Strongly-Bounded Sparse Decompositions of Minor Free Graphs

Ittai Abraham
The Hebrew University, Jerusalem Israel
ittaia@cs.huji.ac.il

Cyril Gavoille
Université Bordeaux, France
gavoille@labri.fr

Dahlia Malkhi
Microsoft Research
dalia@microsoft.com

Udi Wieder
Microsoft Research
uwieder@microsoft.com

Abstract

As networks grow large and complex, a key approach in managing information and constructing algorithms is to decompose the network into locality-preserving *clusters*. Then, information and/or management can be divided between the clusters, such that every node is responsible only for clusters for which it belongs. Such decompositions into locality sensitive clusters have become key tools in network and graph theory and span a large body of literature. A key property of a useful decomposition is that each cluster has a short diameter. This property comes in two variants: a *weak* short diameter in which the short path between nodes may leave the common cluster and a *strong* diameter, in which each cluster has a small diameter when considered as an induced sub-graph.

In this work we present sparse covers and sparse partitions with a bounded *strong diameter* for the family of minor-free graphs. The tools we present are useful in a large variety of distributed applications such as compact routing and synchronizers.

Regular Submission

1 Introduction

As networks grow large and complex, a key approach in managing information and constructing algorithms is to decompose the network into locality-preserving *clusters*. Then, information and/or management can be divided between the clusters, such that every node is responsible only for clusters for which it belongs. Such decompositions into locality sensitive clusters have become key tools in network and graph theory and span a large body of literature.

Sparse Covers

A *cover* is a set of subsets (clusters) of graph nodes that jointly cover the entire set of nodes. A cover is *sparse* if a node belongs to some a priori bounded number of clusters. A *Sparse Partition*, introduced by Awerbuch and Peleg in [10], is a graph cover in which a ball around each node is contained in some cluster.

Specifically, consider an undirected weighted graph $G = (V, E, \omega)$ i.e. $E \subseteq V \times V$ and $\omega : E \rightarrow \mathbb{R}^+$. Let $d_G(u, v)$ be the cost of a minimum cost path between u and v where the cost of a path is the sum of weights of its edges. When G is clear from the context we omit the subscript and write $d(u, v)$. Let $\mathrm{diam}(G) = \max_{u,v} d(u, v)$. Given $U \subseteq V$, let $G[U]$ be the induced subgraph whose nodes are U and whose edges are the edges in G whose endpoints both belong to U. Let $B_G(u, r) = \{v \mid d_G(u, v) \leq r\}$. For $A \subseteq V$ and $v \in V$ let $d(A, v) := \min_{u \in A} d(u, v)$.

Definition 1.1. A (k, Δ, τ) *sparse cover* is a set of clusters $\mathcal{C} \subset 2^V$ with the following properties:

1. [Cover]: For every $v \in V$, there exists $C \in \mathcal{C}$ such that $B(v, \Delta) \subseteq C$.
2. [Small strong diameter]: For each $C \in \mathcal{C}$, $\mathrm{diam}(G[C]) \leq k\Delta$.
3. [Sparse cover]: For each $u \in V$, $|\{C \mid u \in C, C \in \mathcal{C}\}| \leq \tau$.

Sparse Partitions are used as a building block in the solution of a variety of applications. These include distance coordinates [7, 19], routing with succinct routing tables [7, 8, 10, 1], mobile user tracking [10] and resource allocation [6], synchronization in distributed algorithms [9], and others.

For general graphs, the seminal construction in [10] provides a $(2k - 1, \Delta, 2kn^{1/k})$ sparse cover. This result asymptotically matches the known lower bound. For certain restricted families of graphs, better covers are known to exist. For example, for graphs with a doubling dimension α, Abraham et al. demonstrate in [20](Lemma 1) a $(2, \Delta, 4^\alpha)$ cover. For a two-dimensional Euclidean space, a $(2, \Delta, 4)$ cover is easily obtain, e.g., see in [3]. We now describe the family of graphs addressed in this paper.

Minor-free graphs. The *contraction* of an edge $e = (u, v)$ is the replacement of nodes u, v with a new vertex whose incident edges are the edges other than e that were incident to u or v. A graph H is a *minor* of G if H is a subgraph of a graph obtained by a series of edge contractions of G. A celebrated theorem of Robertson and Seymour states that every (possibly infinite) set of graphs $\mathcal{G}$ that is closed to edge contractions and edge removals is represented by a finite set of graphs called its *obstruction set*, where a graph G is in the set $\mathcal{G}$ if and only if none of the graphs in the obstruction set is a minor of G. Our present work addresses $K_{r,r}$ free graphs. Note that every minor-free family of graphs is contained in a $K_{r,r}$-free family for a large enough r. For instance, planar graphs are $K_{3,3}$-free.

For minor-free graphs the following bounds are known. A (k, Δ, τ) *weak sparse cover* is defined as above, but the diameter bound is on distances in the original graph. That is, a short path between cluster nodes may exist **outside** the cluster. Abraham et al. provide in [1] a $(O(r^2\rho), \Delta, O(2^{r \log r}))$ weak sparse cover that builds on the principles of a sparse partition by Klein et al. [16]. However, no (strong) sparse cover was known. In fact, one can demonstrate concrete, planar graphs in which the KPR construction yields clusters with arbitrarily high diameter (see Section 2 below). The challenge of providing minor-free graph decompositions whose diameter is strongly bounded remained open, and is addressed by the present work. Our first result is stated in the following theorem:

Theorem 1.2. *For every graph $G = (V, E)$ excluding a $K_{r,r}$ minor, and parameter $\rho > 0$, there exists a polynomial algorithm that constructs a $(O(r^2), \rho, 2^r(r+1)!)$ sparse cover.*

During the preparation of this manuscript, it has been brought to our knowledge that independent work in progress [12] has achieved a $(4, \rho, O(\log n))$ for minor free graphs, and a $(O(1), \rho, O(1))$ sparse cover for planar graphs.

