-Weight Matching problem we want to find a matching of total weight which is the largest possible. In the Minimum-Weight Dominating Set problem the objective is to find a dominating set such that the sum of weights on vertices of the set is the smallest possible. Both problems have many natural applications. For example, the Minimum-Weight Dominating Set problem can arise naturally in a distributed network when considering a problem of nearly optimal placement of servers in a network.

Our first algorithm approximates a maximum-weight matching in a tree. Note that finding the exact maximum is unfeasible for a distributed algorithm even in very simple cases as to find a maximum matching in an unweighted path on n vertices, $\Omega(n)$ rounds are required as proved by Linial in [L92]. Consequently recent research has focused on designing distributed approximation algorithms (see an excellent survey of Elkin [E04] on distributed approximation algorithms). It is interesting to note that the state of knowledge in the case of randomized distributed algorithms and deterministic distributed algorithms is far apart. In fact it is even not known if the maximal independent set problem admits an efficient deterministic distributed algorithm yet simple randomized algorithms for the problem are known. Let ω^* denote the weight of a maximum-weight matching. In [WW04] two randomized distributed approximations are given for the matching problem. For weighted trees, M. Wattenhofer, R. Wattenhofer ([WW04]) gave a $O(1)$-time randomized algorithm which finds with high probability a matching M of weight $\Omega(\omega^*)$. In addition, [WW04] contains a randomized algorithm which in an arbitrary graph finds in $O(\log^2 n)$ rounds a matching M which with high probability has weight $\Omega(\omega^*)$. In the case of unweighted graphs much more is known. When a graph is bipartite then a matching M with $(1-\epsilon)\omega^*$ edges can be found in $O(\log^{1/\epsilon} n)$ rounds ([CH03]) and in the case of general graphs, a $2/3$-approximation can be found in the poly-logarithmic number of rounds ([CHS04],[CHSz04]). Both of these algorithms however are based on the augmenting paths technique which can be applied only to the unweighted version of the problem. In the case of a dominating set, [JRS02] gives an efficient $O(\log n \log \Delta)$-time randomized algorithm which finds a matching of the expected ratio of $O(\log \Delta)$ where Δ is the maximum degree in the graph. In the unweighted case, a recent paper of Kuhn and Wattenhofer [KW03] uses a distributed linear programming approach to find a dominating set of the expected ratio of $k\Delta^{2/k}\log\Delta$ in $O(k^2)$ rounds. On the deterministic side, in [KP95], Kutten and Peleg showed how to find a k-dominating set (every vertex is within distance k of the set) of size $n/(k+1)$ in $O(k\log^* n)$ rounds in general graphs. Curiously, the approach from [KP95] is based on finding first a k-dominating set in a tree. The algorithm for trees is further used to design an algorithm which finds a k-dominating set in general graphs.

In this paper, we present two deterministic algorithms for trees. The first one finds a matching M of weight which is at least $(1-O(1/\log n))\omega^*$ in a tree on n vertices. The algorithm runs in the number of rounds which is $polylog(n)$ but assumes that the value of the maximum weight is known to vertices in the graph. The second algorithm for trees approximates the minimum-weight dominating set provided a certain global parameter L of a tree is known to all nodes in the network. The algorithm finds a dominating set D such that $\omega(D) \leq (1+O(1/\log n))\omega^*$ where ω^* is the weight of an optimal solution. Again the running time of the algorithm is poly-logarithmic in the number of vertices. The approach to both problems is based on clustering. In fact, our main auxiliary algorithm can be regarded as a weighted analog of the ruling-forest clustering from [AGLP89]. However, to approach weighted problems, our clusters must satisfy a few additional properties. Our third algorithm gives an almost exact approximation of the *Maximum-Weight Independent Set* problem in planar graphs. In this problem, given a graph with weights on vertices, we want to find an independent set such that the sum of weights is the largest possible. We present a deterministic algorithm which in a planar graph on n vertices finds an independent set weight of which is at least $(1-O(1/\log n))\,\omega^*$ where ω^* is the optimum. The algorithm runs in a poly-logarithmic number of rounds and uses a different clustering procedure. Again finding the exact maximum even in the case of unweighted paths requires $\Omega(n)$ rounds. In the case of general graphs, efficient deterministic algorithms even for the maximal independent set problem are not known.

The rest of the paper is organized as follows. In the next section, we present algorithms for trees. The section is divided into two parts, in the first part we present the clustering algorithm and in the second, we show how to apply the algorithm to matchings and dominating sets. In the last section, we present an algorithm for the independent set problem in planar graphs. Again the discussion is divided into two subsections. First we mention some easy auxiliary facts and then we present the main algorithm.

2 Trees

In this section, we will discuss algorithms for trees. We will give an almost exact approximation for the maximum-weight matching problem and for the minimum-weight dominating set problem. Algorithms for both problems are based on clustering and our first objective will be to give an efficient distributed algorithm which finds a clustering of a tree such that clusters have poly-logarithmic diameter and the total weight of edges connecting different clusters is small. In Section 2.1 we present the clustering algorithm-procedure HeavyRulingForest. The algorithm is used in Section 2.2 in WMatchinginTree to find a matching and in WDSinTree to find a dominating set.

2.1 Clustering algorithm

In this section, we will present a clustering algorithm. The main procedure of this section, HeavyRulingForest, finds a clustering of a weighted tree $T = (V, E, \omega)$ on n vertices which has two properties:

- Diameter of each cluster is $polylog(n)$ (Lemma 2.4).
- If there is an edge of weight ω connecting two clusters X_1 and X_2 then each X_i contains a path of length $\Omega(\log n)$ such that each edge on the path has weight at least $\omega/2$ (Lemma 2.5).

