Technische Universität Berlin

FAKULTÄT II
MATHEMATIK UND
NATURWISSENSCHAFTEN
Institut für Mathematik

New Length Bounds for Cycle Bases

by

Michael Elkin
Ben-Gurion Univ. of the Negev, Dept. Computer Science, Israel
Christian Liebchen
TU Berlin, Institut für Mathematik, Germany
Romeo Rizzi
Università di Udine, DIMI, Italy
romeo.rizzi@dimi.uniud.it

No. 2007/22

New Length Bounds for Cycle Bases*

Michael Elkin
Ben-Gurion Univ. of the Negev, Dept. Computer Science, Israel
Christian Liebchen
TU Berlin, Institut für Mathematik, Germany
Romeo Rizzi
Università di Udine, DIMI, Italy
romeo.rizzi@dimi.uniud.it

29th May 2007

Abstract

Based on a recent work by Abraham, Bartal and Neiman (2007), we construct a strictly fundamental cycle basis of length $O(n^2)$ for any unweighted graph, whence proving the conjecture of Deo et al. (1982).

For weighted graphs, we construct cycle bases of length $O(W \cdot \log n \log\log n)$, where W denotes the sum of the weights of the edges. This improves the upper bound that follows from the result of Elkin et al. (2005) by a logarithmic factor and, for comparison from below, some natural classes of large girth graphs are known to exhibit minimum cycle bases of length $\Omega(W \cdot \log n)$.

We achieve this bound for weighted graphs by not restricting ourselves to strictly fundamental cycle bases—as it is inherent to the approach of Elkin et al.—but rather also considering weakly fundamental cycle bases in our construction. This way we profit from some nice properties of Hierarchically Well-Separated Trees that were introduced by Bartal (1998).

1 Introduction

We consider a simple undirected 2-connected graph $G = (V, E)$ with n nodes and m edges. A non-negative weight function w may be defined over the edge set E. Otherwise we are in the unweighted case, and $w \equiv 1$ is implicitly assumed. In the weighted case we denote by W the sum of the weights of all the edges. In the unweighted case $W = m$. A *circuit* C of G is a connected subgraph of G, in which each vertex has either degree two or zero. Sometimes, we may refer to C only through its edge set. The *incidence vector* $\gamma_C \in \{0,1\}^E$ of a circuit C

*Supported by the DFG Research Center MATHEON in Berlin and by the Israeli Academy of Science, grant 483/06.

is the characteristic vector of its edge set. The weight $w(C)$ of a circuit C is defined as the sum of the weights of its edges.

Cycle Bases. The *cycle space* $\mathcal{C}(G)$ of G is the linear vector subspace of $\mathrm{GF}(2)^E$ which is spanned by the incidence vectors of the circuits of G. It is well known that $\nu := \dim(\mathcal{C}(G)) = m - n + 1$.

A *cycle basis* $\mathcal{B}$ of G is a set of ν circuits of G which span $\mathcal{C}(G)$. The weight $w(\mathcal{B})$ of a cycle basis $\mathcal{B}$ is defined as $w(\mathcal{B}) = \sum_{C \in \mathcal{B}} w(C)$. In the *Minimum Cycle Basis (MCB) Problem*, we seek a cycle basis of minimum weight. These turn useful in many applications, see [Hor87] and references therein.

Given a spanning tree T of G, and a non-tree edge $e \in E \setminus T$, the unique circuit in $T \cup \{e\}$ is denoted by $C_T(e)$ and called the *fundamental circuit* induced by e with respect to T. It is well known that the set of all the fundamental circuits w.r.t. a spanning tree T is a cycle basis; these cycle bases have special and often desirable properties, and are called *strictly fundamental*. A cycle basis is called *weakly fundamental* if its circuits can be labeled as $C_1, C_2, \ldots, C_\nu$ such that

$$C_i \setminus (C_1 \cup \cdots \cup C_{i-1}) \neq \emptyset, \quad \text{for all } i = 2, \ldots, \nu. \tag{1}$$

Let $e_i \in C_i \setminus (C_1 \cup \cdots \cup C_{i-1})$. Notice that $\{e_1, \ldots, e_\nu\}$ are the co-tree arcs with respect to some spanning tree T of G. In particular, the circuit C_ν is a fundamental circuit with respect to that spanning tree T.

Metrics. The *metric* d_G over V that is associated with a weighted connected graph (G, w) is defined by the shortest-path distances in G with respect to w. The *diameter* $diam(G)$ of (G, w) is defined as

$$diam(G) := \max\{d_G(u, v) : \ u, v \in V\}. \tag{2}$$

Let $V' \subset V$ and consider the induced subgraph $G[V']$ of G. The *strong internal diameter of* V' is $diam(G[V'])$. In contrast, the *weak diameter of* V' is the maximum distance *in the original graph* G between two vertices in V'. A metric d on a set of elements V is said to *dominate* another metric d' on V if for all $u, v \in V$ we have $d'(u, v) \leq d(u, v)$. In particular, given a family $V_1, \ldots, V_k$ of connected subsets of V, a metric d dominates the strong internal diameters of $V_1, \ldots, V_k$ over (G, w), if for every subset V_i, $i = 1, \ldots, k$, and pair of vertices $u, v \in V_i$,

$$d(u, v) \geq diam(G[V_i]) \ . \tag{3}$$

Related Work. Deo et al. ([DKP82]) conjecture that every unweighted graph has a strictly fundamental cycle basis of length $O(n^2)$. Since strictly fundamental cycle bases specialize general cycle bases, their conjecture may also serve as the first estimate on the length of an MCB. For unweighted graphs, Horton ([Hor87]) introduced a heuristic for computing a cycle basis of length $O(n^2)$. The running time of Horton's algorithm is $O(m^3 n)$. Later, it was shown in [Lie03] that the resulting cycle basis is always weakly fundamental. As this

basis is not strictly fundamental in general, these results provide only a partial answer to Deo et al.'s conjecture.

Alon et al. ([AKPW95]) gave a more direct answer to Deo et al.'s conjecture. They prove that every weighted graph has a strictly fundamental cycle basis of length $O(W \cdot \exp(O(\sqrt{\log n \log\log n})))$, hereby proving Deo's conjecture in the case of graphs that are not too dense. Recently, this was improved by Elkin et al. ([EEST05]) to $O(W \cdot \log^2 n \log\log n)$. It is known that there exist graphs with large girth such that the length of their MCB is $\Omega(W \cdot \log n)$, see [Bol78]. Hence, for general graphs the bound of Elkin et al. is asymptotically tight up to a factor of $\Theta(\log n \log\log n)$.

Bartal ([Bar98]) showed that any metric can be approximated by a tree metric, which is defined on an auxiliary graph, called Hierarchically Well-Separated Tree (HST). More precisely, the resulting tree metric dominates the initial metric and over-estimates it—in the deterministic case on average—by a factor of at most $\alpha(n) \in O(\log n \log\log n)$. Later, by introducing further techniques, this bound was improved to the tight $O(\log n)$ ([FRT03]).

In parallel, MCB algorithms that improve the running time of Horton's algorithm [Hor87] ($O(m^3 n)$) were developed. The presently fastest known MCB algorithm has running time $O(m^2 n / \log n + n^2 m)$ [MM07]. Also constant-factor approximation algorithms had been proposed [KMMP04, KMM07, MM07], e.g. a 2-approximation can be constructed in time $\tilde{O}(m^{3/2} n^{3/2}) + O(m^\omega)$, where $\omega < 2.376$ is the matrix multiplication exponent. However, the worst-case running time of these algorithms is at least $\Omega(n^3)$, even for sparse graphs. Faster algorithms are only known for special graph classes. In particular, an $O(n^2 \cdot \log n)$ time algorithm for planar graphs was devised by Hartvigsen and Mardon [HM94]). The algorithm of [EEST05] can be used as an $O(\log^2 n \log\log n)$-approximation for the MCB problem, and its running time is $O(m \log n + n \log^2 n + \sum_{Q \in \mathcal{B}} |Q|)$, where $\{Q \mid Q \in \mathcal{B}\}$ is the cycle basis returned by the algorithm.