Compact Routing

As mentioned above, many applications of sparse covers are known. We highlight one in particular in this paper, the classical problem of compact, loop-free routing. In this problem, we consider a distributed network of nodes connected via a network in which each node has an arbitrary network identifier. A routing scheme allows any source node to route messages to any destination node, given the destination's network identifier. The fundamental trade-offs in compact routing schemes is between the *space* used to store the routing table on each node and the *stretch* factor of the routing scheme, the maximum ratio over all pairs between the length of the route induced by the scheme and the length of a shortest-path between the same pair.

In this paper we assume a network with arbitrary node names. This model is called the *name-independent* model because the designer of the routing scheme has no control over node names. So node names cannot encode any topological awareness, like for instance the X, Y-coordinates in a geographic network. A weak variant of this fundamental problem is called *labeled routing*. In this version of the problem, the designer of a solution may pick node names that contain (bounded size) information about their location in the network. This variant is useful in many aspects of network theory, but less so in practice: Knowledge of the labels needs to be disseminated to all potential senders, as these labels are not the addresses by which nodes of an *existing* network, e.g., an IP network, are known. Furthermore, if the network may admit new joining nodes, all the labels may need to be re-computed and distributed to any potential sender. Finally, various recent applications pose constraints on nodes addresses that cannot be satisfied by existing labeled routing schemes. E.g., Distributed Hash Tables (DHTs) require nodes names in the range $[1, n]$, or ones that form a binary prefix.

Abraham et al. provide in [1] the following result. For every n-node unweighted graph excluding a fixed $K_{r,r}$ minor, they prove the existence of a polynomial time constructible name-independent routing scheme with constant stretch factor, in which every node v requires routing tables of $\tilde{O}(1)$ bits and $O(\log^2 n / \log\log n)$-bit headers. For planar graphs (i.e., $H = K_{3,3}$-free), a $\tilde{O}(1)$ memory $1+\epsilon$ stretch labeled routing scheme is given by Thorup in [18]. This result cannot be extended to name-independent scheme since in that case a stretch of 3 at least is required if less than $\Omega(n \log n)$ bits are used [5].

Our next contribution is stated in the following theorem.

Theorem 1.3. *For every n-node unweighted graph of diameter D excluding a $K_{r,r}$ minor, there is a polynomial time constructible name-independent routing scheme, in the fixed port model[1], with stretch $O(r^2)$ and using $O(\log n)$-bit headers, in which every node requires tables of $O(r!2^{O(r)} \cdot \log D \cdot \log^3 n / \log\log n)$ bits.*

[1]I.e., the port number around each node u is an arbitrary permutation of $\{1, \ldots, \deg(u)\}$ that, as well as the node names, cannot be changed during the design of the routing scheme.

Probabilistic Sparse Partitions

Another important approach to decomposing removes a set of edges which is not too large, and consider the resulting graph partition. More precisely, we have the following definitions.

Definition 1.4. A *strong diameter* Δ *bounded partition of* G is a partition of V into clusters $C_1, C_2, \ldots$ such that for each cluster C, $\operatorname{diam}(G[C]) \leq \Delta$. Given a partition P and a node u, let $P(u)$ be the unique cluster that contains u.

Definition 1.5. A Δ *bounded,* η *padded strong diameter probabilistic partition* is a distribution $\mathcal{P}$ on partitions with the following properties:

1. [Small diameter]: Each $P \in \mathcal{P}$ is a strong diameter Δ bounded partition of G.
2. [Small probability of cutting an edge]: For any $u, v \in V$,

$$\Pr_{P \sim \mathcal{P}}[P(u) \neq P(v)] \leq d(u,v)\eta \ .$$

These sparse partitions play key role in approximation algorithms, such as multi-commodity flow optimization problems [16]. Klein et al. provide a sparse partition for graphs excluding $K_{r,r}$, whose diameter is weakly bounded by $O(r^3\rho))$, while the probability of cutting an edge is $O(1/\rho)$, for any parameter of choice $\rho > 0$. Fakcharoenphol and Talwar improve in [14] to a weak diameter bound of $O(r^2\rho)$. However, both of these decompositions provide a weak diameter bound only.

Our improvement is to provide a strong diameter bound, as follows.

Theorem 1.6. *Let G be an undirected weighted graph excluding a $K_{r,r}$ minor. Let $\rho \leq \operatorname{diam}(G)$, then there exists a polynomial time sampleable $O(r6^r\rho)$ bounded, $O(r/\rho)$ padded strong diameter probabilistic partition of G. Furthermore, it is possible to find in polynomial time a partition with strong diameter $O(r6^r\rho)$, where the total weight of edges crossing the partition is at most $O(\sum_{(u,v)\in E} d_G(u,v)/\rho)$.*

We envision that this result may play an important role in further optimization problems and graph embeddings into trees.

