Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

MINIMUM SPANNING TREES FOR MINOR-CLOSED GRAPH CLASSES IN PARALLEL

by

JENS GUSTEDT

No. 566/1997

Minimum Spanning Trees for Minor-Closed Graph Classes in Parallel*

Jens Gustedt

TU Berlin, Sekr. MA 6-1, D-10623 Berlin – Germany
gustedt@math.TU-Berlin.DE

Abstract. For each minor-closed graph class we show that a simple variant of Borůvka's algorithm computes a MST for any input graph belonging to that class with linear costs. Among minor-closed graph classes are *e.g* planar graphs, graphs of bounded genus, partial k-trees for fixed k, and linkless or knotless embedable graphs. The algorithm can be implemented on a CRCW PRAM to run in logarithmic time with a work load that is linear in the size of the graph. We develop a new technique to find multiple edges in such a graph that might have applications in other parallel reduction algorithms as well.

Keywords: graph algorithms, parallel algorithms, minimum spanning tree, graph minors

1 Introduction and Overview

Minor-closed classes of graphs, *i.e* classes of graphs that are stable under contraction of edges and deletion of vertices and edges, have found a lot of attention in recent years because they cover the topological intuition behind many definitions of graph classes quite well and because of the stimulating work on structure and algorithms for these classes by Robertson and Seymour, see the book of Diestel (1997) for an introduction. Examples for minor-closed graph classes are:

- **Planar graphs**.
- **Graphs of bounded genus** g, *i.e* graphs that can be embedded without crossing edges into some surface of genus g.
- **Partial k-trees**, *i.e* graphs that are intersection graphs of a collection C of subtrees of a tree T such that each vertex v of T is found in at most $k+1$ subtrees in C.
- **Linkless embedable graphs**, *i.e* graphs that can be embedded into 3-space such that no two cycles of the graph are linked.
- **Knotless embedable graphs**, *i.e* graphs that can be embedded into 3-space such that no cycle of the graph forms a non-trivial knot under the embedding.

A **minimum spanning tree** (forest), *MST* for short, in an edge-weighted graph is a tree (forest) that spans (all components of) the graph and whose sum of edge weights is minimal among all spanning trees (forests). Minimum spanning trees and forests are among the basic tools in graph algorithms: it is a classical problem to compute such a

* This is a slightly updated version as it has been accepted for STACS'98.

spanning tree in a graph in the most efficient way. The currently best known deterministic sequential algorithm runs in $O(m\alpha\log\alpha)$, Chazelle (1997), the best randomized algorithm has expected linear time, Karger et al. (1995).

At a first look it might be surprising that such an efficient computation can be done in almost linear time, even if only the comparison model on the edge weights is assumed. The book of Tarjan (1983) gives a good overview about the classical approaches to this problem. It classifies the different approaches into how they apply to specific rules to some edge e, which in the terminology used in this article look as follows:

blue rule: If e is of minimum weight in some cut it is in the *MST* and is contracted.
red rule: If e is of maximum weight on some cycle it is not in the *MST* and is deleted.

In general arbitrary cuts and cycles are not so easy to maintain, so to be efficient we restrict ourselves to the following two rules:

ultraviolet rule: If e is of minimum weight for one endpoint it is in the *MST* and is contracted.
infrared rule: If e has a sibling e' with the same endpoints and a higher weight than e' it doesn't belong to the *MST* and is deleted.

Applying the ultraviolet rule simultaneously on as many edges as possible is often called a **Borůvka step**, see Borůvka (1926). It is easy to see that the number of vertices halves in each such step, so after a logarithmic number of steps the graph is contracted to a single vertex and the *MST* is known. This general scheme has been widely used to compute the *MST* efficiently and forming the base to linear and almost linear time and work algorithms as well as in sequential, Yao (1975), Cheriton and Tarjan (1976), Karger et al. (1995), and parallel, Halperin and Zwick (1996), Poon and Ramachandran (1997). Linear time resp. work algorithms when there is no restriction on the input graphs are only known in a *randomized* setting. The main obstacle in the deterministic case being that after applying a certain number of Borůvka steps the graph gets more and more dense. To reduce the number of edges as well as the number of vertices usually an efficient variant of the red rule is applied, see Dixon et al. (1992), King (1997). This leads to linear expected time or work algorithms, but in a deterministic setting some involved data structures are required that introduce some sub-logarithmic factor.