Contribution. We develop a way to profit from Bartal's techniques in the context of the Minimum Cycle Basis Problem. More precisely, using HSTs [Bar98] we construct a weakly fundamental cycle basis of length $O(W \cdot \log n \log\log n)$ for any weighted 2-connected undirected graph. Hereby, we improve the previously known best upper bound on the length of general minimum cycle bases by a factor of $\log n$. Also, due to the lower bound of $\Omega(W \cdot \log n)$, we conclude that our construction is tight up to a factor $\Theta(\log\log n)$.

Furthermore, based on a recent work by Abraham, Bartal and Neiman [ABN07], we construct a strictly fundamental cycle basis of length $O(n^2)$ for any unweighted graph, whence proving the conjecture of Deo et al. [DKP82].

We also note that our construction of the weakly fundamental cycle basis can be seen as an efficient $O(\log n \log\log n)$-approximation algorithm for computing MCB. Specifically, the running time of our algorithm is $O(n^2 \cdot \log^{c+1} n + \sum_{Q \in \mathcal{B}} |Q|)$, where $\{Q \mid Q \in \mathcal{B}\}$ is the cycle basis constructed by our algorithm, and $c = O(1)$ is the universal constant that is implicit in the result of Cohen

and Zwick [CZ01]. (See Section 4.) In particular, in the important case of sparse graphs, among the class of algorithms with computation time $o(n^3)$ we obtain the best known approximation guarantee. Moreover, we also present a randomized variant of our algorithm that, for an integer parameter $k > 2$, has expected running time $O(\log n \cdot \max\{n^2, k \cdot m \cdot n^{1/k}\} + \sum_{Q \in \mathcal{B}} |Q|)$ and returns an $O(k \cdot \log n \log\log n)$-approximate MCB. The latter variant employs a recent result of Baswana and Kavitha [BK06] for computing almost shortest paths.

This paper is organized as follows. In Section 2 we overview the key properties of HSTs. Based on HSTs, in Section 3 we present the algorithm for computing a weakly fundamental cycle basis of length $O(W \cdot \log n \log\log n)$. In Section 4 we analyze the running time of our algorithm. The proof of the conjecture of Deo et al. [DKP82] is given in Section 5.

The bound of $O(W \cdot \log n \log\log n)$ could be obtained using the construction of HSTs due to Bartal [Bar98], based on [Sey95]. However, it is not clear whether this construction can be implemented using a nearly linear running time. We present a new construction of HSTs with distortion $O(\log n \log\log n)$ that can be implemented in a nearly linear time. This construction is based on the construction of [EEST05] of low stretch spanning trees, and could be of independent interest.

We remark that for our construction to work it is crucial that the HST that it uses provides bounds with respect to *strong diameters*. (See Section 2 for the definition.) To our knowledge the construction of HSTs due to Fakcharoenphol et al. [FRT03] that achieves the optimal $O(\log n)$ bound provides this bound with respect to *weak diameters*. So far we were not able to strengthen it so that the same bound will apply to strong diameters as well. Strengthening the result of [FRT03] in this way will immediately imply the tight $O(W \log n)$ bound for the MCB.

2 Techniques and Main Idea

In following the presentation of our algorithm, the reader should keep in mind the following simple—though useful—property of weakly fundamental cycle bases.

Remark 1 *One way to construct a weakly fundamental cycle basis $\mathcal{B}$ is to define some spanning tree F and provide an order of the co-tree edges $E \setminus F$, say $e_1, \ldots, e_\nu$, where $F = e_{\nu+1}, \ldots, e_m$. Then, the basic circuit C that we associate with a co-tree edge $e_i \in E \setminus F$ has to contain e_i and may use some edges of the tree F and some of the co-tree edges $e_1, \ldots, e_{i-1}$. But C must not contain any of the edges $e_{i+1}, \ldots, e_\nu$.*

The key to our algorithm for constructing a short weakly fundamental cycle basis will thus be the order in which we process the edges of G. And this order will be dictated by a decomposition of G, the HST. This object was introduced by Bartal ([Bar96]).

Consider a nested—or laminar—family $\mathcal{S}$ of connected subgraphs of G. We only consider such families in which the maximal element is the entire vertex set V, and all the singletons in V appear as minimal elements in $\mathcal{S}$. For $U \in \mathcal{S}$ we denote by $G[U]$ its induced subgraph, and call it *cluster*.

We resort to the standard way of representing such a laminar family $\mathcal{S}$ by a rooted tree T: the nodes of T are the sets in $\mathcal{S}$, and there is an arc (U,U') between two sets U and U' in $\mathcal{S}$ iff $U' \subseteq U$ and $U' \subseteq U'' \subseteq U$ holds for no U'' in $\mathcal{S}$. Notice that V is the root of T and the leaves of T are in 1,1-correspondence with the nodes in V. More generally, each node of T is the disjoint union of its children. Such a tree is called a *V-tree*. In a V-tree T, the *least common ancestor* $\Lambda(u,v)$ of a pair of vertices $u \neq v$ is the vertex in the unique uv-path in T that is closest to the root of T. Its associated cluster in G is just $G[\Lambda(u,v)]$.

Definition 2 *A* hierarchically well-separated tree (HST) *of a weighted connected graph (G,w) is a V-tree T with weights c on the arcs, with the following properties:*

1. *$c(U,U_1) = c(U,U_2)$ for any two arcs of T with a common tail;*
2. *$c(U',U'') \leq \frac{1}{2}c(U,U')$ for any two subsequent arcs (U,U') and (U',U'') in T;*
3. *for each node U of T, the induced graph $G[U]$ is connected.*

An HST (T,c) induces a tree metric d_{HST} over V, where $d_{HST}(u,v)$ is the distance between u and v in (T,c), i.e. $d_{HST}(u,v) := c(P_{u,v})$, with $P_{u,v}$ being the unique u,v-path in T. We only consider HSTs in which d_{HST} dominates $d_{G,w}$. Whenever this is the case we say that the HST dominates the metric of (G,w) with respect to *weak diameters*. A stronger condition is to require that for every cluster U of the HST, the metric $d_{G[U],w}$ of the induced graph $(G[U],w)$ of the cluster U is dominated by the metric d_{HST}. Whenever this is the case we say that the HST dominates the metric of (G,w) with respect to *strong diameters*.

We are interested in HSTs of low average stretch, that is, we need a bound on $\sum_{\{u,v\}\in E} d_{\mathrm{HST}}(u,v)$.

Theorem 3 (stated in [Bar98] for weak diameters) *There exists a function $\alpha(n) \in \Theta(\log n \log\log n)$ such that for every weighted graph (G,w) there exists a polynomial-time algorithm to construct an HST T with metric d_{HST} that dominates the metric of (G,w) with respect to strong diameters that satisfies*

$$\sum_{\{u,v\}\in E} d_{HST}(u,v) \leq \alpha(n) \cdot W. \tag{4}$$

We provide a sketch of an explicit proof of this theorem in Appendix A. This sketch is based on the construction of low stretch spanning trees due to [EEST05].

In the algorithm for computing a short weakly fundamental cycle basis that we are about to present in the next section, we will process the clusters of T bottom-up, i.e. we will only start working in a cluster when we finished working in all of its descendents.

3 Computing a Short Weakly Fundamental Cycle Basis

Now we present our Algorithm 1 that for a given weighted graph (G, w) of total weight W constructs a weakly fundamental cycle basis of objective value at most

$$(16 \cdot \alpha(n) + 1) \cdot W, \quad \alpha \in O(\log n \log\log n), \tag{5}$$

where α refers to Bartal's function in Theorem 3.

The first line of Algorithm 1 is a call to Procedure MAKE-HST, providing a hierarchically Well-Separated Tree T that dominates the strong internal diameters of its corresponding clusters of(G, w), and satisfies the length-bound in (4) (Theorem 3). In principle, it was shown in [Bar98] how this could be performed in polynomial time. However, in Section 4 and Appendix A we provide a more efficient algorithm.

Algorithm 1 resorts on a further external procedure: we assume that if (G, w) is a weighted graph and u is a node of G, then DIJKSTRA$((G, w), u)$ computes the u-rooted shortest path tree $S = e_1, e_2, \ldots, e_{n-1}$ in (G, w); it is assumed that the edges of S are given in the same order as they would be put into S by the classical Dijkstra's algorithm ([Dij59]). In other words, the sequence $e_1, e_2, \ldots, e_s$ satisfies the following property. Consider a pair of edges $e = (v, w), e' = (v', w')$ in the sequence with v (respectively, v') being closer to the root u of S than w (resp., w'). Suppose that e belongs to the path connecting the root u of the tree S with the endpoint v' of $e' = (v', w')$. For every pair of edges e, e' as above it is required that the edge e appears in the sequence before e'.