The second property implies that the total weight of edges connecting different clusters is $O(W/\log n)$ where $W = \sum_{e\in E} \omega(e)$. In the case of unweighted trees a similar effect can be accomplished by invoking a ruling forest procedure of Awerbuch et al. [AGLP89]. However the fact that a graph is weighted adds complexity to the problem and a slightly different strategy must be pursued. Finding clusters in T is divided into three procedures: LargeIndependentSet, ModifyClusterSet, and HeavyRulingForest. We will give a general idea of the clustering algorithm by starting with the main (and the last) procedure HeavyRulingForest. Let ω_{max} denote the maximum weight of an edge, i.e. $\omega_{max} = \max_{e\in E} \omega(e)$. Procedure HeavyRulingForest starts with a clustering into singletons and in the ith iteration edges with weights from the interval $(\omega_{max}/2^{i+1}, \omega_{max}/2^{i}]$ are considered and new clusters are formed. For example, suppose that there are only two weights in T, ω_{min} and ω_{max} with $2\omega_{min} < \omega_{max}$. Then in the first iteration of HeavyRulingForest, edges of weight ω_{max} are considered and clusters with these edges are formed in a sub-tree of T. Next the edges of weight ω_{min} are exposed, old clusters are enlarged, and possibly new clusters are formed. As a result either large clusters are obtained in the first iteration and so the weight of each edge in these clusters is ω_{max}, or the clusters are enlarged in the second iteration. If an edge of weight ω_{max} connects two different clusters after the first iterations then these clusters have "large" diameter and they will not be enlarged in the second iteration. In the case clusters are enlarged in the second iteration, edges connecting different clusters after the second iteration have the weight of ω_{min} and the second property of clustering follows from the lower bound for the diameter of a cluster. Clearly the main problem which must be addressed in the algorithm is how to incorporate new edges into old clusters. This is accomplished by ModifyClusterSet which assumes that a set of old clusters is provided and two graphs: G of old edges and H of new edges on the same vertex set are given. The algorithm considers the cluster graph with the vertex set equal to the set of old clusters and an edges between two clusters coming from H. Note that $G \cup H$ is a forest and a cluster induces a connected subgraph of $G \cup H$. As a result the cluster graph is a forest as well and there is at most one edge in H connecting vertices from two different clusters. ModifyClusterSet creates new clusters from the old ones by first finding a set of vertex-disjoint stars in the cluster graph. The process is then iterated so that all edges from H are examined. Consider one such star $\{D, C_1, C_2, \ldots, C_k\}$ around D. First the algorithm checks if every vertex from C_i is within a short distance of D. If it is the case then vertices from C_i are added to D and a new enlarged cluster is created. This one-sided (from C_i's to D) verification is however not enough as it can happen that C_i is simply a large cluster and so it is not the case that all vertices from C_i are close to a possibly small cluster (maybe even a single vertex cluster) D. Consequently in the next step, we check if the new cluster obtained from D is within small distance (in the same sense as above) to any of the C_i's which were not added to D. If it is the case then new D is added to any such C_i. Now suppose that C_i and D are not incorporated into a new cluster. Then there is a vertex in C_i which is far from D and there is a vertex in D which is far from C_i. It is then easy to see that diameters of both C_i and D must be large in this iteration. The final piece of the algorithm is to find a large set of vertex-disjoint stars. This is done by an easy procedure LargeIndependentSet which finds a large (constant fraction of vertices) independent set in a cluster graph.

We will now fix some notation and terminology. A *cluster* in a graph $G = (V, E)$ is a connected induced subgraph of G. We will often identify clusters with sets of vertices that induce them and so if $G[X]$ is a connected subgraph induced by X, then X will also be called a cluster in G.

Let $G = (V, E, \omega)$ be a graph with $\omega : E \rightarrow \mathcal{R}^+$. All of the metric properties of G are defined ignoring ω and so the length of a path connecting two vertices of G is the number of edges in the path. If $v, u \in V$ then $dist_G(v, u)$ is the length of a shortest path connecting v with u and for $v \in V$, $U \subseteq V$,

$$dist_G(v, U) = \min_{u \in U} dist_G(v, u).$$

In addition, for $U \subseteq V$ let $diam_G(U)$ be the largest distance (in G) between any two vertices from U, i.e.

$$diam_G(U) = \max_{u,v \in U} dist_G(u, v).$$

Finally for $v \in V$, $deg_G(v)$ is the number of edges incident to v (we will often drop the subscript in $\deg_G(v)$ if G is clear from the context).

We note an easy lemma.

Lemma 2.1 *Let $G = (V, E)$ be a forest and let $A = \{v \in V | deg(v) \leq 2\}$. Then $|A| \geq |V|/2$.*

Proof. Note that adding edges to G can only decrease $|A|$ and so we can assume that G does not have isolated vertices. Suppose that $|A| < |V|/2$. Then

$$\sum_{v \in V} deg(v) \geq 3(|V| - |A|) + |A| > 2|V|$$

which contradicts the fact that $|E| \leq |V| - 1$. □

As explained above the most basic component of our algorithm is an easy procedure that finds a large independent set in a forest.

```
LargeIndependentSet
Input: Forest $F$.
Output: An independent set $I$ such that $|I| = \Omega(|V(F)|)$.

  1. Each vertex $v$ such that $deg(v) \leq 2$ is added to $A$.

  2. Find a maximal independent set $I$ in subgraph of $F$ induced by $A$ using the Cole-Vishkin procedure
     [CV86].

  3. Return $I$.
```

Note that the Cole-Vishkin procedure from [CV86] finds an maximal independent set in a graph on n vertices with constant maximum degree in $O(\log^* n)$ rounds. Since each connected component of $F[A]$ is a path, $|I| \geq |A|/3 \geq |V|/6$. The main clustering procedure iterates over ranges of weights and exposes edges that have weights in a given range. When new edges are exposed a modification procedure is invoked which glues together clusters which have "small" diameter. The algorithm takes the following arguments: graph G , graph H on the same vertex set with property that $G \cup H$ is a forest, and the set of clusters $\mathcal{C}$ in G. There are $O(\log n)$ main iterations in the procedure. In each iteration, first a large independent set I is found in the subgraph Aux of the cluster graph which contains non-isolated vertices, second the set of stars is obtained by selecting one edge incident to each vertex from I, finally the clusters in each star are connected to form a larger cluster if their diameters are small (steps 2(e) and 2(f)).

ModifyClusterSet
Input: Graphs G and H on the same vertex set; $G \cup H$ is a forest. Set $\mathcal{C}$ of clusters in G. Integer K.
Output: Set of cluster $\bar{\mathcal{C}}$ in $G \cup H$.

1. Consider the cluster graph with vertex set $\mathcal{C}$ and two clusters connected if there is an edge in H connecting a vertex from one cluster to a vertex in another. The cluster graph is a forest. Let Aux be the subgraph of the cluster graph induced by clusters of degree at least one.

2. Iterate $O(\log n)$ times:

   (a) Call LargeIndependentSet in Aux to find an independent set I in Aux with properties: $|I| = \Omega(|V(Aux)|)$ and
   $$\forall_{C \in I} 1 \leq deg_{Aux}(C) \leq 2.$$

   (b) $W := V(Aux) \setminus I$.

   (c) For each $C \in I$, in parallel, select one edge e_C incident to C and let $D_C \in W$ denote the second endpoint of e_C. The edge e_C is called a *special edge.* Note that the graph induced by $\{e_C | C \in I\}$ is a set of stars around the vertices from $\{D_C | C \in I\}$ and recall that D_C and C are vertices in the auxiliary graph which correspond to sets of vertices (clusters) in $G \cup H$.

   (d) For every $C \in I$, in parallel, if
   $$\forall_{w \in C} dist_{G \cup H}(w, D_C) \leq K \tag{1}$$
   then add all vertices from cluster C to cluster D_C and delete C from I.

   (e) Let $\bar{D} := D_C$ and so $\bar{D}$ is the set of all vertices which were in the original D_C in step 2(c) and vertices added to D_C in step 2(d).