The picture changes a bit when the input graphs are restricted to some graph classes that guarantee sparseness: Cheriton and Tarjan (1976) also give a deterministic linear time algorithm for *planar* graphs that generalizes to –in todays vocabulary– minor-closed graph classes, see Tarjan (1983). The later fact doesn't seem to be too well known in the community, we will discuss an approach that achieves this below.

The key observation here is that simple graphs in such a minor-closed graph class are sparse. So all that an algorithm must do after a Borůvka step is to eliminate duplicate edges, *i.e* to apply the ultraviolet rule to all possible edges. In a sequential setting this is relatively simple, sorting the edges with a two-fold bucketsort easily finds duplicates. But in parallel, no efficient equivalent to bucketsort is known. The aim of this paper is to circumvent this problem and to show how at least a substantial fraction of the duplicate edges can be found. By that we are able to prove the following theorem.

Main Theorem. *On any non-trivial minor closed graph class C the MST problem can be solved with a linear work load and a time of $O(\log n\log^* n)$ on a EREW PRAM or a time of $O(\log n)$ on a CRCW PRAM.*

This paper is organized as follows. The next section provides the necessary notations and facts for graphs in general, minor-closed graph classes and *MST*s. Section 3 then revisits the optimal sequential algorithm and proves the linear time bound for all minor-closed graph classes. Section 4 develops the approach that helps to deal with multiple edges. In particular it shows that if the graph has a certain amount of edges per vertex, for high degree vertices the edge lists always contains duplicate edges that are only of constant distance in the list. Section 5 then plugs all the tools together into the final PRAM algorithm.

2 Graph-Theoretic Background

In general, graphs in this paper are finite undirected multi-graphs with loops, *i.e* may contain several edges between the same one or two-element set of vertices. If e and e' are edges with the same endpoints they are called **siblings** of each other. If a graph has no loops or multiple edges it is called **simple**. If G is a multigraph its **simplification** G^0 is the graph obtained by deleting loops and multiple copies of edges. For a graph G we will denote the number of vertices by n_G and the number of edges by m_G. A graph is **connected** if for any partition V_1, V_2 of the vertex set there is an edge e with one endpoint in V_1 and the other in V_2. A connected graph is a **tree** if the removal of any edge disconnects it. Clearly that a tree T has no loops and multiple edges and that m_T is always equal to $n_T - 1$.

Let G be a graph and $v \in V_G$ and $e \in E_G$ be a vertex resp. edge of the graph. We will use the following standard notations for certain graphs obtained from G:

$G \setminus \{e\}$, a **subgraph**, obtained by deleting e from E_G.
$G \setminus \{v\}$, an **induced subgraph**, obtained by deleting v and its incident edges.
$G/\{e\}$, a **contraction**, obtained by identifying the endpoints of e.

Observe, that contraction of an edge may create multiple edges between vertices or loops at a vertex. Since the result of a sequence consisting of operations of one type only does not depend on the order in which these operations are performed, we extend the above notation to sets of vertices and edges.

A subgraph H of G is a **reduction** of G if $V_H = V_G$. A reduction H is **spanning** if it has the same number of components as G and for all $v \in V_G$ there is an edge $e \in E_H$ such that $v \in e$. It is a **spanning tree** if it is a tree and spanning.

A graph H obtained from another graph G by a sequence of the above operations is called a **minor** of G. We denote this relation among graphs by $H \preceq G$. Observe that $\preceq$ is a transitive relation. A property P of graphs is called **minor-closed** if for any pair of graphs with $H \preceq G$ and such that P holds for G, P also holds for H.

If $H \preceq G$ the vertices $V(H)$ correspond to pairwise disjoint subgraphs of G. If not stated otherwise, we keep loops and multiple copies of edges that may appear after contractions. So if we are explicitly given given a series of contractions and deletion, for any edge $e \in E_H$ we may uniquely identify a pre-image $e' \in E_G$ under these operations. Usually we will not distinguish e and its corresponding pre-image e'.

2.1 Minors and Minimum Spanning Trees

In the rest of the paper all graphs will be edge weighted, *i.e* a graph G is considered as being a triple (V_G, E_G, ω_G) where $\omega_G : E_G \to R$, the **weight function**. The weight $\omega_G(H)$ of a minor $H \preceq G$ is just the sum $\sum_{e \in E_H} \omega_G(e)$ where the edges of H are identified with edges of G as stated above. For convenience we assume that all weights of edges are pairwise distinct. This can be achieved by adding small values on the weights that differ for each edge, $a\varepsilon$ where a is the unique "number" of the edge and $\varepsilon > 0$ is a value that is suitable small. Given a weighted connected graph G, the **minimum spanning tree**, *MST*, problem asks for a spanning tree T of minimum weight over all spanning trees. It is well known that under the uniqueness condition on ω_G the minimum spanning tree is unique. We denote it by T_G.