After the hierarchically well-separated tree T and the corresponding metric d_{HST} have been obtained, the actual iterative construction of the circuits to be put in the basis $\mathcal{B}$ can start. As anticipated in Remark 1, we will accompany the construction of $\mathcal{B}$ by also computing a spanning tree F of G plus an ordering of the non-tree edges $E(G) \setminus F$ of G. Thus, in our iterative process we start with $\mathcal{B} := F := Z := \emptyset$, and collect in the set Z all the edges that were already processed according to the ordering that we are about to define. Observe that Z will contain tree edges *and* non-tree edges. When we process an edge e (and thus add e to Z), this has one of the two possible results: Either $F \cup \{e\}$ is acyclic, then we add e to F. Or $F \cup \{e\}$ contains some cycle, then we add to $\mathcal{B}$ a circuit through e which only uses edges in Z, and which is sufficiently short. For simplicity, you may think of a shortest circuit through e that only uses edges in Z.

We now give a description of the order in which we are going to process the edges of G. Processing the edges will be grouped according to the clusters

of T. We will color a node $U \in V(T)$ *green*, if all the edges of its corresponding cluster $G[U]$ have been processed, i.e. $E(G[U]) \subseteq Z$. Otherwise, U will have to remain colored *red*. At the start of the algorithm we may thus color in green all the leaves of T, and the algorithm terminates when the root of T finally becomes green. The order by which we pass through the clusters of the HST will be bottom-up, i.e. we may only work in a node $U \in V(T)$, if all its descendents are colored green. Hence, when we start working in $G[U]$, some edges of $G[U]$ may have already been processed while working in clusters that correspond to descendents of U in T. The other edges, i.e. $E(G[U]) \setminus Z$, are called *U-proper*. We process all the U-proper edges in a specific order which we specify in the very next paragraph. After doing so, we may color U green.

The order in which we process the U-proper edges of a cluster $G[U]$ is as follows. We first compute Dijkstra's shortest path tree $S \subseteq E(G[U])$ in this cluster, rooted at some arbitrary vertex u of U. During this procedure, whenever a U-proper edge e gets added to S, we process this edge. Second, after all the edges of S have been processed, i.e. $S \subseteq Z$, we process—in arbitrary order—all the U-proper edges in $E(G[U]) \setminus S$.

In practice, we suggest to compute the basic circuits which we add in Step 24 of Algorithm 1 as the shortest circuits through e in $Z \cup \{e\}$. In general, this would lead to shorter bases. However, our asymptotic analysis would not improve.

For a cluster $G[U]$, let

$$\Delta(U) := diam_{\mathrm{HST}}(G[U]) := \max\{d_{\mathrm{HST}}(u,v),\, u,v \in U\}.$$

By Theorem 3, d_{HST} dominates the metric with respect to strong diameters. Hence

$$\Delta(U) \geq diam(G[U]). \tag{6}$$

Moreover, for any two vertices $v_1, v_2 \in V(G)$ for which $\Lambda(v_1, v_2) = U$, we may bound $\Delta(U)$ as follows,

$$\Delta(U) \leq 2 \cdot \left(\frac{1}{2} d_{\mathrm{HST}}(v_1, v_2) \sum_{k=0}^{\infty} \frac{1}{2^k} \right) \leq 2 \cdot d_{\mathrm{HST}}(v_1, v_2), \tag{7}$$

where inside the brackets we are considering one path in the HST from U to one of its leaves, and use Property (2) of Definition 2.

Lemma 4 *Let U be the cluster that is currently processed by Algorithm 1. Each circuit C that we add to the basis $\mathcal{B}$ in Step 18 as the shortest circuit through $e_i = \{v_1, v_2\} \in S$ in Z has weight $w(C)$ at most*

$$w(C) \leq 5 \cdot d_{HST}(v_1, v_2). \tag{8}$$

Proof. Here, $e_i \notin Z$, and thus $\Lambda(v_1, v_2) = U$. Hence we will first identify a bound on $w(C)$ in terms of $\Delta(U)$, and finally apply (7). The bound on $w(C)$ is obtained by identifying a (possibly different) cycle C' in Z and with $e_i \in C'$, and establishing $w(C') \leq 5 \cdot \Delta(U)$. The cycle C' will be constructed out of two subpaths of S plus a shortest path within a subcluster of U.

Algorithm 1 `wfcb`

Require: A weighted input graph (G, w).

Ensure: A weakly fundamental cycle basis $\mathcal{B}$ of G with $w(\mathcal{B}) = O(W \cdot \log n \log\log n)$.

1: $(T, d_{\mathrm{HST}}) :=$ MAKE-HST(G, w); // see appendix
2: $\mathcal{B} := \emptyset$;
3: $F := \emptyset$; // becomes a spanning tree of G related to $\mathcal{B}$ as in Remark 1
4: $Z := \emptyset$; // the set of edges already considered. In particular, $F \subseteq Z$ and each edge in $Z \setminus F$ belongs to some circuit in $\mathcal{B}$.
5: Color with *green* all the leaves of T, and with *red* all other nodes;
6: **while** some node of T is *red* **do**
7: Let U be any *red* node of T whose children are all *green*;
8: Color U with *green*;
9: Let u be an arbitrary vertex of $G[U]$;
10: $S = \{e_1, \ldots, e_s\} :=$ DIJKSTRA$((G[U], w), u)$; // Recall that $G[U]$ is connected.
11: **for** $i = 1$ to s **do**
12: **if** $e_i \in Z$ **then**
13: // void — e_i already processed in a subcluster of U.
14: **else if** $F \cup \{e_i\}$ is acyclic **then**
15: $F := F \cup \{e_i\}$, $Z := Z \cup \{e_i\}$;
16: **else**
17: // $e_i \notin Z$, and $F \cup \{e_i\}$ contains some cycle
18: Add to $\mathcal{B}$ the shortest circuit through the edge e_i that only uses edges in Z;
19: $Z := Z \cup \{e_i\}$;
20: **end if**
21: **end for**
22: // By now we have $S \subseteq Z$.
23: **for all** $e \in E(G[U]) \setminus Z$ **do**
24: Add to $\mathcal{B}$ the unique circuit in $S \cup \{e\}$;
25: $Z := Z \cup \{e\}$;
26: **end for**
27: // Here we have $E(G[U]) \subseteq Z$.
28: **end while**

We assume without loss of generality that v_1 is closer to the root u of the shortest-path tree S in $G[U]$ than v_2. Denote by U_2 the child of U in T that contains v_2.

We are in the situation where $F \cup \{e_i\}$ contains some circuit Q with $e_i \in Q$. Consider the cut $X \subset E(G[U])$ that corresponds to $(U_2, U \setminus U_2)$. Observe that $e_i \in X$. As every cycle has even intersection with any cut, and since $\emptyset \neq \{e_i\} \in Q \cap X$, we know that $|Q \cap X| \geq 2$, and in particular $|F \cap X| \geq 1$.[1]

But since edges in X have precisely one endpoint in U_2, they cannot be contained in any subcluster of U and thus are processed during U's iteration of the **while**-loop. Hence, each element of $F \cap X$ is part of the shortest-path tree S in U.

We compose the not necessarily simple cycle C' out of the following three parts: First, the unique path $P_{ue_i} \subset S$ from u to e_i. Second, one path $P_{uf} \subset S$ from u to some edge f such that $P_{uf} \cap X = \{f\} \neq \{e_i\}$, see Figure 1 for the location of e, f, u, and U_2. As both these paths are subpaths of the shortest-

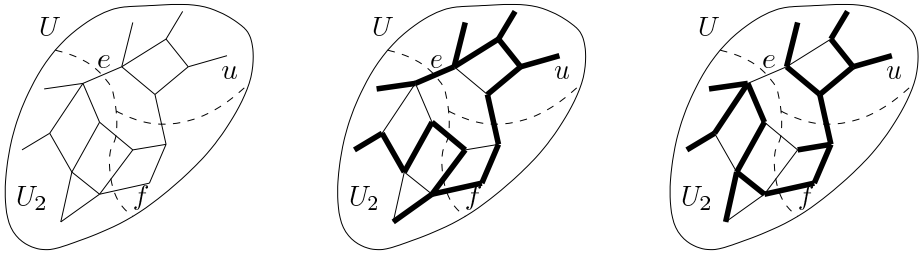

Figure 1: Comparing for some cluster U (on the left) the u-rooted shortest-path tree S (in the middle) to the growing spanning tree F of the input graph (on the right).

path tree S in $G[U]$, for $P \in \{P_{ue_i}, P_{uf}\}$ we know that $w(P) \leq \mathit{diam}(G[U]) \overset{(6)}{\leq} \Delta(U)$...