   (f) Let $\{\bar{D}_i\}$ denote the set of clusters created in 2(e). For every $\bar{D}_i$ in parallel, if there exists a cluster $C \in I$ (not deleted in 2(d)) such that $D_C \subseteq \bar{D}_i$ and
   $$\forall_{w \in \bar{D}_i} dist_{G \cup H}(w, C) \leq K \tag{2}$$
   then add all vertices from the cluster $\bar{D}_i$ to the cluster C. Let $\bar{C}$ denote the cluster obtained from C.

   (g) Modify $\mathcal{C}$ as follows. Let $\bar{\mathcal{C}}$ consists of all new clusters and all unmodified clusters from $\mathcal{C}$.

   (h) Modify Aux: $V(Aux)$ contains clusters from $\bar{\mathcal{C}}$ of degree at least one. In addition, if there exists a special edge connecting two clusters from $\bar{\mathcal{C}}$ then delete the edge from Aux.

3. Return $\bar{\mathcal{C}}$.

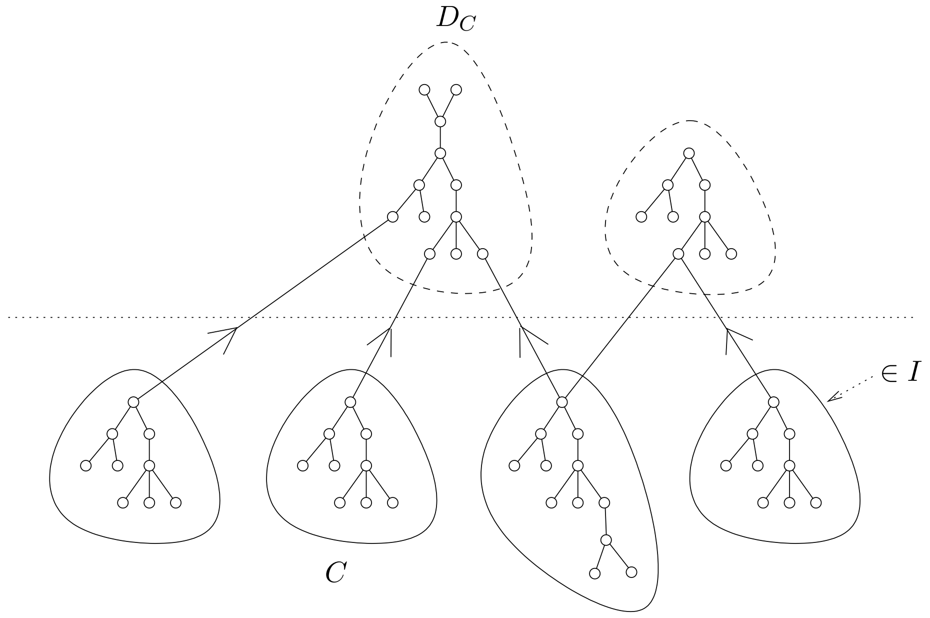

Figure 1: Modifying clusters in a tree

We now proceed with the analysis of ModifyClusterSet. First we will prove that the diameter of new clusters can increase by an additive factor of $O(K \log n)$. In the second lemma we will show that if $G \cup H$ contains connected components of large diameter than clusters contained in these components must have a large diameter as well.

Lemma 2.2 *Let $P = \max_{X \in \mathcal{C}} diam_G(X)$. Then the set of clusters $\bar{\mathcal{C}}$ obtained by* ModifyClusterSet *has the property*

$$\max_{\bar{X} \in \bar{\mathcal{C}}} diam_{G \cup H}(\bar{X}) \leq P + O(K \log n).$$

Proof. There are $O(\log n)$ iterations of step 2 in ModifyClusterSet and so it is enough to prove that in each iteration the diameter of a cluster can increase by an additive factor of $O(K)$. To that end, let C, C' be two cluster from I such that $D_C = D_{C'}$ and suppose that both C and C' are added to $\bar{D}$ in step 2(d). Then, by (1), for $v \in C, v' \in C'$,

$$dist_{G \cup H}(v, v') \leq 2K + diam_G(D_C).$$

Consequently, the diameter of D_C increases by $O(K)$. Similarly, suppose that $\bar{D}$ is added to cluster C in step 2(f). By (2), if $v \in \bar{D} \cup C$ and $w \in \bar{D} \cup C$ then

$$dist_{G \cup H}(v, w) \leq 2K + diam_G(C).$$

Therefore the diameter of C increases by $O(K)$. □

Lemma 2.3 *Let $\bar{\mathcal{C}}$ be the set of clusters in $G \cup H$ obtained by* ModifyClusterSet*. If Q is a connected component in $G \cup H$ then for every cluster $X \in \bar{\mathcal{C}}$ with $X \subseteq V(Q)$,*

$$diam_{G \cup H}(X) \geq \min\{K, diam_{G \cup H}(Q)\}.$$

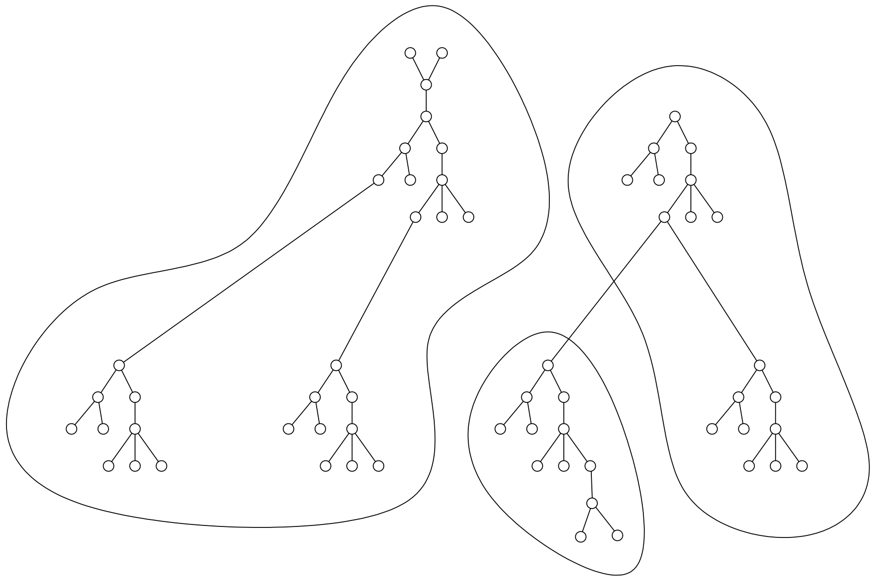

Figure 2: New clusters after modification

Proof. Let X, Y be two clusters in Q such that there is an edge e connecting X and Y. First observe that an edge from H will be deleted from the auxiliary graph during the execution of the algorithm (step 2(h)) only if it is a special edge in some iteration i. In addition the number of special edges in an iteration is equal to the constant fraction of the edges from Aux. Therefore after $O(\log n)$ iterations of step 2, Aux will be an empty graph. Consequently, the edge e was special in some iteration of step 2. Let C and D_C denote two clusters in iteration i such that $e = \{C, D_C\}$. We can assume that $C \subseteq X$ and $D_C \subseteq Y$. Since C was not added to D_C in step 2(d), condition (1) was not satisfied and so for some $w \in C$, $dist_{G\cup H}(w, D_C) > K$. Since e connects C with D_C, this implies that $diam_{G\cup H}(C) \geq K$. Consequently, $diam_{G\cup H}(X) \geq K$. Now let $\bar{D}$ denote the cluster obtained from D_C by possibly adding vertices in step 2(d). There are two possibilities:

1. $\bar{D}$ was added to some cluster $\bar{C}$ in the step 2(f) or
2. condition (2) was not satisfied for any C with $D_C \subseteq \bar{D}$ and $\bar{D}$ was left intact.