If G is not connected the *minimum spanning forest* of G is the union of the minimum spanning trees of its components. In general the distinction between looking for a spanning forest versus a spanning tree will not be in the focus of our attention. We will denote such a spanning forest also with the symbol T_G.

The applications of the ultraviolet and infrared rules as presented in the introduction are essential for our discussion. It is already known in the early literature on *MST*, see Borůvka (1926): contracting an edge of the minimum spanning tree and deleting multiple copies of edges doesn't influence the rest of the spanning tree.

2.2 Minor Closed Graph Classes are Sparse

The following theorem is a crucial prerequisite for the discussion in this paper.

Theorem 1 (Mader (1967)). *There is a function h such that for every $r > 0$ every simple graph G with average degree at least $h(r)$ contains K_r as a (topological) minor.*

A *topological minor* is a certain restriction of the minor relation, which by itself is not important for our discussion here. We use this theorem in the form of following corollary which basically states that minor-closed graph classes are sparse.

Corollary 2. *For any non-trivial minor closed graph class C there is a value μ_C such that all simple graphs in C have average degree at most μ_C.*

Proof. Since C is non-trivial there is some graph $H \notin C$. Choose r in Thm. 1 as being n_H and $\mu_C = h(r)$. Then every simple graph G of average degree more than μ_C has K_r as a minor which in turn has H as a subgraph. So $H \preceq G$ and so it can't belong to C. □

In the following we will assume that for any minor-closed graph class C a threshold μ_C as in Cor. 2 is given and fixed. For convenience we will also assume that $\mu_C \geq 1$.

Corollary 3. *For any non-trivial minor closed graph class C and for any simple graph G in C there are at least $n_G/2$ vertices that have degree at most $2\mu_C$.*

Proof. Suppose the contrary, *i.e* there are more than $n_G/2$ vertices of degree $> 2\mu_C$. Then the average degree is larger than $2\mu_C n_G/(2n_G) = \mu_C$, a contradiction to Cor. 2. □

In fact it may seem annoying that only existence of the constant μ_C is known for a particular class C. But whenever we know some obstruction, *i.e* a graph that is not in the class, we may estimate μ_C with the following theorem.

Theorem 4 (Kostochka (1982), Thomason (1984)). *There is a $c > 0$ such that, for every positive integer r, every graph G of average degree $d(G) \geq cr\sqrt{\log r}$ has a K_r minor. Up to the value of c, this bound is best possible as a function of r.*

3 An Optimal Sequential Algorithm

Now we are ready to formulate our prototype of an algorithms that will solve the *MST* problem for any minor-closed graph class. It is rather a prototype than a complete algorithm, since it depends on a certain parameter u, and it leaves the parts that depend on the machine model (*i.e* sequential or parallel) open.

Theorem 5. *Let C be any fixed minor-closed graph class and $u = \mu_C$ the degree threshold for C as given above. Then, given as input a member G of C with weight function ω_G, Alg. 1 with parameter u, MST^u, can be implemented on a sequential machine such that it outputs the MST of G in linear time.*

Proof. Correctness follows immediately from the validity of the ultraviolet and infrared rules. For the complexity it suffices to show, that each performance of the while-loop needs linear time in the number of edges m of the graph, and that this number is reduced by a factor $0 < \varepsilon_u < 1$ in each round. The total time then is $O(\frac{1}{1-\varepsilon_u}m) = O(m)$.

To see that the time for each round is linear, observe first that line 1 poses no problem at all and that 2 can be easily done if we assume that the edges are sorted. To re-insure this invariant after every round we just have to rename all vertices with consecutive values, propagate these values to the edges, and to apply a two-fold bucketsort to newly order the edge lists. Because the remaining statements basically perform operations on vertices of bounded degree, in total they also do need no more than linear time.

After line 2 the graph has no more than $m \leq \mu_C n/2 = un/2$ edges (Cor. 2). Because of Cor. 3 we know that $|L| \geq n/2$. Every edge might be chosen at most twice, so at

Algorithm 1: The Prototype of Algorithms *MST* for Parameter u, MST^u.