1.1 Summary of Contributions

In summary, the paper provides the following results for any $K_{r,r}$-free undirect weighted graph.

- For any choice of a parameter ρ, there is a polynomially constructible sparse cover of the radius-ρ balls around every node, such that each cluster in the cover has diameter at most $O(r^2\rho)$, and every node belongs to at most $O2^{O(r\log r)}$ clusters.
- There is a polynomially constructible name-independent routing scheme with $O(r^2)$ stretch and tables of size $O(r!2^{O(r)} polylog(n)$ bits.
- There is a polynomially constructible sparse partition with where for every $\rho > 0$ the probability an edge (u,v) is cut is $\frac{r \cdot d(u,v)}{\rho}$ and the *strong* diameter of each cluster is at most $r6^r\rho$

2 Large Diameter KPR Clusters

In this section, we briefly review the KPR [16] construction, and exemplify its unbounded diameter.

KPR performs three recursive tree cuts into *stripes* of height ρ. Each tree is a BFS of a connected component that is left from the previous cut.

More precisely, initially start with G_0, the whole graph. Select an offset $H \in [\rho..2\rho - 1]$ uniformly at random. Build a BFS tree T_1 on G_0, rooted at an arbitrary node r_1. Slice T_1 into stripes of height ρ: The i'th stripe contains nodes whose BFS distance from the root of T_1 is between $H + (i-1)\rho$ and $H + i\rho - 1$. Recurse on any connected component G_1 contained within any stripe.

Figure 1 depicts a simple, planar graph in which the KPR cut results in a final cluster whose strong-diameter is arbitrarily large. The picture also shows that adding more iterations of KPR-cuts does not remedy the situation. We now explain the example in detail.

In the original graph, pivot node p_1 has distance less than ρ to all the nodes below the cut line marked '$G1$ cut'. It has a path of length (about) H to r_1. Hence, the BFS tree T_1 rooted at r_1 may perform the cut marked '$G1$ cut' and leave G_1 as the connected component containing all nodes under the cut. Note that, despite the fact that nodes r_2 and p_2 have small distance in G_0 (going through p_1, for example), their distance in G_1 is large.

Continuing on, we build the next steps in a similar manner. The tree cut of T_2 might perform the cut marked as '$G2$ cut', leaving all nodes below it as G_2. And so on. Finally, the distance in G_3 between the nodes marked as x and y have arbitrarily large distance: All the pivot nodes p_1, p_2 and p_3 that shorten the distance from x to y have been cut away from the cluster.

3 Sparse Cover with Strong Diameter

We now show how to remedy the problem depicted in Figure 1 above. We provide a graph cover procedure that yields clusters whose diameters are strongly-bounded. The construction builds on the partition of Abraham et al. in [1], and enhances it in order to obtain a strong diameter constraint.

3.1 The Strong Diameter Cover

The main result of this section is stated in the following theorem.

Theorem 1.2. *For every graph $G = (V, E)$ excluding a $K_{r,r}$ minor, and parameter $\rho > 0$, there exists a polynomial algorithm that constructs a $(O(r^2), \rho, 2^r(r+1)!)$ sparse cover.*

The rest of the section is devoted to the proof of Theorem 1.2. We first describe a *partitioning algorithm* of a graph G that depends on a parameter $\rho > 0$. It returns a cluster H and implicitly graphs G_i, H_i, for $i \in R$.

Actually, in order to create a partition of the graph into clusters, we apply this algorithm for all possible choices of indices $j_i \geq 0$.

Partitioning algorithm:

Initially $H_1 = G_1 = G$. Given G_i and H_i set a root $\tau_i \in H_i$ and choose an index j_i. Let $T_i = BFS(G_i, \tau_i)$. Let H_{i+1} be a connected component of the subgraph graph induced by the nodes

$$H_i \cap C(T_i, j_i)$$

Let G_{i+1} be the connected component of

$$\{v \in \bigcup_{\ell \in \{j_i - i, \ldots, j_i, \ldots, j_i + i\}} C(T_i, \ell) \mid d(v, H_i) \leq i * \rho\}$$

that contains H_{i+1}. If $i = r$ then return $H = G_{i+1}$ and stop. Otherwise repeat.

As an intuition, in Figure 2 below we revisit the graph used to exemplify the large diameter in KPR (Figure 1). Suppose that in our construction, the bottom nodes are included in the final cluster's 'core' H_{r+1}. Then having survived the layer cut in T_1, there must be nodes within distance at most ρ from H_{r+1} through which H_{r+1} is reachable in T_1 (likewise, there are nodes in T_2 within distance 2ρ through which H_{r+1} is reachable, and so on). In the case depicted in the figure, these are the 'pivot' nodes p_2, p_3, etc. These pivot nodes are all included in the final cluster G_{r+1} since any node at distance ρ from H_{r+1} is included in it.

More generally, these might be multiple such pivot nodes. The more careful argument (see analysis below) makes use of the fact that for every r nodes in H_{r+1}, and at tree T_i, there must be r paths through which the r nodes are reachable from T_i's root. That all of these paths are disjoint within G_{r+1} would violate violate the $K_{r,r}$ exclusion property; hence, some of these path are in fact connected.

3.2 Analysis

For the analysis it will be convenient to use the following notation: for all $u \in G_i$ and $i \in R$, let $\text{tail}_i(u) = \text{tail}(T_i, u) \cap G_{i+1}$.