The third and last part of the cycle C' is selected as a shortest path P_{e_if} in $G[U_2]$ between the corresponding endpoints of e_i and f. Such a path exists because any cluster is connected, and P_{e_if} only uses edges in Z because it stays within the cluster U_2 all of whose edges were already processed, cf. the comment in Line 27 of Algorithm 1.

To bound the length of P_{e_if}, recall the second property of an HST and consider the subgraph $G[U_2]$ of G: Our bound $\Delta(U_2)$ on the strong internal diameter of $G[U_2]$ is by (at least) a factor of k smaller than the corresponding bound $\Delta(U)$ on the strong internal diameter of $G[U]$. Hence, we finally

[1] One could even show by induction that in fact $|F \cap X| = 1$.

obtain $w(P_{e_i f}) \leq diam(G[U_2]) \leq \Delta(U_2) \leq \frac{\Delta(U)}{2}$. Overall we obtain

$$\begin{aligned} w(C) &\leq w(C') \leq w(P_{ue_i}) + w(P_{uf}) + w(P_{e_i f}) \\ &\leq 2 \cdot \Delta(U) + \frac{\Delta(U)}{2} = \frac{5}{2} \cdot \Delta(U) \overset{(7)}{\leq} 5 \cdot d_{\mathrm{HST}}(v_1, v_2). \end{aligned}$$

□

Lemma 5 *Let U be the cluster that is currently processed by Algorithm 1. Each circuit C that we add to the basis $\mathcal{B}$ in Step 24 for some non-tree edge $e = \{v_1, v_2\} \in E(G[U]) \setminus S$ has weight $w(C)$ at most*

$$w(C) \leq 4 \cdot d_{HST}(v_1, v_2) + w(e). \tag{9}$$

Proof. We identify a cycle C' that respects the length-bound in (9), but which is already a superset of the circuit C that we add to the basis in Step 24. Again, from $e = \{v_1, v_2\} \notin Z$ we conclude that $U = \Lambda(v_1, v_2)$. Let P_1 be the unique uv_1-path in S (thus being a shortest path), and P_2 be the unique uv_2-path in S. By defining $C' := P_1 \cup P_2 \cup \{e\}$, we obtain

$$\begin{aligned} w(C) \leq w(C') &\leq w(P_1) + w(P_2) + w(e) \overset{(6)}{\leq} 2 \cdot \Delta(U) + w(e) \\ &\overset{(7)}{\leq} 4 \cdot d_{\mathrm{HST}}(v_1, v_2) + w(e). \end{aligned}$$

□

Theorem 6 *Every weighted graph (G, w) with total edge weight W admits a weakly fundamental cycle basis of length at most $W \cdot O(\log n \log\log n)$. Such a basis can be computed in polynomial time.*

Proof. We apply Algorithm 1 to construct a weakly fundamental cycle basis $\mathcal{B}$. The length $w(\mathcal{B})$ of $\mathcal{B}$ is the sum of the weights of all the basic circuits which we add in Steps 18 and 24 of the algorithm. For every non-tree edge $e = \{v_1, v_2\} \in E(G) \setminus F$ according to Lemma 4 and Lemma 5, the length $w(C)$ of the circuit C that is added while processing e is bounded by

$$w(C) \leq w(e) + 5 \cdot d_{\mathrm{HST}}(v_1, v_2).$$

By summing this bound over all edges—although only non-tree edges with respect to F actually count—we bound $w(\mathcal{B})$ from above with

$$\sum_{\{u,v\} \in E} w(e) + 5 \cdot d_{\mathrm{HST}}(u, v) \overset{\text{Thm.3}}{\leq} (5 \cdot \alpha(n) + 1) \cdot W, \tag{10}$$

where $\alpha(n) \in O(\log n \log\log n)$ refers to Bartal's coefficient in Theorem 3. □

We remark that no effort was made to optimize constant factors in this proof.

4 The Running Time

In this section we show that a variant of our algorithm can be implemented very efficiently.

The algorithm starts constructing an HST T for G by means of Algorithm 4 (described in Appendix A). By Theorem 12, this step requires $O(m \log n + n \log^2 n)$ time. In addition to the HST, Algorithm 4 can be made to provide a shortest-paths tree (henceforth, SPT) S_U for every cluster U of the HST. Morever, these spanning trees are constructed within the same running time $O(m \log n + n \log^2 n)$. Indeed, these trees get essentially constructed as a side product of the other computations, and, in any case, one can construct them from scratch starting from the HST, and within the same time.

Also, more below we will exploit more than once the following fact: in a rooted tree S, given a pair of vertices u, v in S, the least common ancestor (henceforth, LCA) of u and v, denoted $\Lambda_S(u, v)$, can be found in $O(|S[u, v]|)$ time, where $S[u, v]$ denotes the unique u, v-path in S. Indeed, when computing the LCA of nodes u and v we simply go up the tree from the two nodes u and v in parallel and rise towards the root of the tree meanwhile marking the nodes encountered as visited, until finding an already visited node. This process clearly takes $O(|S[u, v]|)$ time since, proceeding in parallel, we visit at most $2\,|S[u, v]|$ nodes.

Next, the algorithm invokes a routine for all-pairs-almost-shortest-paths (henceforth, APASP) computation in each cluster U of the HST. To this end we employ the algorithm of Cohen and Zwick [CZ01] that produces paths of lengths at most 3 times greater than the corresponding shortest paths. The running time of this algorithm in a cluster U is $O(|U|^2 \cdot \log^c |U|)$, where $c = O(1)$ is the universal constant introduced in [CZ01].

After the preprocessing is finished, the algorithm proceeds to performing steps 2-9, 11-28 of the Algorithm 1. (All steps except for the step 1 that constructs an HST, and for the step 10 that constructs SPTs.) Details on the implementation of these steps are provided below.

The edge set Z can be maintained using a balanced search tree. We need to support $O(m)$ insert and search queries for it, and altogether this requires $O(m \log n)$ time. (We denote $m = |E|$.)

The algorithm also needs to maintain a forest F, and given an edge e to test whether $F \cup \{e\}$ is acyclic. If this is the case, the edge e is inserted into F. Such incremental maintaining of a forest can be implemented in a standard way in $O(m \cdot \gamma(n))$ time, where $\gamma(n)$ is the inverse-Ackermann function [Tar75]. (See also [CLRS01], Chapters 21 and 23.)

Also, when an edge $e = (v_1, v_2)$ is added to F, we associate this edge with the edge (U', U) of the HST such that $v_1, v_2 \in U$, $v_1 \in U'$, $v_2 \in U \setminus U'$. Note that edges are inserted into F only during the first for-loop (lines 11-21 of the Algorithm 1), and all edges e processed in this loop belong to a cut between a child cluster U' of the processed cluster U and $U \setminus U'$. To implement this association, for every edge (U', U) of the HST T we maintain a list of edges $e = (v_1, v_2) \in F$ with $v_1 \in U'$, $v_2 \in U \setminus U'$. When an edge e is inserted into F,

it is also added to the list associated with the appropriate edge of the HST T. Locating the appropriate edge of the HST T requires $O(\log n)$ time, and thus the overall running time for maintaining the forest F becomes $O(m \log n)$.

Perhaps the most significant modification to the Algorithm 1 that we introduce is on step 18. This step adds the shortest circuit through e_i that uses only edges of Z, and a straightforward implementation of this step requires $O(|Z|)$ time. To speed up this step we form a circuit as described in the proof of Lemma 4.