Input: A graph G with edge weight ω_G.
Output: A minimum spanning tree T.
while *G is not trivial* **do**
1 Delete loops;
2 Replace duplicate edges by the one with the least weight;
3 Find the set L of all vertices of degree less than $2u$;
foreach $v \in L$ **do**
4 Find the edge e_v of v with least weight;
5 Add e_v to T;
6 Contract e_v;

least $n/4$ edges are contracted in line 6. So we reduced their number by a factor of $\varepsilon_u \leq (m-n/4)/m \leq (m-m/2u)/m = 1-1/2u$. □

Observe that the dependency on $u = \mu_C$ in the running time is $\frac{1}{1-\varepsilon_u} = 2u$. This is so since the running time in the sequential setting of each round does not depend on u. *E.g* finding the minimum weight edge in line 4 only gives a total work that is linear in $m_G + n_G$.

4 High Degree Vertices: A Challenge for Parallelism

When heading to parallelize Alg. 1, most other parts but line 2 can be treated with some standard tricks. Line 2, *i.e* the application of the infrared rule, makes problems, since no way to achieve the linear work load of bucket sort in a parallel setting is known up to now.

So for our *MST* problem finding duplicates of edge is the real problem. Low degree vertices can be detected with a time of $O(2u) = O(1)$ each, and whether their edge lists have duplicates can be checked in $O(u \log u) = O(1)$. So for them we can circumvent the problem by raising the constants for the work to be done.

The problem on the high degree vertices is much harder. Sorting their edges would introduce a log-factor (or similar, depending on the model of parallelism) into the overall work that is performed.

Our main idea here, is not to find all duplicate edge in one round, but to find sufficiently many, such that the size of the graph is reduced substantially. Our method owes a lot to Bodlaender and de Fluiter (1997) and de Fluiter (1997) where a parallel recognition algorithm for partial 2-trees is given. Their idea was to let each edge look ahead a constant number of edges in the edge list, whether it finds a sibling of itself. The argument then uses an estimation of a certain kind of degree in the graphs in question. This estimation in turn is done by arguing with a potential tree-decomposition of the graph.

To be able to estimate the number of matches here, we can not rely on something like a tree-decomposition. Instead, we generalize the idea of Cor. 3, namely we control the number of vertices in a *simple* graph that have low degree.

4.1 Grades in Simple Graphs

For a simple graph G of class C we say that a vertex v has **grade** 0, denoted by $g_G(v) = 0$, if it has degree at most $2\mu_C$ and that it is of **higher grade** otherwise. The subgraph G^+ of G is the subgraph of G induced by the vertices of higher grade. If we iterate this process of deleting the low degree vertices we get a kind of *shelling* of our graph G. A vertex v is of higher grade $g(v)$ is recursively defined as $g(v) = g_{G^+}(v) + 1$. The **grade** of G, denoted by $g(G)$, is the maximum grade over all its vertices. So $g(G)$ reflects the number of levels in the shelling of G.

The following two observations follow easily from Cor. 3.

Remark 6. Let G be a simple graph of class C.
(a) $g(G) \leq \log n_G$.

(b) Let $k \le g(G)$. Then there are at most $2^{-k} n_G$ vertices in G of grade $\ge k$.

The grade $g_G(e)$ of an edge e is the minimum grade of its endpoints. It has **higher grade** if $g_G(e) > 0$. So the grade of an edge reflects the level in the shelling of G for which one or both of its endpoints are of low degree. As an easy consequence of Cor. 3 we get:

Remark 7. For a simple graph $G \in C$ there are at most $n_G \mu_C / 2$ edges of higher grade.

Proof. Indeed, $n_{G^+} \le n_G/2$ and since $G^+ \in C$ we get $m_{G^+} \le 2\mu_C n_{G^+} \le \mu_C n_G$. □

It follows directly from the definition that a vertex v is never incident to an edge e such that $g(e) > g(v)$. We say that an edge e is **accounted** to one of its endpoints v if $g(e) = g(v)$. The **grade-degree** $d_G^+(v)$ of a vertex v is the amount of edges e that are accounted to it. The following easy observation is crucial for our discussion.

Remark 8. Let G be a simple graph in class C. Then $d^+(v) \le 2\mu_C$ for all $v \in V_G$.

4.2 Bad Edges in a Multigraph

We switch back to *multigraphs*. For such a graph G in class C the values and terms **grade, grade-degree, accounted...** of a vertex (edge, the graph itself...) are the values and terms of the corresponding objects in the simplification G^0.

Let us suppose that a graph G of class C is given in the conventional data structure as cyclic list of edges for each vertex. For what follows we assume that one out of many possible representations of G in such a data structure is given and fixed. We also assume that a "magic" constant $\zeta > 2$ is fixed for the moment. A concrete value for it will be chosen later. Let e be an edge that is accounted to v. We call e **bad** for v if it has some sibling, but it has no sibling in the edge list of v that is closer than distance $\zeta \cdot \mu_C$ in the list. e is bad if it is bad for one of its endpoints.