Lemma 3.1. *Let x be any node in $\cap_{1 \leq i \leq r} H_i$. Then $\forall 1 \leq i \leq r : Tail_j(x) \subseteq G_i$.*

Proof. For $i \leq j$, the claim clearly holds since $T_j \subseteq G_j \subseteq G_{j-1} \subseteq \ldots \subseteq G_1$. We now prove the claim for $i > j$. By construction, we have that the length of $Tail_j(x)$ is at most $(j+1)\rho$. Hence, the distance of any node $t \in Tail_j(x)$ from x in T_i satisfies $d_{T_i}(t, x) \leq (j+1)\rho \leq i * \rho$. Therefore, t is included in G_i. $\square$

Lemma 3.2. *Let H be the graph returned after the partitioning algorithm and assume that $j_i > r+1$ for each $i \in R$. Let $X \subseteq H_r$ be any r nodes of H_r. If for each $x \neq x' \in X$ and $i, i' \in R$ the tails $\text{tail}_i(x)$ and $\text{tail}_{i'}(x')$ are pairwise disjoint then G contains a $K_{r,r}$ minor.*

Proof. Such a minor is composed of r sets called the *left super-nodes*, denoted as $B_{x_1}, B_{x_2}, \ldots, B_{x_r}$, such that $\forall x \in X, x \in B_x$; and from r sets called the *right super-nodes*, denoted $B_1, B_2, \ldots, B_r$. Each super-node is a connected sub-graph of G_1, all sets are disjoint and there is an edge connecting each set from the first group with a each set from the second group.

The left super-nodes: For each $x \in X$, we set $B_x = \bigcup_{i \in R} Tail_i(x)$. Observe that each such super node indeed induces a connected subgraph and that these super nodes are pairwise disjoint by the assumption.

The right super-nodes: For all $i \in R$ let U_i denote the subtree of T_i formed by the paths from each $x \in X$ to τ_i the root of T_i. The right super nodes will be the V_i for $i \in R$, where $V_i = U_i \setminus G_{i+1}$. Observe that each V_i induces a connected subtree of G.

The super edges will be the edges in T_i connecting each V_i with each $\text{tail}_i(x) \in B_x$ for each $i \in R, x \in X$. Since all tails, for distinct $x \in X$, are disjoint, we are left only with showing that all V_i's are pairwise disjoint and disjoint from all the tails.

Let $1 \leq j < i \leq r$. By construction, $B_i \subseteq G_i$. Also, by construction, $B_j \subseteq (G_j \setminus G_{j+1})$. Since $G_{j+1} \supseteq G_i$, it follows that B_i and B_j are disjoint.

We now show that B_x is disjoint from B_i, for all $x \in X$, $i \in R$. From Lemma 3.1, we obtain that $B_x \subseteq G_i$. Once again, by construction, $B_i \subseteq G_i \setminus G_{i+1}$ and hence, B_i is disjoint from B_x. □

Lemma 3.3. *If G has no $K_{r,r}$ minor and $j_i > r+1$ for all $i \in R$ then H has strong diameter $4(r+1)^2\rho$.*

Proof. The proof need to show every two nodes of H have distance at most $4(r+1)^2\rho$ within H.

Suppose not, and let x, y be a pair of nodes in H such that $d_H(x,y) > 4(r+1)^2\rho$. Then we can find r nodes $x_1, x_2, ..., x_r$ satisfying (1) $d_H(x_i, x_j) > 2(r+1)\rho$, for all $i \neq j$, and (2) $x_i \in H_{r+1}$. Indeed, it is easy to partition the shortest path from x to y into $r+1$ segments of length $4(r+1)\rho$ each. Call the endpoints of the segments $y_1, y_2, ..., y_{r+1}$. By construction, for each y_k there is a node $x_k \in H_{r+1}$ within distance $(r+1)\rho$. It follows that the distance from x_k to x_{k+1} is at least $2(r+1)\rho$, and the x_k's maintain properties (1) and (2) as needed.

However, by Lemma 3.2, the tails of some pair x_i, x_j must be connected. Since tails have length at most $(r+1)\rho$, we get $d_H(x_i, x_j) \leq 2(r+1)\rho$. A contradiction. □

Proof of Theorem 1.2. Creating the cover is done by running the partition algorithm in the following manner. Note that the output of the partition algorithm (a cluster H and implicitly for each $i \in R$ graphs G_i, H_i) depends only on the choice of the roots $\tau_1, \ldots, \tau_t$ of the trees, the indices $j_1, \ldots, j_t$, and the choice of connected components $H_1, \ldots, H_t$. We fix a consistent choice of roots. Each time a root τ_i is to be chosen from subgraph H_i, we choose it as the node with minimal lexicographic order among the nodes in H_i.

The sets $\mathcal{H}_\rho$ consist of all the clusters H for all possible choices of connected components and all possible choices of indices $j_1, \ldots, j_r$ such that for each $i \in R$ either $j_i \in \mathbb{N}$ or $j_i + 1/2 \in \mathbb{N}$.