In the first case, both $\bar{D}$ and $\bar{C}$ are subsets of Y and $D_{\bar{C}} = D_C$. In addition, $\bar{C}$ was not added to $\bar{D}$ in step 2(d). Then however, by the above argument, $diam_{G\cup H}(\bar{C}) \geq K$ and so $diam_{G\cup H}(Y) \geq K$. In the second case, there exists a $w \in \bar{D}$ such that $dist_{G\cup H}(w, C) > K$ and so $diam_{G\cup H}(\bar{D}) \geq K$. Consequently, $diam_{G\cup H}(Y) \geq K$. □

We can now describe the main procedure which finds clusters in a weighted tree. The clusters have two properties that we indicated before: The diameter of a cluster is $polylog(n)$ (Lemma 2.4) and for every edge e connecting two different clusters X and Y, the weight of e is much smaller than the total weight of edges in X and much smaller than the total weight of edges in Y (Lemma 2.5).

HeavyRulingForest
Input: Weighted tree F with maximum weight ω_{max}.
Output: Set of cluster C.

1. Let $i := 0$ and $G := \emptyset$.

```
2. For a vertex $v$, let $C_v := \{v\}$. Let $C := \bigcup_v \{C_v\}$.
3. While $i \leq 2\log n$ do:
    (a) $H := \{e \in F | \omega(e) \in (\omega_{max}/2^{i+1}, \omega_{max}/2^i]\}$
    (b) Invoke ModifyClusterSet with $G, H, C$ and $K = \log n$.
    (c) Let $C'$ denote the obtained set of clusters.
    (d) $G := G \cup H$, $C := C'$, $i := i + 1$.
4. Return $C$.
```

We summarize the properties of clusters obtained by HeavyRulingForest in the next two lemmas.

Lemma 2.4 *For every cluster X obtained from* HeavyRulingForest*,*

$$diam_F(X) = O(\log^3 n).$$

Proof. There are $O(\log n)$ iterations of the while loop in step 3. By Lemma 2.2 in each iteration the diameter of a cluster can increase by an additive factor of $O(\log^2 n)$. Consequently the diameter of a cluster is $O(\log^3 n)$. $\square$

Lemma 2.5 *Let X_1, X_2 be two clusters in the set of clusters obtained by* HeavyRulingForest*. If there is an edge connecting X_1 with X_2 of weight ω then each of X_i's ($i = 1, 2$) contains a path of length* $\log n$ *such that every edge on the path has the weight at least $\omega/2$.*

Proof. Let e denote the edge connecting X_1 and X_2. Assume that the weight of e is ω and $\omega \in (\omega_{max}/2^{i+1}, \omega_{max}/2^i]$. Then e was an edge of H in the ith iteration of HeavyRulingForest. Since e connects X_1 and X_2 after all iterations it had to connect two clusters X_1' and X_2' in the ith iteration. Then however, by Lemma 2.3, $diam_{G\cup H}(X_1')$ and $diam_{G\cup H}(X_2')$ are at least $\log n$. Since X_1', X_2' induce connected subgraphs of F, $diam_{G\cup H}(X_i') = diam_F(X_i')$. In addition, all edges in graphs X_1' and X_2' have weights which are at least $\omega/2$ as only such edges where considered in iterations up to i. Consequently X_i contains a path of length $\log n$ with each edge of weight $\omega/2$. $\square$

2.2 Applications

In this section, we will show how to use HeavyRulingForest clustering procedure to approximate a maximum-weight matching and a minimum-weight dominating set in trees. Note that HeavyRulingForest is general enough to handle other problems. For example, we can obtain similar approximations for the maximum-weight independent set problem (this of course also follows from the result in the next section) or the maximum-cut problem. Our first algorithm will approximate a maximum-weight matching. Let $F = (V, E, \omega)$ be a weighted tree on n vertices with $\omega : E \to \mathcal{R}^+$ and let ω_{max} denote the maximum weight. The idea of the algorithms is very basic. First find a clustering using HeavyRulingForest. Disregard edges connecting different clusters and as each cluster has a poly-logarithmic diameter find in each cluster a matching of the maximum weight. Finally return the union of these matchings.

```
WMatchinginTree
Input: Weighted tree $F$ with maximum weight of an edge $\omega_{max}$.
Output: Weighted matching $M$.
1. If $e$ has weight $\omega(e) < \omega_{max}/n^2$ then let $\omega(e) := \omega_{max}/n^2$. Let $T$ denote the tree with new weights.
2. Use HeavyRulingForest to find the set of clusters $\mathcal{C}$ in T.
```

3. In each cluster X from $\mathcal{C}$ find a maximum weighted matching M_X.

4. Return $M := \bigcup_{X \in \mathcal{C}} M_X$.

Theorem 2.6 *Let F be a weighted tree on n vertices, let ω^* denote the weight of a maximum-weight matching in F. Then* WMatchinginTree *finds a matching M such that*

$$\omega(M) \geq (1 - O(1/\log n))\omega^*.$$

The algorithm runs in polylog(n) rounds.

Proof. For a subset $M \subseteq E(F)$, let $\omega_T(M)$ ($\omega_F(M)$) denote the weight of M in T (in F). In addition let $\omega^*_F = \omega^*$ and ω^*_T be the weight of the maximum-weight matching in T. We have

$$\omega_T(M) \leq \omega_F(M) + \frac{\omega_{max}}{n}$$

and of course

$$\omega_T(M) \geq \omega_F(M).$$

Let $\mathcal{C}$ denote the set of clusters in T returned by HeavyRulingForest. Consider the cluster graph $T_{\mathcal{C}}$ with clusters from $\mathcal{C}$. Then $T_{\mathcal{C}}$ is a tree. Note that edges in $T_{\mathcal{C}}$ correspond in a unique way to edges in T and we will treat them as both. We claim that for each edge e in the cluster graph $T_{\mathcal{C}}$ there is a unique cluster $C_e \in \mathcal{C}$ such that the weight of e is smaller than $O(\omega(M_{C_e})/\log n)$ where M_{C_e} is the matching computed locally in C_e in step 3. Indeed , select a root R in $T_{\mathcal{C}}$ arbitrarily and give an orientation to edges to create a directed tree with root R so that every vertex in $T_{\mathcal{C}}$ but R has exactly one arc leaving it. Then for each cluster X from $\mathcal{C}$ different than R there is exactly one edge e_X that starts at X. By Lemma 2.5, there is a path in X of length at least $\log n$ such that each edge on this path has weight at least $\omega(e_X)/2$. Consequently, the weight of the maximum weighted matching in X is at least $\frac{\log n}{6}\omega(e_X)$. Let $\bar{T}$ denote the subgraph of T obtained by deleting all edges that connect different clusters from $\mathcal{C}$. Then WMatchinginTree finds a maximum-weight matching M in $\bar{T}$. Let $\omega^*_T(X)$ denote the weight of a maximum-weight matching in cluster X in T. We have