Specifically, consider an edge $e_i = (v_1, v_2)$ for which the step 18 needs to construct a circuit. This edge satisfies that $e_i \notin Z$, and $F \cup \{e_i\}$ contains a cycle. Let u be the root of the SPT S for the cluster $U = \Lambda(v_1, v_2)$, and U' be the child of U in the HST T that contains the vertex v_2. As in the proof of Lemma 4, the vertex v_2 is the endpoint of e_i that is farther from the root u of S. Assume without loss of generality that $u \in U \setminus U'$. Then $v_1 \in U \setminus U'$, $v_2 \in U'$.

Let X denote the cut between U' and $U \setminus U'$, and let $f \in F \cap X$, $f \neq e_i$, be some other edge in this cut. Such an edge exists and belongs to S, as was shown in the proof of Lemma 4. Moreover, this edge can be found within $O(\log n)$ time using the data structures of our algorithm. Let $f = (w_1, w_2)$, and assume without loss of generality that $w_1 \in U \setminus U'$, $w_2 \in U'$.

Let $x = \Lambda_S(v_1, w_1)$ be the LCA of v_1 and w_1 in S. As we have already mentioned, in S, the paths $P_{v_1,x}$ and P_{x,w_1} among these nodes can be computed within $O(|P_{v_1,x}| + |P_{x,w_1}|)$ time.

The approximate shortest path P_{v_2,w_2} in U' between v_2 and w_2 can be computed within $O(|P_{v_2,w_2}|)$ time using the APASP data structure constructed during the preprocessing. Recall that the circuit $Q(e_i)$ associated with the edge $e_i \in S$ is constructed by concatenation of the three paths $P_{v_1,x}$, P_{x,w_1}, and P_{v_2,w_2}, along with the edges e_i and f. Hence constructing the circuit $Q(e_i)$ requires $O(|Q(e_i)| + \log n)$ time.

Finally, consider the second for-loop (lines 23-26). In this loop, for every edge $e = (v_1, v_2) \in E(G[U]) \setminus Z$, the unique circuit Q in $S \cup \{e\}$ is constructed. To implement this efficiently, the algorithm finds $x = \Lambda_S(v_1, v_2)$, and computes the two paths $P_{v_1,x}$ and P_{x,v_2}. This computation requires $O(|Q(e)|)$ time.

The overall running time of the algorithm is

$$O(n^2 \log^{c+1} n) + \sum_{e \in E \setminus F} (|Q(e)| + O(\log n)) \;=\; O(n^2 \log^{c+1} n) + \sum_{e \in E \setminus F} |Q(e)| \,,$$

where $\{Q(e) \mid e \in E \setminus F\}$ is the circuit basis returned by the algorithm. Observe that the second term $\sum_{e \in E \setminus F} |Q(e)|$ is the size of the output. As shown in Section 3, this circuit basis is an $O(\log n \log\log n)$-approximation for the MCB. (The fact that we use 3-approximate shortest paths between v_2 and w_2 instead of exact shortest paths between them may increase the constant factor hidden in the O-notation by a factor of at most 3.) In the special case of unweighted graphs, since the MCB has length $O(n^2)$ by Theorem 9, then $\sum_{e \in E \setminus F} |Q(e)| = O(n^2 \cdot \log n \log\log n)$.

We summarize the analysis of the running time with the following theorem.

Theorem 7 *Consider a graph $G = (V, E, \omega)$ with $W = \sum_{e \in E} \omega(e)$. The variant of the Algorithm 1 that was described in this section computes a circuit basis $\{Q(e) \mid e \in E \backslash F\}$, for some spanning tree F of G, of length $O(W \cdot \log n \log\log n)$ in time $O(n^2 \cdot \log^{c+1} n + \sum_{e \in E \backslash F} |Q(e)|)$. Moreover, if G is unweighted, then the running time is $O(n^2 \log^{c+1} n)$, where c is the universal constant from the result of [CZ01].*

Finally, if randomization is allowed, then the recent results of Baswana and Kavitha [BK06] can be used instead of those of Cohen and Zwick [CZ01] to improve the running time of our algorithm further. Specifically, for an integer parameter $k > 2$, an algorithm with expected running time $O(\min\{n^2, k \cdot m \cdot n^{1/k}\})$ that constructs $(2k-1)$-approximate distance oracles is described in [BK06]. These oracles answer distance queries in $O(k)$ time.

Consider the following modification of our algorithm. During the preprocessing, instead of performing the APASP computation in each cluster U, the algorithm will construct the oracles of [BK06] in each cluster. The running time of this step is $O(\log n \cdot \min\{n^2, k \cdot m \cdot n^{1/k}\})$. When the algorithm needs a path P_{v_2,w_2} between v_2 and w_2 in U', it invokes the distance oracles. Since it does so at most once for each edge $e \in E \setminus F$, there are altogether $O(m)$ invocations of distance oracles in the algorithm. Hence, altogether they require $O(k \cdot m)$ time. Hence, overall the randomized variant of our algorithm requires $O(\log n \cdot \min\{n^2, k \cdot m \cdot n^{1/k}\} + \sum_{e \in E \backslash F} |Q(e)|)$ expected time. Since the oracles provide a $(2k-1)$-approximation of the actual distances, the length of the constructed circuit basis may also have length at most $O(k)$ times greater than in the deterministic variant. (In the deterministic variant we use 3-approximation.) In other words, the produced circuit basis is an $O(k \log n \log\log n)$-approximation of the MCB.

5 Strictly Fundamental Circuits

In this section we show that for every unweighted undirected graph there exists a strictly fundamental cycle basis of size $O(n^2)$. Our proof is based on a recent result of Abraham et al. ([ABN07]) that is described below.

Given a spanning tree T and a pair of vertices $x, y \in V$ as above, the *distortion* of the pair x, y with respect to T, denoted $distort_T(x, y)$, is defined as the ratio between the distance between x and y in T and the distance between x and y in G, i.e.,

$$distort_T(x, y) \;=\; \frac{d_T(x, y)}{d_G(x, y)} \;.$$

The ℓ_1-*distortion* of a tree T is defined as the expectation of the value of $distort_T(x, y)$, where the pair of vertices $\{x, y\}$ is drawn uniformly at random from the set of all (non-ordered) pairs of vertices $\binom{V}{2}$. In other words,

$$distort(T) \;=\; \mathbb{E}_{\mathcal{U}}(distort_T(x, y)) \;,$$

where $\mathcal{U}$ stands for the uniform distribution over $\binom{V}{2}$.

We will use the following result of Abraham et al. ([ABN07]). (Theorem 2).

Theorem 8 (Thm. 2 in [ABN07]) *Any weighted graph G with n vertices contains a spanning tree T with distort$(T) = O(1)$.*

Note that

$$\mathbb{E}_{\mathcal{U}}(distort_T(x,y)) = \frac{1}{\binom{n}{2}} \cdot \sum_{\{x,y\}\in\binom{V}{2}} \frac{d_T(x,y)}{d_G(x,y)} \ ,$$

and thus,

$$\sum_{\{x,y\}\in\binom{V}{2}} \frac{d_T(x,y)}{d_G(x,y)} \ = \ O(n^2) \, .$$

¿From this point on we consider only unweighted graphs $G = (V, E)$. For every edge $e = \{u, v\} \in E$, $d_G(u, v) = 1$. Hence

$$\begin{aligned} \sum_{e=\{u,v\}\in E\setminus E(T)} d_T(u,v) \ &= \ \sum_{e=\{u,v\}\in E\setminus E(T)} \frac{d_T(u,v)}{d_G(u,v)} \ \leq \ \sum_{\{x,y\}\in\binom{V}{2}} \frac{d_T(x,y)}{d_G(x,y)} \\ &= \ O(n^2) \, . \end{aligned}$$

Let $e_1 = \{u_1, v_1\}, e_2 = \{u_2, v_2\}, \ldots, e_\nu = \{u_\nu, v_\nu\}$ be the edges of $E \setminus E(T)$, ordered arbitrarily. Let $C_1, C_2, \ldots, C_\nu$ be the fundamental cycle basis with respect to T, with C_i being the fundamental cycle of T induced by the edge e_i, for $i \in \{1, 2, \ldots, \nu\}$. Then

$$\sum_{i=1}^{\nu} |C_i| \ = \ \sum_{i=1}^{\nu} (d_T(u_i, v_i) + 1) \ = \ \nu + \sum_{i=1}^{\nu} d_T(u_i, v_i) \ = \ O(n^2) \ .$$

Theorem 9 *For any n-vertex graph there exists a cycle basis of length $O(n^2)$.*

6 Conclusions

After upper bounds of $O(n^2)$ (for unweighted graphs) and $O(W \cdot \log^2 n \log\log n)$ on the weights of Minimum Cycle Bases (MCB), we offer a polynomial-time algorithm that computes a weakly fundamental cycle basis of weight $O(W \cdot \log n \log\log n)$. Given the fact that there exist classes of graphs of large girth whose MCB has length $\Omega(W \cdot \log n)$, we consider our general bound fairly close to being asymptotically tight. At present we are working at upper bounds of $O(W \cdot \log n)$ for strictly fundamental cycle basis.