Property 1. *(Cover).* It follows from the simple observation that for any $v \in G$ and tree BFS tree T spanning G with root τ there must exist some integer $j \in \mathbb{N}$ such that either $B_G(v, \rho/4) \subset C(T, j)$ or $B_G(v, \rho/4) \subset C(T, j+1/2)$. Then, given any v, we construct by induction a sequence of indices $j_1, \ldots, j_i$ such that $B_G(v, \rho/4) \subseteq H_{i+1}$. Thus, when i reaches r, the procedure returns a cluster $H \in \mathcal{H}_\rho$ containing $B_G(v, \rho/4)$.

Property 2. *(Sparse clusters).* For each graph G_i that v belongs to, it belongs to at most $2 \times (2i+1)$ graphs G_{i+1} due to the use of $2i+1$ stripes in the definition of G_{i+1}, and the fact that we build two BFS trees (one offset by $rho/2$) over each G_i. Hence for each $i \in R$, by simple induction, a node belongs to at most $\prod_{1 \leq j \leq i} 2 \times (2j+1) \leq 2^{2i}(i+1)!$ graphs G_i. Therefore a node belongs to at most $2^{O(r)}(r+1)!$ clusters $H = G_{r+1}$.

Property 4. *(Strong diameter).* When $j_i > r+1$ for each $i \in R$ this follows directly from Lemma 3.2. If there is some $i \in R$ for which $j_i \leq r+1$ then any two nodes in H are tail-connected via tree T_i.

□

4 Name Independent Routing

In this part we consider the problem of routing messages between any pair of nodes of an undirected graph G with precomputed compact routing tables. The performances of the routing scheme for G is measured in term of the size of the local routing tables and the maximum *stretch*, i.e., the ratio between the length of the route from x to y and the minimum possible route length, $d_G(x,y)$.

We concentrate our attention on *name-independent* routing schemes, that is node names cannot be relabeled in order to optimize the routing tables. Indeed, *labeled* routing scheme of stretch $1+\epsilon$ and with polylog routing tables, labels, and headers are known for every weighted graph excluding a fixed minor [2], whereas any name-independent routing scheme on unweighted stars (depth one trees, so excluding K_3) requires a stretch at least 3 if less than $\Omega(n \log n)$ bits per node are used [5].

We assume that G is unweighted, since it has been proved in [4] that there are stars with edge cost 1 or k for which every name-independent routing scheme of stretch $< 2k+1$ requires routing tables of $\Omega((n \log n)^{1/k})$ bits, for every integer $k \geq 1$.

In the remaining of the paper, we will assume that the n node names of G range arbitrary in $\{1, \dots, n^{O(1)}\}$, i.e., are on $O(\log n)$ bits. The scheme extends easily to longer names by the use of hashing techniques.

Thanks to Theorem 1.2 we can show:

Theorem 1.3. *For every n-node unweighted graph of diameter D excluding a $K_{r,r}$ minor, there is a polynomial time constructible name-independent routing scheme, in the fixed port model*[2]*, with stretch $O(r^2)$ and using $O(\log n)$-bit headers, in which every node requires tables of $O(r!2^{O(r)} \cdot \log D \cdot \log^3 n / \log\log n)$ bits.*

The proof of Theorem 1.3 is deferred to Appendix A.

5 Probabilistic Sparse Partition

Let $K_{r,r}$ be the complete bipartite graph with r nodes in each set.

Theorem 1.6. *Let G be an undirected weighted graph excluding a $K_{r,r}$ minor. Let $\rho \leq \mathrm{diam}(G)$, then there exists a polynomial time sampleable $O(r6^r\rho)$ bounded, $O(r/\rho)$ padded strong diameter probabilistic partition of G. Furthermore, it is possible to find in polynomial time a partition with strong diameter $O(r6^r\rho)$, where the total weight of edges crossing the partition is at most $O(\sum_{(u,v)\in E} d_G(u,v)/\rho)$.*

Our proof strategy focuses on an unweighted graph first. Later, in Section 5.3, we detail the reduction from the weighted case to this construction. Henceforth, assume G in unweighted.

We describe the partition procedure which is to be performed r times. The following describes the k'th iteration step by step where $1 \leq k \leq r$.

1. Denote by G_{k-1} the current graph. $G_0 =< V, E >$ is the base of the recursion. Sample two numbers h_k, ℓ_k uniformly and independently from $[0, \dots, \rho - 1]$.
2. Perform BFS from an arbitrary root node $s \in G_{k-1}$: $T_k = BFS(s, G_{k-1})$
3. Divide into layers: $L_k(i) = \{u \mid ib_k\rho + \ell_k \leq d_{G_{k-1}}(u, s) < (i+1)b_k\rho + \ell_k\}$, where $b_k = 6b_{k-1} = 6^k$. We omit i in most cases below, and mention it explicitly only when needed.[3]
4. For each layer $L_k(i)$ let $M(L_k(i))$ be the set of nodes at its middle; i.e. $M(L_k(i)) := \{u \in L_k(i) \mid d_{G_{k-1}}(u, L_k(i-1)) = (b_k/2)\rho$. When $L(i)$ is clear from the context we may abbreviate notation and write M_k.
5. Define a distance function $\gamma_k(\cdot, \cdot)$ on the edges of G_k by giving zero weight to **directed** edges of T_k, and G edge weights to all other edges; i.e.

$$\gamma_k(u,v) = \begin{cases} 0 & (u,v) \text{ is a directed edge of } T_k \\ d_G(u,v) & \text{otherwise} \end{cases}$$

   For two nodes u, v not connected by an edge we extend the definition of the distance $\gamma_k(u, v)$ to be the length of the shortest path from u to v with edges weighted according to γ_k. Note that $\gamma_k(u, v) = 0$ if v is on the T_k sub-tree rooted at u and that $\gamma_k(\cdot, \cdot)$ is *not symmetric*.