$$\omega^*_T \leq \sum_{X \in \mathcal{C}} [\omega^*_T(X) + \omega(e_X)] \leq \left(1 + \frac{6}{\log n}\right) \sum_{X \in \mathcal{C}} \omega^*_T(X) = \left(1 + \frac{6}{\log n}\right) \omega_T(M),$$

and so

$$\omega_T(M) \geq \left(1 - \frac{6}{\log n}\right) \omega^*_T \geq \left(1 - \frac{6}{\log n}\right) \omega^*_F.$$

Consequently,

$$\omega_F(M) \geq \omega_T(M) - \frac{\omega_{max}}{n} \geq \left(1 - \frac{6}{\log n}\right) \omega^*_F - \frac{\omega_{max}}{n} \geq \left(1 - \frac{7}{\log n}\right) \omega^*_F$$

as $\omega^*_F \geq \omega_{max}$. □

We will now turn the attention to the Minimum-Weight Dominating Set Problem. Recall that if $G = (V, E)$ is an unweighted graph then a dominating set in G is a subset $D \subseteq V$ such that for every vertex v in V either v is in D or a neighbor of v is in D. Minimum Dominating Set is a dominating set of the smallest size. We will be interested in the weighted version of the problem. Let $G = (V, E, \omega)$ where $\omega : V \to \mathcal{R}^+$. For a subset $D \subseteq V$, we set $\omega(D) := \sum_{v \in D} \omega(v)$. A minimum-weight dominating set in G is a dominating set D in (V, E) such that $\omega(D)$ is the smallest possible.

The main idea of the dominating set algorithm is similar to the maximum matching procedure. This time however the weights are defined on vertices and the first task of the procedure is to carefully define

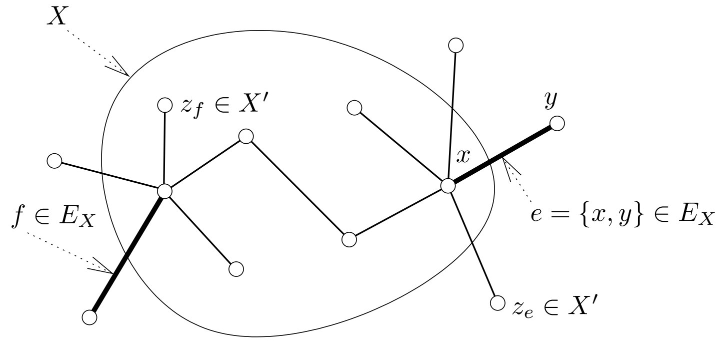

Figure 3: Modifying clusters in WDSinTree

the weights on edges. Once this is done the algorithm finds a clustering using HeavyRulingForest procedure. Vertices from a cluster C can be dominated either by vertices from C or the vertices from clusters which are connected to C in the cluster graph. Therefore we must enlarge C and include some vertices from other clusters. This again must be done carefully so that enlarged clusters contain only vertices which are "useful" in dominating C. Finally the union of all locally found dominating sets is returned. We will however need one more assumption about tree F. Let $N[v]$ denote the closed neighborhood of v, i.e. $N[v] = N(v) \cup \{v\}$, let

$$L(v) = \min_{u \in N[v]} \omega(u),$$

and

$$L = \max_{v \in V(F)} L(v).$$

We shall assume that L is globally known. Clearly $L(v)$ measures how much weight is needed to dominate vertex v and L is related to the weight of a minimum-weight dominating set. In particular, if ω^* denotes the weight of a minimum-weight dominating set then $L \leq \omega^* \leq nL$. The weights on edges are defined it terms of L, specifically the weight of $\{u, v\}$ is equal to $\max\{L(u), L(v)\}$. Note that the sum of weights of edges connecting cluster C with different cluster bounds from above the weight of vertices added when enlarging C in step 4 of procedure WDSinTree.

WDSinTree
Input: Tree F with weights $\omega : V(F) \to \mathcal{R}^+$, L where $L = \max_{v \in V(F)} L(v)$.
Output: Dominating set D.

1. For each vertex $v \in V(F)$ in parallel:
   If $\omega(v) < L/n^2$ then set $\omega(v) := L/n^2$. Otherwise do not change $\omega(v)$. Let T denote the tree with changed weights.

2. For each edge $\{u, v\}$ in T, let
   $$\bar{\omega}(u, v) := \max\{L(u), L(v)\}.$$
   Let T' denote graph with weights on edges.

3. Use HeavyRulingForest (with L as the maximum weight) to find the set of clusters $\mathcal{C}$ in T'.

4. For each cluster $X \in \mathcal{C}$, in parallel:
   Let E_X denote the set of edges in T' which have exactly one endpoint in X. For any edge $e \in E_X$ if $e = \{x, y\}$ with $x \in X$ then let z_e be a vertex in $N[x]$ with $\omega(z_e) = L(x)$. Let $X' := \bigcup_{e \in E_X} \{z_e\}$ and $\bar{X} := X \cup X'$.

5. In each cluster $\bar{X}$ find a dominating set $D(\bar{X})$ of the minimum weight.
6. Return $\bigcup_X D(\bar{X})$.

Before we analyze the procedure in more details let us immediately remark that clusters $\bar{X}$ obtained in step 3 have the diameter of at most $diam_F(X)+2$ and so the diameter of each $\bar{X}$ will be $O(\log^3 n)$ by Lemma 2.4. In addition, note that if the value of L is not known precisely but instead an upper bound U for L is known and the weights are positive integers then an easy modification of the algorithm will run in $O(\log U polylog(n))$. Indeed, simply make HeavyMatching iterate $O(\log U)$ times with U as a maximum so that all edges will be exposed in the process.

First we observe the following property of connecting edges.

Lemma 2.7 *Let $e \in E_X$. Then*

$$\bar{\omega}(e) = O\left(\frac{\omega(D(\bar{X}))}{\log n}\right).$$

Proof. Let $\bar{\omega} = \bar{\omega}(e)$. By Lemma 2.5, X contains a path $P = v_1, \ldots, v_k$ of length $\log n$ such that $\bar{\omega}(v_i, v_{i+1}) \geq \bar{\omega}/2$. By definition of $\bar{\omega}$, at least $\Omega(k)$ vertices w on the path have $L(w) \geq \bar{\omega}/2$. Let V' be the set of such vertices. As the graph is a tree a vertex can dominate at most three vertices from V'. As a result the weight of any set which dominates P is $\Omega(k\bar{\omega})$. In addition for any set S which dominates P there is a set $T \subset \bar{X}$ which dominates P and such that $\omega(T) \leq \omega(S)$. Indeed, suppose that $s \in S$ dominates a vertex v in P and s is not in $\bar{X}$. Then s is not on the path and so it dominates a unique vertex from P. Moreover there exists a vertex $w \in \bar{X}$ which is a neighbor of v and such that $\omega(w) \leq \omega(s)$. We can delete s from S and add w.