A strong form of what is now known as Deo's conjecture was also proposed by Deo et al. in [DKP82]:

> Every unweighted graph has a strictly fundamental cycle basis of length at most $\frac{3}{2}n^2$.

Our approach and techniques seem inadequate to prove this statement.

Acknowledgment

We thank Kurt Mehlhorn for his feedback on an earlier version of the manuscript, and Alan Bertossi for making our collaboration possible.

References

[ABN07] Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion. In *Proc. of the Symp. on Discrete Algorithms, to appear*, 2007.

[AKPW95] Noga Alon, Richard M. Karp, David Peleg, and Douglas B. West. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comput.*, 24(1):78–100, 1995.

[Awe85] Baruch Awerbuch. Complexity of network synchronization. *J. ACM*, 32(4):804–823, 1985.

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th IEEE FOCS*, pages 184–193, 1996.

[Bar98] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *STOC*, pages 161–168, 1998.

[BK06] S. Baswana and T. Kavitha. Faster algorithms for approximate distance oracles and all-pairs small stretch paths. In *International Symposium on Foundations of Computer Science*, pages 591–602, 2006.

[Bol78] Béla Bollobás. *Extremal Graph Theory*. Academic Press, 1978.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms: Second Edition*. The MIT Press, 2001.

[CZ01] E. Cohen and U. Zwick. All-pairs small-stretch paths. *J. Algorithms*, 38(2):335–353, 2001.

[Dij59] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[DKP82] Narsingh Deo, M.S. Krishnomoorthy, and G.M. Prabhu. Algorithms for generating fundamental cycles in a graph. *ACM Transactions on Mathematical Software*, 8(1):26–42, 1982.

[EEST05] Michael Elkin, Yuval Emek, Daniel A. Spielman, and Shang-Hua Teng. Lower-stretch spanning trees. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 494–503. ACM, 2005.

[FRT03] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC*, pages 448–455. ACM, 2003.

[HM94] D. Hartvigsen and R. Mardon. The all-pairs min cut problem and the minimum cycle basis problem on planar graphs. *Journal of Discrete Mathematics*, 7(3):403–418, 1994.

[Hor87] Joseph Douglas Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, 1987.

[KMM07] Telikepalli Kavitha, Kurt Mehlhorn, and Dimitrios Michail. New approximation algorithms for minimum cycle bases of graphs. In Wolfgang Thomas and Pascal Weil, editors, *STACS 2007*, volume 4393 of *Lecture Notes in Computer Science*, pages 512–523. Springer, 2007.

[KMMP04] Telikepalli Kavitha, Kurt Mehlhorn, Dimitrios Michail, and Katarzyna E. Paluch. A faster algorithm for minimum cycle basis of graphs. In Josep Diaz, Juhani Karhumäki, Arto Lepistö, and Donald Sanella, editors, *ICALP*, volume 3142 of *Lecture Notes in Computer Science*, pages 846–857. Springer, 2004.

[Lie03] Christian Liebchen. Finding short integral cycle bases for cyclic timetabling. In Giuseppe Di Battista and Uri Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*, pages 715–726. Springer, 2003.

[MM07] Kurt Mehlhorn and Dimitrios Michail. Minimum cycle bases: Faster and simpler. Kurt Mehlhorn's List of Publications 196, Max-Planck-Institute Saarbrücken, 2007.

[Sey95] P. D. Seymour. Packing directed cycles fractionally. *Combinatorica*, 15(2):281–288, 1995.

[Tar75] R. E. Tarjan. Efficiency of a good but not linear set union algorithm. *J. of ACM*, 22(2):215–225, 1975.

A A Construction of HST

In this appendix we provide a sketch of an explicit proof of Theorem 3, and argue that for an m-edge n-vertex weighted graph (G,ω) the HST with the desired properties can be constructed in $O(m\cdot\log n+n\cdot\log^2 n)$ time. Our argument is based on a variant of the construction of [EEST05]. The latter construction provides a spanning tree with an average stretch $O(\log^2 n\log\log n)$, and its running time is $O(m\cdot\log n+n\cdot\log^2 n)$. We show that one can use it to obtain an HST with an average stretch (in the sense of Theorem 3) $O(\log n\log\log n)$, while maintaining the running time of the construction of [EEST05].

Another crucial property of our construction is that the obtained HST guarantees an $O(\log n\log\log n)$ bound with respect to *strong* internal diameters. More specifically, let $diam(G[U])=\max\{d_{G[U]}(u,v)\mid u,v\in U\}$, and $diam_G(U)=\max\{d_G(u,v)\mid u,v\in U\}$. Consider a cluster U of the HST, an edge $e=(u,v)$ in a cut between two sub-clusters $W_1,W_2\subseteq U$. (The clusters W_1,W_2 are children of U in the HST T.) Then our construction guarantees that $d_{HST}(u,v)\geq diam(G[U])$. This is in contrast to a relaxed requirement that $d_{HST}(u,v)\geq diam_G(U)$ guaranteed by the construction in [FRT03]. This relaxed requirement is not sufficient for our purposes.

The main procedure in our approach is the recursive Algorithm 4 `HST`. First, it invokes Algorithm 3 `Decomp` in order to obtain a partition of the vertices of its input graph, the clusters. After the recursion is completed, the HST is constructed by creating a new artificial node in the HST, which represents the input graph of this call to `HST`, and linking this root node to the root nodes of the subclusters, for which `HST` was called recursively. In Algorithm 3 `Decomp` the decomposition itself is computed by the repetitive growing technique. In particular, each cluster is the result of a call to Algorithm 2 `BallCut`, which keeps growing a ball around a given input core cluster until its boundary becomes sufficiently sparse.

We first need to introduce some definitions and notation from [EEST05].

The *boundary* of a vertex set $S\subseteq V$, denoted $\partial(S)$, is the set of edges with exactly one endpoint in S.

The *volume* of a set F of edges, denoted $\mathrm{vol}(F)$, is the size of the set $|F|$.

The *cost* of a set F of edges, denoted $\mathrm{cost}(F)$, is the sum of the reciprocals of the weights of the edges in F.

The *volume* of a set S of vertices, denoted $\mathrm{vol}(S)$, is the number of edges with at least one endpoint in S.

The *ball* of radius r around a vertex set S, denoted $B(S,r)$, is the set of vertices of distance at most r from S.

We next describe the Procedure `BallCut` that enables to partition a graph into clusters with small diameters and small-cost boundaries. This procedure employs the standard ball-growing technique introduced by Awerbuch [Awe85].

Taken as input a graph G, a core cluster C_0, and a radius ρ, Procedure `BallCut` returns a cluster C that contains C_0, and such that $Rad(C, C_0) = \max\{d_{G(C)}(u, w) \mid u \in C, w \in C_0\} \leq \rho$. Moreover, the boundary of the cluster C has a small cost. The fulfilment of all these requirements is possible borrowing from the spirit and the main ingredients of the Procedure `ConeCut` of [EEST05]. The pseudo-code of Procedure `BallCut` is here below.

Algorithm 2 `BallCut`

Require: A weighted input graph (G, w), a connected cluster $C_0 \subseteq V$, and a positive parameter ρ.