[2] I.e., the port number around each node u is an arbitrary permutation of $\{1, \dots, \deg(u)\}$ that, as well as the node names, cannot be changed during the design of the routing scheme.

[3] In KPR this step is taken with $b_k = 1$ for all k

6. Define $S_k^+(i)$ to be the set of nodes within $L_k(i)$ with γ_k distance of at most $\frac{b_k}{2}\rho + h_k$ from $M(L_k(i))$. $S_k^+(i) = \{u \mid u \in L_k(i),\ \gamma_k(M(L_k(i)), u) \le \frac{b_k}{2}\rho + h_k\}$. Note that $S_k^+(i)$ is a set grown around M_k which is at the middle of $L_k(i)$. It includes all the nodes in $L_k(i)$ which have an edge to $L_k(i+1)$ but does not include all of $L_k(i)$.

7. After performing the previous steps to *all* layers in the decomposition, add all **unassigned** nodes from $L(i+1)$ which were not taken by the $S_k^+(i+1)$.
   $S_k(i) := S_k^+(i) \cup \{L_k(i+1) \setminus S_k^+(i+1)\}$

8. Note that now the sets $S_k(i)$ partitions G_{k-1}. For each i recurse on all connected components: $G_k \in CC(S_k(i))$.

Claim 5.1. *In any iteration $1 \le k \le r$ of the algorithm, if $u, v \in G_{k-1}$ then the probability that the kth iteration cuts an edge (u, v) into different clusters is at most $\frac{2}{\rho}$.*

Proof. In each execution of the procedure there are two ways an edge u, v could be cut. The first is that u, v are cut in stage (3); i.e. $u \in L_k(i)$ while $v \in L_k(i+1)$ for some i. The probability the edge is cut by the layers is at most $1/\rho$ due to the randomness of ℓ_k. The second way in which (u, v) might be cut, given $u, v \in L_k(i)$, is if $u \in S_k^+(i)$ and $v \notin S_k^+(i)$. In other words it must be the case that one of the nodes (say w.l.o.g u) has a small γ_k distance from M_k while node v has a large γ_k distance from M_k. Note however that

$$|\gamma_k(M_k, u) - \gamma_k(M_k, v)| \le d_G(u, v)$$

The threshold distance for inclusion in $S_k^+(i)$ is $\frac{b_k\rho}{2} + h_k$ where $h_k \in_R [0, \rho - 1]$. It follows that given that $u, v \in L_k(i)$, the probability (u, v) is cut at most $\frac{1}{\rho}$, which concludes the proof of the lemma. □

There are a r recursive calls (3 in the case of planar graphs) so by the union bound the probability of an edge (u, v) being cut is at most $2r \cdot d_G(u, v)/\rho$.

A note on derandomization The expected total weight of a cut in each iteration k is $2W_k/\rho$ where W_k is the total weight of edges in G_k. There are ρ^2 possibilities for a choice of h_k, ℓ_k. At least one value of h_k, ℓ_k yields a partition where the weight of cut edges is at most $2W_k/\rho$. Thus an exhaustive search would yield a cut with this value. If this is done in every recursive call then the total weight of cut edges is at most $2rW_0/\rho$.

We are now left with proving that the diameter of each component is $O(\rho)$. We do this by showing that if there are two nodes in G_r such that the distance between any two of them is greater than $b^*\rho$ (for some constant b^* which depends on r but not on $|G|$) then the graph contains a $K_{r,r}$ minor.

From now one we omit the notation that states which strip we are talking about (the subscript i in the previous section). The following lemma characterizes the properties we will need in order to show the existence of the $K_{r,r}$ minor. Fix some iteration k.

Lemma 5.2. *Each node $u \in G_k$ has an anchor $a_k(u) \in M_k$ such that $a_k(u) \in G_k$ and $d_{G_k}(u, a_k(u)) \le (3b_k/2 + 2)\rho$.*

Proof. Consider the construction of G_k out of G_{k-1}. The node u can be assigned to G_k either in Step (6) or Step (7) of the construction. If it were assigned in Step (6) then there is a node $a(u)$ such that $\gamma_k(a(u), u) \le (\frac{b_k}{2} + 1)\rho$. The shortest path includes at most $\frac{b_k}{2}\rho + (\frac{b_k}{2} + 1)\rho$ edges of T_k which have a γ_k distance of 0, therefore $d_{G_k}(a(u), u) \le (b_k + 1 + b_k/2 + 1)\rho = (3b_k/2 + 2)\rho$.