Lemma 9. *Let C be a non-trivial minor-closed graph class and $G \in C$ be non-trivial given as above. Then G has at most $2m_G/\zeta$ bad edges.*

Proof. For any vertex v there are at most $2\mu_C$ different neighbors w such that v and w can be the endpoints of a bad edge (Rem. 8). Since bad edges that have the same w as other endpoint must have distance at least $\zeta\mu_C$ in the list of v there are at most $d(v)/\zeta\mu_C$ such bad edges incident to v and w. So in total there are at most $2\mu_C d(v)/\zeta\mu_C = 2d(v)/\zeta$ bad edges incident to v. The claim follows by summation over all vertices. □

Call an edge **good for contraction** if it has grade 0 and is the one with lowest weight among all edge incident to the same vertex v of grade 0. Call an edge **good for deletion** if has a sibling and is not bad. Call an edge **good** if it is so for deletion or contraction and call it **indifferent** if it is neither good nor bad. Observe that an edge can be only indifferent if it is has no sibling. The following is an easy consequence of Cor. 7.

Remark 10. There are at most $n_G \mu_C$ indifferent edges in G.

The main result in this section is the following:

Theorem 11. *Let C be a non-trivial minor-closed graph class and $G \in C$ be non-trivial given as above. Then G has at least $m_G/8\mu_C$ good edges.*

Proof. We distinguish two cases:

$m_G \leq 2\mu_C n_G$**:** In this case we know by a little variation of Cor. 3 that we always find $n/4 \geq m/8u$ edges that are good for contraction.

$m_G > 2\mu_C n_G$**:** If we plug all our estimations together we see that we have at least the following number of good edges

$$m_G - 2m_G/\zeta - n_G\mu_C > m_G - 2m_G/\zeta - m_G/2 = m_G((\zeta-4)/2\zeta) \qquad (1)$$

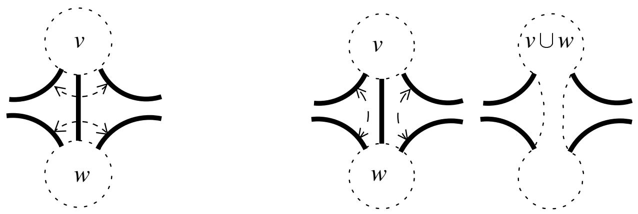
So by choosing $\zeta = \left\lceil \frac{16\mu_C}{4\mu_C-1} \right\rceil$, a value between 4 and 6, we easily get the claim. □

5 Designing a PRAM Algorithm

To build an efficient PRAM algorithm we will use the technique of Bodlaender and Hagerup (1995) for *parallel reduction algorithms*. In fact the **reduction rules** that are of interest for us are among the most simple ones – we only have the infrared and the ultraviolet rules, *i.e* deletions and contractions of good edges.

The main idea for parallel reduction algorithms is to choose a non-interfering set of graph reductions that still is at least of size εn. This can be done by observing, that any of the above reduction rules, *i.e* good edges, can only interfere with a constant number of such other rules. If we consider the cyclic order of edges as being fixed, good edges have at most 4 neighbors in the edge lists of their endpoints.

We need some further technical definitions. Two deletable edges e and f are **interfering** if they share a common end point v and they are neighbors in the circular edge list of v. Clearly that every such edge may interfere with at most 4 other edges.



(a) Deletion (b) Contraction

Fig. 1. Applying Rules Locally

Unfortunately for contractable edges we need a definition that is slightly more complicated. Two contractable edges e and f are **interfering** if they share a common end point v and

Algorithm 2: The Algorithms *MST* for Parameter u on a PRAM.

```
    Input: A graph G with edge weight ω_G.
    Output: A minimum spanning tree T.
    while G is not trivial do
1       Delete loops;
2       Find a non-interfering set of edges good for deletion and eliminate these;
3       Find the set L of all vertices of degree less than 2u;
        foreach v ∈ L do
4           Find the edge e_v of v with least weight;
+4      Find a non-interfering set L′ of contractable edges;
        foreach e_v ∈ L′ do
5           Add e_v to T;
6           Contract e_v;
+6      Update the sets of endpoints for all edges;
```

- v is a low degree vertex, or
- v is a high degree vertex and they are neighbors in the circular edge list of v.