Ensure: A connected cluster C s.t. $C_0 \subseteq C$, $Rad(C, C_0) \leq \rho$, and $\mathrm{cost}(\partial(C))$ is small.

if $\mathrm{vol}(E(C_0)) = 0$ **then**
  $\mu \leftarrow (\mathrm{vol}(C_0) + 1) \cdot \log(m + 1)$;
else
  $\mu \leftarrow \mathrm{vol}(C_0) \cdot \log\left(\frac{m}{\mathrm{vol}(E(C_0))}\right)$;
end if
$r \leftarrow 0$;
while $\mathrm{cost}(\partial(B(C_0, r))) > \mu/\rho$ **do**
  let $w \notin B(C_0, r)$ be the closest vertex to $B(C_0, r)$;
  $r \leftarrow r + d_G(B(C_0, r), w)$;
end while
return $B(C_0, r)$.

The properties of Procedure `BallCut` are summarized in the following lemma.

Lemma 10 *Let $G = (V, E, \omega)$ be a connected weighted graph, let $C_0 \subseteq V$ be a connected vertex set, and let ρ be a number, $0 < \rho \leq Rad_G(C_0)$. Then* `BallCut`(G, C_0, ρ) *returns a connected set C of vertices, $C_0 \subseteq C$, such that $Rad_G(C, C_0) \leq \rho$, and*

$$cost(\partial(C)) \;\leq\; \frac{\mathrm{vol}(C) + \tau}{\rho} \cdot \max\left\{\log \frac{m + \tau}{\mathrm{vol}(E(C_0)) + \tau}\right\}$$

with $m = |E|$, and

$$\tau \;=\; \begin{cases} 1, & \text{if } \mathrm{vol}(E(C_0)) = 0 \\ 0, & \text{otherwise.} \end{cases}$$

This lemma follows directly from Lemma 4.2 (Concentric System Cutting) of [EEST05], and from the fact that the collection of balls is a concentric system. (See [EEST05] for the definition of concentric systems.)

We next describe Procedure `Decomp`. This procedure is used to form a partition $V_1, \ldots, V_k$, for some positive integer k, of a given cluster $G' = (U, G(U), \omega)$. The procedure accepts as input the graph G'. In addition, it accepts as input the parameter Δ which determines the maximum radii of the resulting clusters $V_1, \ldots, V_k$. It also accepts as input two auxiliary parameters. One of them is

$\hat{m}$, which is the number of edges in the original graph $G = (V, E, \omega)$. (The HST is constructed for G, and $\hat{m} = |E(G)|$. We also denote $\hat{n} = |V|$.) The other parameter is t, and it controls the number of levels of the *repetitive growing*. (See below and [EEST05].) This parameter will be set as $\log\log\hat{m}$ to optimize the resulting bound on the $\sum_{e\in E} d_{HST}(e)$.

Each iteration of the external while-loop corresponds to construction of a single cluster V_j. Once a cluster is constructed, it is removed from the graph, and the next iteration starts. The loop continues as long as the residual graph is not empty.

The inner while-loop constructs one single cluster, and it employs the repetitive growing technique. It runs for at most t iterations. On each iteration it attempts to grow the cluster to an additional radius of Δ/t, and thus overall, the radius of the resulting cluster V_j with respect to its core vertex x_j is at most Δ. On each iteration the algorithm tests whether the volume of the edge set of the cluster is sufficiently small, and leaves the loop if this is the case. Recall that Procedure `BallCut` guarantees that the boundary of the constructed cluster has low cost. Note also that the larger is the core input set $K = C_0$ to Procedure `BallCut`, the smaller is the bound provided by Lemma 10 on the cost of the boundary. This observation plays a key role in the analysis of the repetitive growing technique.

The pseudo-code of Procedure `Decomp` is given below. This procedure is based on the Procedure `ImpConeDecomp` of [EEST05].

Algorithm 3 `Decomp`

Require: A cluster G', a number $\Delta > 0$, $t = 1, 2, \ldots$, and the number of edges $\hat{m}$ of the original graph G.

Ensure: A partition $\{V_1, \ldots, V_k\}$, for some $k = 1, 2, \ldots$ of G' with clusters of diameter at most Δ and low-cost set of inter-cluster edges.

$j \leftarrow 0$; $U \leftarrow V'$; $G_0 \leftarrow G$;
while $U \neq \emptyset$ **do**
  $j \leftarrow j + 1$; $p \leftarrow t - 1$;
  Let x_j be a vertex of G_{j-1};
  $K \leftarrow \{x_j\}$;
  while $p > 0$ **do**
    $C \leftarrow$ `BallCut`$(G_{j-1}, K, \Delta/t)$;
    if $\mathrm{vol}\,(E(C)) \leq \frac{m}{2^{\log^{p/t}\hat{m}}}$ **then**
      exit the loop;
    else
      $p \leftarrow p - 1$; $K \leftarrow C$;
    end if
  end while
  $V_j \leftarrow C$; $G_j \leftarrow G(V \setminus \bigcup_{i=1}^{j} V_i)$;
end while
return $\{V_1, \ldots, V_j\}$.

The properties of the partition constructed by Procedure `Decomp` are summarized in the following lemma.

Lemma 11 *For an n-vertex m-edge graph G, the invocation $\{V_1,\ldots,V_k\} = \texttt{Decomp}(G,\Delta,t,\hat{m})$, in time $O(m+n\cdot\log n)$, returns a partition that satisfies that for every index $j\in\{1,2,\ldots,k\}$ there exists $p=p(j)\in\{0,1,\ldots,t-1\}$ such that*

$$cost\left(E\left(V_j, V-\cup_{i=0}^{j}V_i\right)\right) \leq t\cdot\frac{2\,\mathrm{vol}\,(V_j)\log^{(p+1)/t}(\hat{m}+1)}{\Delta}\ , \tag{11}$$

and unless $p=0$,

$$\mathrm{vol}\,(E(V_j)) \ \leq\ \frac{m}{2^{\log^{p/t}\hat{m}}}\ . \tag{12}$$

The proof of this lemma is analogous to that of Lemma 5.1 (Improved Low-Cost Star Decomposition) of [EEST05].

Finally, we use Procedure `Decomp` to construct the HST with the desired properties. Procedure `HST` accepts as input a connected graph $G=(V,E,\omega)$, and returns the HST T, rooted at a node x, denoted by (T,x), for short. Notice that the node x does *not* belong to the vertex set V. The termination condition of the recursion is the case in which the input set consists of at most two vertices. If it consists of just one single vertex, then this vertex is returned as the resulting HST. If it consists of two vertices v,w then, since G is connected, it contains an edge $e=(v,w)$ between these vertices. Then the algorithm returns a tree rooted at x with two edges (x,v), (x,w), both of weight $\omega(e)/2$.

The algorithm invokes Procedure `Decomp` to form a partition $\{\tilde{V}_1,\ldots,\tilde{V}_k\}$ of the graph G. It then recurses in each of the clusters of this partition to form HSTs $(T_1,x_1),\ldots,(T_k,x_k)$ for these clusters. Finally, it connects x to each root x_i of T_i, $i=1,2,\ldots,k$, via edges of weight $diam(G)/4$. (This description ignores the issue of contracting short edges. This issue is handled exactly like in [EEST05].)

The pseudo-code of Procedure `HST` is given below. It is based on the Procedure `LowStretchTree` from [EEST05].

The following theorem summarizes the properties of Procedure `HST`.

Theorem 12 *Let $G=(V,E,\omega)$ be a connected weighted graph with $W=\sum_{e\in E}\omega(e)$. Then $(T,x)=\texttt{HST}(G)$, in time $O(\hat{m}\cdot\log\hat{n}+\hat{n}\cdot\log^2\hat{n})$, returns a spanning tree of G that satisfies $\sum_{e\in E}\frac{d_{HST}(e)}{\omega(e)}=O(\log n\log\log n\cdot\hat{m})$.*

The proof of this theorem follows the lines of the proof of Theorem 5.2 (Lower-Stretch Spanning Tree) of [EEST05].

Finally, we derive the inequality $\sum_{e\in E}d_{HST}(e)=O(\log n\log\log n\cdot W)$ in the following way.

Given a graph $G=(V,E,\omega)$, construct a multi-graph $\check{G}=(V,\check{E},\omega)$ in which each edge $e\in E$ is replicated $\omega(e)$ times with the same weight. (The argument above applies to multi-graphs in which edges are allowed to admit fractional

Algorithm 4 `HST`

Require: A weighted graph $G = (V, E, \omega)$.
Ensure: An HST (T, x) of G with an average stretch $O(\log n \log\log n)$.