Therefore a set of minimum weight which dominates P has weight $\Omega(k\bar{\omega})$ and is contained in $\bar{X}$. Consequently $\omega(D(\bar{X})) = \Omega(k\bar{\omega}) = \Omega(\log n\bar{\omega})$. □

Lemma 2.7 immediately gives the following corollary.

Corollary 2.8 *Let $W = \sum_{X\in\mathcal{C}} \omega(D(\bar{X}))$. Then*

$$\sum \bar{\omega}(e) = O(W/\log n)$$

where the sum is taken over all edges e which connect different clusters from $\mathcal{C}$.

Our next lemma shows that the restriction of the minimum-weight dominating set D^* to a cluster X and the dominating set $D(\bar{X})$ found locally in the enlarged cluster $\bar{X}$ differ in weight which is equal to the sum of weight of edges connecting X to other clusters.

Lemma 2.9 *Let D^* be a dominating set of the minimum weight in T. Then*

$$\omega(D(\bar{X})) \leq \omega(X \cap D^*) + \sum_{e\in E_X} \bar{\omega}(e).$$

Proof. First observe that $(X \cap D^*) \cup X'$ is a dominating set in $\bar{X}$. Indeed, suppose that there is a vertex v in X such that v is dominated by $v^* \in D^*$ and v^* is not in X. Then v^* is an element of a different (than X) cluster from $\mathcal{C}$. Then however $\{v, v^*\} \in E_X$ and so, by definition of X', there is a vertex $v' \in X'$ such that v is a neighbor of v'. Since $D(\bar{X})$ is a dominating set of the minimum weight in $\bar{X}$, we have

$$\omega(D(\bar{X})) \leq \omega((X \cap D^*) \cup X') \leq \omega(X \cap D^*) + \omega(X').$$

Recall that in the third step of the algorithm we add a vertex x' to X' if $x' = z_e$ for some $e \in E_X$. As a result $\omega(x') \leq \bar{\omega}(e)$ and so

$$\omega(D(\bar{X})) \leq \omega(X \cap D^*) + \sum_{e\in E_X} \bar{\omega}(e).$$

□

Finally, we can summarize WDSinTree in the next theorem.

Theorem 2.10 *Let F be a weighted tree and let ω^* denote the weight of a minimum-weight dominating set in F. Algorithm* WDSinTree *finds in $polylog(n)$ rounds a dominating set D in F such that*

$$\omega(D) \leq (1 + O(1/\log n))\omega^*.$$

Proof. Let A be a set of vertices and let $\omega_F(A)$ ($\omega_T(A)$) denote the weight of A in F (in T). Let D_T be a dominating set of the minimum weight in T. Recall that $L = \max_{v\in V(F)} L(v)$ and observe that $L \leq \omega^*$. First note that if D_F is a dominating set of the minimum weight in F then

$$\omega_T(D_F) \leq \omega_F(D_F) + L/n = \omega^* + L/n \leq (1 + 1/n)\omega^*$$

where the first inequality follows from the fact that we increase a weight of a vertex by at most L/n^2 when creating T. Consequently, as D_F is a dominating set in T as well, we have $\omega_T(D_T) \leq \omega_T(D_F) \leq (1 + 1/n)\omega^*$.

Let $W = \sum_{X\in\mathcal{C}} \omega_T(D(\bar{X}))$. By Lemma 2.9, we have

$$W \leq \sum_{X\in\mathcal{C}} \left[\omega_T(X \cap D_T) + \sum_{e\in E_X} \bar{\omega}(e)\right] = \omega_T(D_T) + 2 \sum_{e\in\bigcup_X E_X} \bar{\omega}(e)$$

which by Corollary 2.8 is at most

$$\omega_T(D_T) + O(W/\log n).$$

As a result $W \leq (1 + O(1/\log n))\omega_T(D_T)$. Finally, as $D = \bigcup_{X\in\mathcal{C}} D(\bar{X})$

$$\omega(D) = \omega_F(D) \leq \omega_T(D) \leq W \leq \left(1 + \frac{O(1)}{\log n}\right)\omega_T(D_T) \leq$$

$$\leq \left(1 + \frac{O(1)}{\log n}\right)\left(1 + \frac{1}{n}\right)\omega^* \leq \left(1 + \frac{O(1)}{\log n}\right)\omega^*.$$

□

3 Planar graphs

We will present an algorithm which approximates maximum-weight independent set in planar graphs. The algorithm is again based on a clustering procedure. We first discuss some preliminary facts in Section 3.1 and then in Section 3.2 we first give procedure Clustering which finds a set of clusters in a planar graph and then present WISinPlanar which finds an almost exact approximation of a maximum-weight independent set in planar graphs.

3.1 Preliminaries

Our algorithm again uses an auxiliary clustering procedure. This time however, the clustering is less powerful than the clustering from previous section (in particular it does not give a lower bound for the diameter of each cluster) and it is much less clear which problems can be attacked using this method. In fact, at the moment, we had success only with applying it to the maximum-weight independent set problem.

We first introduce necessary notation and state some simple auxiliary facts. Let $G = (V, E, \omega)$ be a graph with weight function $\omega : V \to \mathcal{R}^+$. For a subset $A \subseteq V$, let $\omega(A) = \sum_{v\in A} \omega(v)$. Similarly, if $G = (V, E, \omega)$ is a graph with weight function $\omega : E \to \mathcal{R}^+$ then for $F \subseteq E$, $\omega(F) = \sum_{e\in F} \omega(e)$. We will need the following easy fact about distribution of degrees in a planar graph.

Lemma 3.1 *Let G be a planar graph on n vertices and let $A = \{v \in V(G) | deg(v) \leq 6\}$. Then $|A| > n/6$.*

Proof. Note that adding edges to G can only decrease $|A|$ and so we can assume that G does not have isolated vertices. Suppose that $|A| \leq n/6$. Then

$$\sum_{v \in V} deg(v) \geq 7(n - |A|) + |A| \geq 6n$$

which contradicts the fact that $|E| \leq 3n - 6$. □

A low-degree decomposition of a planar graph $G = (V, E)$ is a partition of V into K independent sets $V_1, \ldots, V_K$ which satisfies the following conditions:

1. $K = O(\log |V|)$.
2. For every $i = 1, \ldots, K-1$, if $v \in V_i$ then v has at most six neighbors in $\bigcup_{l=i+1}^{K} V_l$.