Observe that if we have a set S of good edges that are pairwise non-interfering, we may perform the necessary deletions or contractions of these edges in parallel in constant time. For a deletion a processor may just link the two neighboring edges of each endpoint without running into conflicts with other good edges in S. For a deletion we have to link the four neighboring edges across, see Fig. 1.

Now we have all utilities in hand to build a PRAM algorithm, see Alg. 2. Observe that the only substantial changes when comparing to Alg. 1 are in lines 2, +4 and +6. Lines 2 and +4 take care that we only delete or contract a set of non-interfering edges. Line +6 ensures that after the contractions all edges still know about their endpoints – in the contraction phase this information might get corrupted since edge lists of several vertices might be linked into one large cyclic list.

We will only prove the EREW part of the Main Theorem. The derandomization tricks for symmetry breaking that are needed to avoid the $\log^* n$ factor on a CRCW PRAM are quite the same as given in Bodlaender and Hagerup (1995) and don't give new insights into the *MST* problem itself. So we content ourselves by proving the following proposition.

Proposition 12. *Algorithm 2 is correct and can be implemented in such a way such that it runs with a linear work load and a time of $O(\log n \log^* n)$ on a EREW PRAM.*

Proof. Correctness follows immediately from the correctness of Alg. 1. For the complexity, we will show that

(a) We always find a substantial fraction of the edges to be deleted or to contracted to obtain a logarithmical number of rounds of the algorithm.
(b) There are not more than $O(\log^* n)$ rounds that need $O(\log n)$ time.
(c) For the remaining rounds the time is not more than $O(\log^* n)$ per round.
(d) The overall work is linear.

The first tool to guarantee that we always have sufficiently many good edges is Thm. 11. Not every edge that is good for deletion can in fact be deleted. To be deleted it must have a weight that is higher than the one of a sibling that is close in one of the adjacency lists of the endpoints. But it is easy to see that at least every third edge that is good for deletion can in fact be deleted.

But these remaining good edges may interfere. If we build the conflict graph of these edges, see Bodlaender and Hagerup (1995), we notice that this graph has bounded degree, d say. For every constant d there is a constant $\varepsilon_d > 0$ such that a graph of bounded degree d has an independent set that consists of at least a ε_d-fraction of the vertices. In our conflict graph, an independent set of vertices corresponds to a set of rules that do not interfere, and so if we are also able to find such an independent set we get (a).

A fractional independent set, *i.e* an independent set of size at least a ε_d fraction, in a graph of bounded degree d can be found by two different methods:

Method A In $O(\log n)$ time and linear work (Hagerup (1990)).
Method B In $O(\log^* n)$ time and a work of $O(n \log^* n)$ by symmetry breaking (Goldberg et al. (1987)).

None of these methods alone would allow us to achieve the right complexity. As we will see below they lead to implementations of Alg. 2:

Variant A with $O(\log^2 n)$ time and linear work,
Variant B with $O(\log^* n \log n)$ time and $O(n \log^* n)$ work.

We use an argument of Bodlaender and Hagerup (1995) to combine the two methods, using Variant A for the first $\log^* n$ rounds, **phase A**, and Variant B for the remaining ones, **phase B**. By that combination we easily achieve the desired running time of $O(\log n \log^* n)$.

For the total work observe that after the $\log^* n_G$ rounds of phase A the size of the graph G is already reduced by a factor of $(1-\varepsilon)^{\log^* n_G}$ and the overall work done in phase B does not exceed

$$O\left(\log^*\left((1-\varepsilon)^{\log^* n_G} n_G\right)(1-\varepsilon)^{\log^* n_G} n_G\right) \leq O\left(\log^*(n_G)\,(1-\varepsilon)^{\log^* n_G} n_G\right). \quad (2)$$

But since $1-\varepsilon$ is a constant that is strictly smaller than 1 the factor $\log^*(n_G)(1-\varepsilon)^{\log^* n_G}$ is itself globaly bounded by a constant and phase B does a work of $O(n_G)$. So if we are able to show that phases A and B can be implemented as claimed above we are done.

Work of Phase X, X equal to A or B: We use Method X for line 2 and +4. For line +6, *i.e* the update of the endpoints for each edge, we will see below how this can in fact be done in constant time and a linear work load.

So in total we achieve a running time and work load as desired *per round*. But then by the same type of argument as in the proof of Thm. 5 the overall work for all rounds is only worsened by a constant factor of $\frac{1}{1-\varepsilon}$.

Updating the endpoints of edges: After having chosen a set of non-interfering edges for contraction consider a set S of vertices that are identified into one. Only two cases remain.