If u was assigned to G_k in Step (7) then all its parents in the BFS tree were also assigned to G_k, therefore the path to the root of T_k reaches a node in M_k after distance at most $b_k\rho$. □

Lemma 5.3. *Let $u \in M_k$. For every $v \in G_{k-1}$ such that $d_{G_{k-1}}(u,v) \leq b_k\rho/2$ it holds that $v \in G_k$. In other words a ball around u in G_{k-1} of radius $b_k\rho/2$ is contained in G_k.*

Proof. Step (6) above includes in S_k^+ all the nodes at distance $b_k\rho/2$, thus $v \in S_k^+$. The lemma then follows since the path between u and v is contained in S_k. □

5.1 The Super-nodes

Assume there are two nodes $x, y \in G_r$ such that $d_{G_r}(x,y) \geq 12rb_r\rho$. There must be therefore r nodes $x = x_1, x_2, \dots, x_r = y$ in G_r such that $d_{G_r}(x_i, x_j) \geq 12b_r\rho$ for every $i \neq j$. The proof follows the KPR paradigm; i.e. we will have to show that this implies that the graph contains a $K_{r,r}$ minor which contradicts the assumption that G is $K_{r,r}$ free. Such a minor is composed of r sets denoted as $B_1, \dots, B_r$ such that $x_i \in B_i$ and r sets $R_1, \dots, R_r$ such that each set is a connected sub-graph of G, all sets are disjoint and there is an edge connecting B_i and R_j for every i, j. This yields a contradiction since each set could be contracted to a single node creating a $K_{r,r}$ minor.

The Set B_i

The node x_i has an anchor in $a_r(x_i) \in M_r$. Call this node a_r (for brevity we omit the subscript i), and denote by A_r the path between x_i and a_r . The node a_r has an anchor $a_{r-1}(a_r) \in M_{r-1}$. Call this node a_{r-1} and define recursively $a_{j-1} = a_{j-1}(a_j)$, and A_j to be the path between a_{j-1} and a_j.

Let $u \in M_k$. Define $Tail_k(u)$ to be the path in T_k which connects u to the upper boundary of L. In other words $Tail_k(u)$ is a path of length at most $\frac{b_k}{2}$ in T_k starting from u towards the root. Now define:

$$B(k) := \bigcup_{j=1}^{k} A_j \bigcup_{j=1}^{k} Tail_j(a_j)$$

The set B_i is now defined as $B(r)$. Clearly the induced graph $G[B_i]$ is connected. This however turns out not to be enough.

Lemma 5.4. *All the nodes in B_i belong to G_r. Furthermore $Diam_{G_r}(B_i) \leq 3b_r\rho$.*

Proof. We prove that $B(k) \subseteq G_k$ by induction on k. For the base case we have $B(1) = A_1 \cup Tail_1(a_1)$ where $A_1 \subseteq G_1$ by Lemma 5.2. We have that $Tail_1(a_1) \subseteq G_1$ by Lemma 5.3. By the induction hypothesis we have that $B(r-1) \subseteq G_{r-1}$. Furthermore, Lemma 5.2 implies that $A_r \subseteq G_r$ and Lemma 5.3 implies that $Tail_r(a_r) \subseteq G_r$. It remains therefore to show that $B(r-1) \subseteq G_r$. Let $u \in B(r-1)$. By the induction hypothesis $d_{G_{r-1}}(a_r, u) \leq 3b_{r-1}\rho$. We have that $3b_{r-1} \leq \frac{b_r}{2}$ so by Lemma 5.3 $B(r-1) \subseteq G_r$. Furthermore $diam_{G_r} B(r) \leq (3b_r/2 + 1)\rho + 3b_{r-1}\rho \leq 3b_r\rho$. □

The Set R_j

The Set R_j is constructed by pruning the tree T_j at G_j. In other words, $u \in R_j$ if $u \notin G_j$ and there is a node $v \in G_j$ such that u is an ancestor of v in T_j. Clearly the following holds:

Lemma 5.5. *The set R_j has the following properties:*

1. *The induced subgraph $G[R_j]$ is connected.*
2. $R_j \subseteq G_{j-1}$
3. $R_j \cap G_j = \emptyset$

5.2 Putting It All Together

We defined $2r$ sets B_i and R_i. In order to complete the construction of the $K_{r,r}$ minor we have to show the following.

Lemma 5.6. *If there are two nodes x, y in G_r such that $d_{G_r}(x, y) \geq 12rb_r\rho$ then the $2r$ sets of nodes B_i, R_i $i = 1...r$ have the following properties:*

1. *For every i the subgraph $G[B_i]$ and the subgraph $G[R_i]$ are connected.*
2. *The sets B_i, R_i $i = 1...r$ are all mutually disjoint.*
3. *For every i, j there are nodes $u \in B_i$ and $v \in R_j$ such that (u, v) is an edge in G.*

First we show why Lemma 5.6 suffices to prove Theorem 1.6. Since all the sets are connected in G and they are all mutually disjoint, each one of the sets could be contracted into a different single node using only minor operations. Property (3) of the lemma implies that the resulting graph contains a $K_{r,r}$ minor contradicting the fact that G is $K_{r,r}$ free. We conclude that it must be that the *strong* diameter of each G_r is bounded by $12rb_r\rho$.

Proof. The first assertion of the Lemma is immediate from the previous Section.

To see why the third Assertion is true consider two sets R_i, B_j. The set B_j contains the path $Tail_i(a_i)$ which is defined to be a BFS path in T_i. The set R_i is the remaining part of T_i thus the last node in $Tail_i(a_i)$ is connected to a node in T_i.

The involved part is to show all the sets are mutually disjoint. We do it case by case:

First we show that for every $i \neq j$ it holds that $R_i \cap R_j = \emptyset$. Assume w.l.o.g that $i \leq j - 1$. By Lemma 5.5 it holds that $R_i \cap G_{j-1} = \emptyset$ while $R_j \subseteq G_{j-1}$. Conclude that $R_i \cap R_j = \emptyset$.