Clearly the existence of such a decomposition of a planar graph follows immediately from Lemma 3.1. In addition, a low-degree decomposition can be found by a distributed algorithm.

```
DECOMPOSITION
Input: Planar graph $G$ on $n$ vertices.
Output: Low-degree decomposition $V_1, \ldots, V_K$ of $G$.

  1. For $i := 1$ to $O(\log n)$ do:
      (a) Let $A$ be the set of vertices in $G$ of degree at most 6.
      (b) Use the Cole-Vishkin algorithm from [CV86] to find a maximal independent set $I$ in the graph
          induced by $A$.
      (c) $V_i := I$.
      (d) Delete all vertices in $I$ from $G$.
```

We will invoke DECOMPOSITION repeatedly in our main algorithm.

3.2 Algorithm

We will now present the algorithm for approximating maximum-weight independent set in planar graphs. The algorithm is divided into three procedures: TRANSFORMATION, CLUSTERING, and WISINPLANAR. The first procedure takes a planar graph G with weights on vertices and finds an edge-weight function to obtain $\bar{G}$. The second procedure finds clusters in the modified graph $\bar{G}$. Clusters have the property that the total weight of edges between different clusters is small and the diameter of each cluster is polylogarithmic. Finally the third procedure finds the maximum-weight independent set in each cluster, takes the union of them, and deletes some vertices to obtain an independent set in graph G.

```
TRANSFORMATION
Input: Planar graph $G = (V, E, \omega)$ with weight function $\omega : V \to \mathcal{R}^+$.
Output: Planar graph $\bar{G} = (V, E, \bar{\omega})$ with weight function $\bar{\omega} : E \to \mathcal{R}^+$.

  1. Use DECOMPOSITION to find $V_1, \ldots, V_K$.
  2. For each edge $e$ in parallel if $e$ connects a vertex $v$ in $V_i$ with $w$ in $V_j$ where $i < j$ then $\bar{\omega}(e) := \omega(v)$.
```

The weight function $\bar{\omega}$ has the following property.

Lemma 3.2 *Let $\bar{\omega} : E \to \mathcal{R}^+$ be the function obtained by* Transformation*. Then*

$$\sum_{e \in E} \bar{\omega}(e) \leq 6 \sum_{v \in V} \omega(v).$$

Proof. Since $V_1, \ldots, V_K$ is a low-degree decomposition, for every vertex v if $v \in V_i$ then v has at most six neighbors in $\bigcup_{l \geq i+1} V_l$. Therefore there are at most six edges in $\bar{G}$ with weight equal to $\omega(v)$ that correspond to v. $\square$

Next procedure finds clusters in a planar graph and is the core of our algorithm. The procedure proceeds as follows. In the first iteration, a low-degree decomposition $W_1, \ldots, W_K$ of a planar graph $\bar{G}$ is found. Then every vertex $v \in W_i$ examines all edges incident to vertices in $\bigcup_{l>i} W_l$ (there are at most six such edges) and selects the one of a maximum weight. Now consider the subgraph F of $\bar{G}$ consisting of selected edges. Every vertex $v \in W_i$ has at most one neighbor in $\bigcup_{l>i} W_l$ when restricted to F and no neighbors in W_i (as W_i is an independent set). Thus F is a forest. Each tree in F has diameter of $O(K) = O(\log n)$ in G and so every tree in F can perform computations locally. In the next step, each tree T in F finds a subset of stars of T with the maximum weight. Note that each tree can do all the computations locally and so this can be done in $O(\log n)$ steps. Then each star is contracted to a new vertex. Finally these new vertices and the non-contracted old vertices are considered in a new graph H. Graph H is also planar, and the above steps are repeated in H. After $O(\log \log n)$ iterations vertices of a resulting graph H correspond to subsets of vertices of $V(\bar{G})$ which induce connected subgraphs of $\bar{G}$. These subsets are our clusters in $\bar{G}$.

Clustering
Input: Planar graph $\bar{G} = (V, E, \bar{\omega})$ with weight function $\bar{\omega} : E \to \mathcal{R}^+$ and $n = |V|$.
Output: Partition of V into L sets $V_1, \ldots, V_L$.

1. $H = \bar{G}$

2. Iterate $\log \log n / \log \frac{12}{11}$ times:

   (a) Call Decomposition to find a partition $W_1, \ldots, W_K$ of H with $K = O(\log n)$. Let $W_{K+1} := \emptyset$. In addition let $Z_i := \bigcup_{l>i} W_l$.

   (b) For every vertex w in parallel:

   (c) If $w \in W_i$ and $N(w) \cap Z_i \neq \emptyset$ then:

   - Let $u(w)$ be a vertex in $N(w) \cap Z_i$ such that

   $$\bar{\omega}(\{w, u(w)\}) := \max_{v \in N(w) \cap Z_i} \bar{\omega}(\{w, v\}).$$

   - Add $\{w, u(w)\}$ to the auxiliary graph F.

   (d) Each connected component of F is a tree of diameter $O(K) = O(\log n)$. For each tree T in F, in parallel, find a set of disjoint stars $S_1 \ldots S_k$ in T of the maximum weight.

   (e) Modify H as follows:

   - In each star, contract vertices to create a new vertex. Let $V(H)$ consist of new vertices and those vertices which were not contracted.
   - For each new vertex v and $w \in V(H)$ set the weight of $\{v, w\}$ to be the sum of weights of edges between vertices contracted to v and vertices contracted to w.

3. Let $V(H) = \{v_1, ..., v_L\}$. For each v_i add to the clusters the set V_i which contains all vertices contracted to v_i in the above iterations.

First let us note that after contractions the graph remains planar. As a result, the graph H is a planar graph in all iterations of Clustering. We need three simple facts about the clusters $V_1, V_2, \ldots, V_L$ obtained by Clustering.

Lemma 3.3 *Let $V_1, V_2, \ldots, V_L$ be the clusters obtained by* Clustering *in $\bar{G}$. Then each V_i induces a graph in $\bar{G}$ of diameter $O(\log^d n)$ where $d = \log 3/\log \frac{12}{11}$.*

Proof. Let $d_i(v)$ be the diameter of the subgraph of $\bar{G}$ induced by vertices which have been contracted to v in iterations $1, \ldots, i$ and let d_i be the maximum of $d_i(v)$. Then $d_i \leq 3d_{i-1} + 2$ and $d_0 = 1$ as a vertex v in the graph from the ith iteration either corresponds to a single vertex in the graph from the iteration $i-1$ or corresponds to a star of vertices in the iteration $i-1$. In the first case $d_i(v) = d_{i-1}(v)$ and in the second $d_i(v) \leq 3d_{i-1} + 2$. One can easily solve the recursive inequality to obtain $d_i \leq 2 \cdot 3^i$ and for $k = \log\log n/\log \frac{12}{11}$, $d_k = O(\log^d n)$. □