Case 1: If all vertices in S are of low degree then $|S| = 2$. This holds, since at most one edge per low degree vertex is chosen. We arbitrarily choose one vertex $v \in S$ and update all edges to have v as the other endpoint.

Case 2: Otherwise there is a unique vertex v_0 of high degree in S. This is so, since at least one endpoint of a contractable edge is of low degree. We update all edges of the low degree vertices to have v_0 as a new endpoint.

Since these updates only must be done for vertices of low degree we can easily avoid read conflicts that would occur when some edges of the same vertex are asking for the new endpoint at the same time. The updates are simply performed sequentially for each vertex.

So this update procedure concludes the proof that phases A and B have the complexities as claimed. □

Observe that the reduction factor of $1/8\mu_C$ from Thm. 11 for the number of good edges in each round depends on μ_C. In the presentation of the proof above, this factor gets much worsened because the maximum degree of the conflict graph also depends on μ_C. This can be avoided by a more involved argument. In fact we may change the definition of interference of contractable edges to be the same as for deletables. Then, in a more careful analysis of the symmetry breaking step it can be shown that this interference graph allows an independent set of size of about $1/4$ that still allows an $O(1)$ update of the endpoint information of low degree vertices that are contracted.

References

Bodlaender, H. and de Fluiter, B. (1997). Parallel algorithms for treewidth two. In Möhring et al., editors, *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science. Springer-Verlag. 23rd International Workshop WG '97, to appear.

Bodlaender, H. L. and Hagerup, T. (1995). Parallel algorithms with optimal speedup for bounded treewidth. In Fülöp, Z. and Gécseg, F., editors, *Automata, Languages and Programming*, volume 944 of *Lecture Notes in Comp. Sci.*, pages 268–279. Springer-Verlag. Proceedings of the 22nd International Colloquium ICALP'95.

Borůvka, O. (1926). O jistém problému minimálním. *Práca Moravské Přírodovědecké Společnosti*, 3:37–58. In Czech, cited after Tarjan (1983).

Chazelle, B. (1997). A faster deterministic algorithm for minimum spanning trees. In *38th Annual Symposion On Foundations of Computer Science*. IEEE, The Institute of Electrical and Electronics Engineers, IEEE Computer Society Press.

Cheriton, D. and Tarjan, R. E. (1976). Finding minimum spanning trees. *SIAM J. Computing*, 5:724–742.

de Fluiter, B. (1997). *Algorithms for Graphs of Small Treewidth (*Algoritmen voor grafen met kleine boombredte*)*. PhD thesis, Universiteit Utrecht.

Diestel, R. (1997). *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer-Verlag.

Dixon, B., Rauch, M., and Tarjan, R. E. (1992). Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21(6):1184–1192.

Goldberg, A. V., Plotkin, S. A., and Shannon, G. E. (1987). Parallel symmetry-breaking in sparse graphs. In *Proceedings of the Nineteenth Anual ACM Symposion on Theory of Computing*, pages 315–324. ACM, Assoc. for Comp. Machinery.

Hagerup, T. (1990). Optimal parallel algorithms on planar graphs. *Information and Computation*, 84:71–96.

Halperin, S. and Zwick, U. (1996). An optimal randomised logarithmic time connectivity algorithm for the EREW PRAM. *J. Comput. System Sci.*, 53(3):395–416.

Karger, D. R., Klein, P. N., and Tarjan, R. E. (1995). A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328.

King, V. (1997). A simpler minimum spanning tree verification algorithm. *Algorithmica*, 18(2):263–270.

Klein, P. N. and Tarjan, R. E. (1994). A randomized linear-time algorithm for finding minimum spanning trees. In *Proceedings of the Twenty Sixth Anual ACM Symposion on Theory of Computing*, pages 9–15. ACM, Assoc. for Comp. Machinery.

Kostochka, A. V. (1982). On the minimum of the Hadwiger number for graphs with given mean degree of vertices. *Metody Diskretn. Anal.*, 38:37–58.

Mader, W. (1967). Homomorphieeigenschaften und mittlere Kantendichte von Graphen. *Math. Ann.*, 174:265–268.

Poon, C. and Ramachandran, V. (1997). A randomized linear work EREW PRAM algorithm to find a minimum spanning tree. In Jaffar, J. and Leong, H. W., editors, *Algorithms and Computation*. Springer-Verlag. Proceedings of the Eighth Annual International Symposium ISAAC'97, to appear.

Tarjan, R. E. (1983). *Data structures and network algorithms*. Society of Industrial and Applied Mathematics (SIAM), Philadelphia.