1: **if** $V = \{v\}$ **then**
2: return(v);
3: **else if** $V = \{v, w\}$ **then**
4: return$((x, v), (x, w)$ of weight $diam(G)/2$ each);
5: **end if**
6: $\rho \leftarrow diam(G)$;
7: Let $\tilde{G} = (\tilde{(V)}, \tilde{(E)})$ be the graph obtained by contracting all edges in G of length less than $\rho/\hat{n}$;
8: $\{\tilde{V}_1, \ldots, \tilde{V}_k\} \leftarrow$ `Decomp`$(\tilde{G}, diam(\tilde{G})/4, t = \log\log\hat{n}, \hat{m})$;
9: For each i, let V_i be the preimage under the contraction of step 7 of vertices in $\tilde{V}_i$;
10: **for** $1 \le i \le k$ **do**
11: $(T_i, x_i) \leftarrow$ `HST`$((V_i, E(V_i), \omega))$;
12: **end for**
13: Assign weight $\rho/2$ to each edge (x, x_i), $i \in [k]$;
14: $T \leftarrow \bigcup_{i=1}^{k} T_i \cup \{(x, x_i) \mid i \in [k]\}$;
15: return (T, x).

multiplicity. Alternatively, one can scale and round all weights while incurring only a constant overhead in the stretch bounds.)

Apply the construction below and Theorem 12 to the multi-graph $\check{G}$. Inspection of the algorithm reveals that the multiplicities do not affect the running time, and thus, in time $O(|E| \cdot \log|V| + |V| \cdot \log^2|V|)$, the algorithm returns an HST for $\check{G}$ that satisfies

$$\sum_{e \in E} \frac{d_{HST}(e)}{\omega(e)} \cdot \omega(e) \;=\; O\left(\log n \log\log n \sum_{e \in E} \omega(e)\right) \;=\; O(\log n \log\log n \cdot W)\ ,$$

proving the desired inequality.

Reports from the group

"Combinatorial Optimization and Graph Algorithms"

of the Department of Mathematics, TU Berlin

2007/22 *Michael Elkin and Christian Liebchen and Romeo Rizzi:* New Length Bounds for Cycle Bases

2007/19 *Nadine Baumann and Sebastian Stiller:* The Price of Anarchy of a Network Creation Game with Exponential Payoff

2007/17 *Christian Liebchen and Michael Schachtebeck and Anita Schöbel and Sebastian Stiller and André Prigge:* Computing Delay Resistant Railway Timetables

2007/03 *Christian Liebchen and Gregor Wünsch and Ekkehard Köhler and Alexander Reich and Romeo Rizzi:* Benchmarks for Strictly Fundamental Cycle Bases

2006/32 *Romeo Rizzi and Christian Liebchen:* A New Bound on the Length of Minimum Cycle Bases

2006/24 *Christian Liebchen and Sebastian Stiller:* Delay Resistant Timetabling

2006/08 *Nicole Megow and Tjark Vredeveld:* Approximation Results for Preemptive Stochastic Online Scheduling

2006/07 *Ekkehard Köhler and Christian Liebchen and Romeo Rizzi and Gregor Wünsch:* Reducing the Optimality Gap of Strictly Fundamental Cycle Bases in Planar Grids

2006/05 *Georg Baier and Thomas Erlebach and Alexander Hall and Ekkehard Köhler and Heiko Schilling:* Length-Bounded Cuts and Flows

2005/30 *Ronald Koch and Martin Skutella and Ines Spenke :* Maximum k-Splittable Flows

2005/29 *Ronald Koch and Ines Spenke :* Complexity and Approximability of k-Splittable Flows

2005/28 *Stefan Heinz and Sven O. Krumke and Nicole Megow and Jörg Rambau and Andreas Tuchscherer and Tjark Vredeveld:* The Online Target Date Assignment Problem

2005/18 *Christian Liebchen and Romeo Rizzi:* Classes of Cycle Bases

2005/11 *Rolf H. Möhring and Heiko Schilling and Birk Schütz and Dorothea Wagner and Thomas Willhalm:* Partitioning Graphs to Speed Up Dijkstra's Algorithm.

2005/07 *Gabriele Di Stefano and Stefan Krause and Marco E. Lübbecke and Uwe T.Zimmermann:* On Minimum Monotone and Unimodal Partitions of Permutations

2005/06 *Christian Liebchen:* A Cut-based Heuristic to Produce Almost Feasible Periodic Railway Timetables

2005/03 *Nicole Megow, Marc Uetz, and Tjark Vredeveld:* Models and Algorithms for Stochastic Online Scheduling

2004/37 *Laura Heinrich-Litan and Marco E. Lübbecke:* Rectangle Covers Revisited Computationally

2004/35 *Alex Hall and Heiko Schilling:* Flows over Time: Towards a more Realistic and Computationally Tractable Model

2004/31 *Christian Liebchen and Romeo Rizzi:* A Greedy Approach to Compute a Minimum Cycle Bases of a Directed Graph

2004/27 *Ekkehard Köhler and Rolf H. Möhring and Gregor Wünsch:* Minimizing Total Delay in Fixed-Time Controlled Traffic Networks

2004/26 *Rolf H. Möhring and Ekkehard Köhler and Ewgenij Gawrilow and Björn Stenzel:* Conflict-free Real-time AGV Routing

2004/21 *Christian Liebchen and Mark Proksch and Frank H. Wagner:* Performance of Algorithms for Periodic Timetable Optimization

2004/20 *Christian Liebchen and Rolf H. Möhring:* The Modeling Power of the Periodic Event Scheduling Problem: Railway Timetables — and Beyond

2004/19 *Ronald Koch and Ines Spenke:* Complexity and Approximability of k-splittable flow problems

2004/18 *Nicole Megow, Marc Uetz, and Tjark Vredeveld:* Stochastic Online Scheduling on Parallel Machines

2004/09 *Marco E. Lübbecke and Uwe T. Zimmermann:* Shunting Minimal Rail Car Allocation

2004/08 *Marco E. Lübbecke and Jacques Desrosiers:* Selected Topics in Column Generation

2003/050 *Berit Johannes:* On the Complexity of Scheduling Unit-Time Jobs with OR-Precedence Constraints

2003/49 *Christian Liebchen and Rolf H. Möhring:* Information on MIPLIB's timetab-instances

2003/48 *Jacques Desrosiers and Marco E. Lübbecke:* A Primer in Column Generation

2003/47 *Thomas Erlebach, Vanessa Kääb, and Rolf H. Möhring:* Scheduling AND/OR-Networks on Identical Parallel Machines

2003/43 *Michael R. Bussieck, Thomas Lindner, and Marco E. Lübbecke:* A Fast Algorithm for Near Cost Optimal Line Plans

2003/42 *Marco E. Lübbecke:* Dual Variable Based Fathoming in Dynamic Programs for Column Generation

2003/37 *Sándor P. Fekete, Marco E. Lübbecke, and Henk Meijer:* Minimizing the Stabbing Number of Matchings, Trees, and Triangulations

2003/25 *Daniel Villeneuve, Jacques Desrosiers, Marco E. Lübbecke, and François Soumis:* On Compact Formulations for Integer Programs Solved by Column Generation

2003/24 *Alex Hall, Katharina Langkau, and Martin Skutella:* An FPTAS for Quickest Multicommodity Flows with Inflow-Dependent Transit Times

2003/23 *Sven O. Krumke, Nicole Megow, and Tjark Vredeveld:* How to Whack Moles

2003/22 *Nicole Megow and Andreas S. Schulz:* Scheduling to Minimize Average Completion Time Revisited: Deterministic On-Line Algorithms

2003/16 *Christian Liebchen:* Symmetry for Periodic Railway Timetables

2003/12 *Christian Liebchen:* Finding Short Integral Cycle Bases for Cyclic Timetabling

Reports may be requested from:

Sekretariat MA 6–1
Fakultt II – Institut fr Mathematik
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

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

Reports are also available in various formats from

`http://www.math.tu-berlin.de/coga/publications/techreports/`

and via anonymous ftp as

`ftp://ftp.math.tu-berlin.de/pub/Preprints/combi/Report-`*number*`-`*year*`.ps`