Lemma 3.4 *Let $V_1, V_2, \ldots, V_L$ be the clusters obtained by* Clustering *in $\bar{G}$. Let P denote the sum of weights of edges in $\bar{G}$ and let p be the sum of weights of edges between different clusters. Then $p = O(P/\log n)$.*

Proof. Let P_i be the sum of weights of edges of H in the ith iteration of Clustering. First consider the forest F obtained in 2(d). For each $j = 1, \ldots, K$ any vertex $w \in W_j$ has at most six neighbors in Z_j. In step 2(c), w selects one edge (of the maximum weight) out of at most six and adds it to F. Therefore the sum of weights of edges in F is at least $P_i/6$. In step 2(d), each tree T in F selects a set of disjoint stars $S_1, \ldots, S_k$ of the maximum weight. By considering an arbitrary root r in T and vertices which are at an odd distance to r versus vertices which are at an even distance to r, the set of stars $S_1, \ldots, S_k$ satisfies

$$\sum_{i=1}^{k} \sum_{e \in E(S_i)} \bar{\omega}(e) \geq \frac{1}{2} \sum_{e \in E(T)} \bar{\omega}(e).$$

Consequently, the sum of weights of edges in all stars is at least $P_i/12$. Since all of these stars are contracted, we have $P_{i+1} \leq 11/12 P_i$. Therefore for $k = \log\log n/\log \frac{12}{11}$, $P_k = O(P/\log n)$. □

Lemma 3.5 *Let $d = \log 3/\log \frac{12}{11}$. Procedure* Clustering *finds clusters $V_1, V_2, \ldots, V_L$ in $O(\log\log n \log^* n \log^{d+1} n)$ rounds.*

Proof. First note that by Lemma 3.3, the diameter of each cluster is $O(\log^d n)$. We have $O(\log\log n)$ iterations of step 2. In each iteration we invoke Decomposition in H. Since vertices of H are clusters in $\bar{G}$ of diameter $O(\log^d n)$ and the Cole-Vishkin algorithm needs $\log^* n$ rounds, the number of rounds used to find the decomposition $W_1, \ldots, W_K$ in H is $O(\log^* n \log^{d+1} n)$. Finally, each tree in F has diameter $O(\log n)$ in H and so the diameter in G is $O(\log^{d+1} n)$. Thus the set of stars $S_1, \ldots, S_k$ can be found in $O(\log^{d+1} n)$ rounds. □

In the last procedure we will find an independent set of large weight.

WISinPlanar
Input: Planar graph $G = (V, E, \omega)$ with weight function $\omega : V \to \mathcal{R}^+$ and $n = |V|$.
Output: Independent set I.

1. Call Transformation to obtain the weighted graph $\bar{G}$.
2. Call Clustering to find the set of clusters $V_1, \ldots, V_L$ in $\bar{G}$.
3. In each cluster V_j, in parallel, find an independent set I_j of the maximum weight.
4. $I := \bigcup_{j=1}^{L} I_j$.

5. For every edge $e = \{u, v\}$ which connects different clusters, if $u \in I$ and $v \in I$ then delete u whenever $\omega(u) < \omega(v)$ and delete v otherwise.

Theorem 3.6 *Let $G = (V, E, \omega)$ be a planar graph on n vertices with a weight function $\omega : V \to \mathcal{R}^+$, let ω^* be the weight of a maximum-weight independent set in G, and let $d = \log 3/\log \frac{12}{11}$. Procedure* WISinPlanar *finds in $O(\log\log n \log^* n \log^{d+1} n)$ rounds an independent set I such that*

$$\omega(I) \geq (1 - O(1/\log n))\,\omega^*.$$

Proof. Let $Q = \sum_{v\in V} \omega(v)$. We have $\omega^* \geq Q/4$ as G can be colored using four colors. Let I be the set from step 4 and let I' denote the independent set obtained from I by a "correction" in step 5. First observe that $\omega(I) \geq \omega^*$ and our goal is to show that the weight lost in the correction step is small. To that end, first note that by Lemma 3.2 graph $\bar{G}$ obtained in step 1 is such that

$$\sum_{e\in E} \bar{\omega}(e) \leq 6 \sum_{v\in V} \omega(v).$$

Let $P = \sum_{e\in E} \bar{\omega}(e)$. By Lemma 3.4, the total weight of edges between different clusters is $O(P/\log n)$. In addition if $e = \{u, v\}$ then $\bar{\omega}(e) \geq \min\{\omega(u), \omega(v)\}$. Consequently, the weight of I' satisfies

$$\omega(I') \geq \omega(I) - O(P/\log n) \geq \omega(I) - O(Q/\log n) \geq$$

$$\omega^* - O(\omega^*/\log n) = (1 - O(1/\log n))\,\omega^*.$$

□

References

[AGLP89] B. Awerbuch, A. V. Goldberg, M. Luby, S. A. Plotkin, Network Decomposition and Locality in Distributed Computation, Proc. 30th IEEE Symp. on Foundations of Computer Science , 1989, pp. 364-369.

[CV86] R. Cole, U. Vishkin, Deterministic coin tossing with applications to optimal parallel list ranking, Information and Control, 1986, 70, pp. 32-53.

[CH03] A. Czygrinow, M. Hańćkowiak, Distributed Algorithm for Better Approximation of the Maximum Matching, COCOON 2003 LNCS 2697 , 2003, pp. 242-251.

[CHS04] A. Czygrinow, M. Hańćkowiak, E. Szymańska, Distributed algorithm for approximating the maximum matching, Discrete Applied Mathematics, Volume 143, Issues 1-3, (2004), pp. 62–71.

[CHSz04] A. Czygrinow, M. Hańćkowiak, E. Szymańska, A fast distributed algorithm for approximating the maximum matching, Algorithms - ESA 2004 LNCS 3221, (2004), pp. 252–263.

[E04] M. Elkin, Distributed Approximation - A Survey, ACM SIGACT News, 35(4), pp. 3-18.

[JRS02] L. Jia. R. Rajaramam, T. Suel, An efficient distributed algorithm for constructing small dominating sets, Distributed Computing, 2002, 15, pp. 193-205.

[KW03] F. Kuhn, R. Wattenhofer, Constant-time distributed dominating set approximation. Proc. of the 22nd ACM Symp. on Principles of Distributed Computing, 2003, pp. 25-32.

[KP95] S. Kutten, D. Peleg, Fast distributed construction of k-dominating sets and applications, Proceedings of the fourteenth annual ACM Symposium on Principles of distributed computing, 1995, pp. 238 - 251.

[L92] N. Linial, Locality in distributed graph algorithms, SIAM Journal on Computing, 1992, 21(1), pp. 193-201.

[LS93] N. Linial, M. Saks, Decomposing Graphs into Regions of Small Diameter Proceedings of SODA 91, the Second Annual AC-SIAM Symposium on Discrete Algorithms, San Francisco, pp. 320-330.

[WW04] M. Wattenhofer, R. Wattenhofer, Distributed Weighted Matching, DISC, 2004, Distributed Computing, 18th International Conference, Amsterdam, pp. 335-348.