Thomason, A. (1984). An extremal function for contractions of graphs. *Math. Proc. Cambridge Philos. Soc.*, 95:261–265.

Yao, A. C.-c. (1975). An $O(|E|\log\log|V|)$ algorithm for finding minimum spanning trees. *Inform. Processing Letters*, 4:21–23.

Reports from the group

"Algorithmic Discrete Mathematics"

of the Department of Mathematics, TU Berlin

566/1997 *Jens Gustedt:* Minimum Spanning Trees for Minor-Closed Graph Classes in Parallel

565/1997 *Andreas S. Schulz, David B. Shmoys, and David P. Williamson:* Approximation Algorithms

564/1997 *Uta Wille:* On Extending Closure Systems to Matroids

559/1997 *Matthias Müller–Hannemann and Karsten Weihe:* Improved Approximations for Minimum Cardinality Quadrangulations of Finite Element Meshes

554/1997 *Rolf H. Möhring and Matthias Müller–Hannemann:* Complexity and Modeling Aspects of Mesh Refinement into Quadrilaterals

551/1997 *Hans Bodlaender, Jens Gustedt and Jan Arne Telle:* Linear-Time Register Allocation for a Fixed Number of Registers and no Stack Variables

550/1997 *Karell Bertet, Jens Gustedt and Michel Morvan:* Weak-Order Extensions of an Order

549/1997 *Andreas S. Schulz and Martin Skutella:* Random-Based Scheduling: New Approximations and LP Lower Bounds

542/1996 *Stephan Hartmann:* On the NP–Completeness of Channel and Switchbox Routing Problems

536/1996 *Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein:* Improved Bounds on Relaxations of a Parallel Machine Scheduling Problem

535/1996 *Rainer Schrader, Andreas S. Schulz, and Georg Wambach:* Base Polytopes of Series-Parallel Posets: Linear Description and Optimization

533/1996 *Andreas S. Schulz and Martin Skutella:* Scheduling–LPs Bear Probabilities: Randomized Approximations for Min–Sum Criteria

530/1996 *Ulrich H. Kortenkamp, Jürgen Richter-Gebert, Aravamuthan Sarangarajan, and Günter M. Ziegler:* Extremal Properties of 0/1-Polytopes

524/1996 *Elias Dahlhaus, Jens Gustedt and Ross McConnell:* Efficient and Practical Modular Decomposition

523/1996 *Jens Gustedt and Christophe Fiorio:* Memory Management for Union-Find Algorithms

520/1996 *Rolf H. Möhring, Matthias Müller-Hannemann, and Karsten Weihe:* Mesh Refinement via Bidirected Flows: Modeling, Complexity, and Computational Results

519/1996 *Matthias Müller-Hannemann and Karsten Weihe:* Minimum Strictly Convex Quadrangulations of Convex Polygons

517/1996 *Rolf H. Möhring, Markus W. Schäffter, and Andreas S. Schulz:* Scheduling Jobs with Communication Delays: Using Infeasible Solutions for Approximation

516/1996 *Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein:* Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms

515/1996 *Christophe Fiorio and Jens Gustedt:* Volume Segmentation of 3-dimensional Images

514/1996 *Martin Skutella:* Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem

509/1996 *Soumen Chakrabarti, Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein:* Improved Scheduling Algorithms for Minsum Criteria

508/1996 *Rudolf Müller and Andreas S. Schulz:* Transitive Packing

506/1996 *Rolf H. Möhring and Markus W. Schäffter:* A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

505/1996 *Rolf H. Möhring and Dorothea Wagner:* Combinatorial Topics in VLSI Design: An Annotated Bibliography

504/1996 *Uta Wille:* The Role of Synthetic Geometry in Representational Measurement Theory

502/1996 *Nina Amenta and Günter M.Ziegler:* Deformed Products and Maximal Shadows of Polytopes

500/1996 *Stephan Hartmann and Markus W. Schäffter and Andreas S. Schulz:* Switchbox Routing in VLSI Design: Closing the Complexity Gap

498/1996 *Ewa Malesinska, Alessandro Panconesi:* On the Hardness of Allocating Frequencies for Hybrid Networks

496/1996 *Jörg Rambau:* Triangulations of Cyclic Polytopes and higher Bruhat Orders

Reports may be requested from:

S. Marcus
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: Marcus@math.TU-Berlin.DE

Reports are available via anonymous ftp from:

ftp.math.tu-berlin.de
cd pub/Preprints/combi
file Report–<number>–<year>.ps.Z