Second we show that for every i, j it holds that $B_i \cap R_j = \emptyset$. By Lemma 5.5 the set R_j is disjoint from G_r while by Lemma 5.4 the set B_i is contained in G_r.

Finally we show that for every $i \neq j$ it holds that $B_i \cap B_j = \emptyset$. This follows since $x_1, x_2, \ldots, x_r$ are far from one another in G_r, yet each B_i has a small radius in G_r. To be precise, the radius of each B_i is bounded by $3b_r\rho$ while the distance between x_i and x_j is at least $12b_r\rho$. □

5.3 The Weighted Case

We now present the reduction from the weighted graph case to the unweighted construction above. It is first worthwhile illuminating the key aspects of the above construction that are affected by having non-uniform edge weights. First, we need to find a node in M_k, the middle-strip, whenever two connected nodes cross M_k. Second, for two nodes u, v whose distance is larger than r, we need to find r nodes along the shortest path from u to v at distance $d(u, v)/r$ apart. Finally, we need the distances in G to uphold the triangle inequality; without it, the construction above may not yield the required cluster diameter bound.

We address all of these issues with the following reduction. Scale weights so that every edge weighs at least 1. Round up edge weights to the nearest integer. Note that edge weights increase by at most 2 by these transformations. Introduce virtual intermediate nodes along each edge, at intervals of length 1. Remove all weights. Let the new unweighted graph be denoted G'. It is easy to see that virtual nodes do not change the topological properties of the graph. Hence, if G excludes $K_{r,r}$, then so does G'. Now, perform the probabilistic sparse partition above on G', and let the resulting clusters in G' be $C'_1, ..., C'_m$. Output the set of clusters $G[C'_1], ..., G[C'_m]$ induced by G''s clusters.

To see that the resulting partition satisfies the required properties, first observe that for any $u, v, \in C'$, distances satisfy $d_G(u,v) \leq d_{C'}(u,v)$. Hence, any bound on the diameters of the C' clusters is maintained in the clusters induced on G.

Second, let us consider the probability that an edge $(u,v) \in G$ is cut by the partition. This edge is represented in G' by at most $\lfloor d_G(u,v) + 2 \rfloor$ unweighted edges. By union bound, the probability that (u,v) is cut is at most $\lfloor d_G(u,v) + 2 \rfloor \frac{2}{\rho} \leq \frac{6 d_G(u,v)}{\rho}$.

We remark that the time complexity of the construction does suffer from the transformation, by a factor that is proportional to the aspect ratio of G.

6 Open Problems

The results of this paper could be utilized and optimized in many ways. The work suggests two main open problems.

First, all our theorems show an exponential dependency on the size of the forbidden minor. When weak diameter is concerned it is possible to achieve a polynomial dependency [16]. It would be very interesting to find sparse covers and sparse partitions with strong diameter and a polynomial dependency in r. Note that the exponential dependency is an artifact of the technique of doubling the width of the cutting strips each iteration. This is a key ingredient of our approach, thus such an improvement would probably require a different approach.

Finally, Theorem 1.6 seems to be a strong partitioning tool that may be a tool for substantial advancement in various approximation algorithms. Natural candidates are metric embeddings and building spanners.

References

[1] I. Abraham, Cyril Gavoille, and Dahlia Malkhi. Compact routing for graphs excluding a fixed minor. In *The 19th Intl. Symposium on Distributed Computing (DISC).*

[2] Ittai Abraham and Cyril Gavoille. Object location using path separators. In 25^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 188–197. ACM Press, July 2006.

[3] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. Routing with improved communication-space trade-off. In 18^{th} *International Symposium on Distributed Computing (DISC)*, volume 3274 of Lecture Notes in Computer Science, pages 305–319. Springer, October 2004.

[4] Ittai Abraham, Cyril Gavoille, and Dahlia Malkhi. On space-stretch trade-offs: Lower bounds. In 18^{th} *Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 217–224. ACM Press, July 2006.

[5] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noan Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In 16^{th} *Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA)*. ACM PRESS, June 2004.

[6] B. Awerbuch and D. Peleg. Locality-sensitive resource allocation, 1990.

[7] Baruch Awerbuch, Amotz Bar-Noy, Nathan Linial, and David Peleg. Compact distributed data structures for adaptive routing. In 21^{st} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 479–489. ACM Press, May 1989.

[8] Baruch Awerbuch, Amotz Bar Noy, Nati Linial, and David Peleg. Improved routing strategies with succinct tables. *Journal of Algorithms*, 11(3):307–341, 1990.

[9] Baruch Awerbuch and David Peleg. Network synchronization with polylogarithmic overhead. In *IEEE Symposium on Foundations of Computer Science*, pages 514–522, 1990.

[10] Baruch Awerbuch and David Peleg. Sparse partitions. In 31^{th} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 503–513. IEEE Computer Society Press, October 1990.

[11] Baruch Awerbuch and David Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discret. Math.*, 5(2):151–162, 1992.

[12] Costas Busch, Ryan LaFortune, and Srikanta Tirthapura. Improved sparse covers for graphs excluding a fixed minor. Technical Report 06-16, Department of Computer Science, Rensselaer Polytechnic Institute, November 2006.

[13] J. Lawrence Carter and Mark N. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.

[14] Jittat Fakcharoenphol and Kunal Talwar. An improved decomposition theorem for graphs excluding a fixed minor. In *RANDOM-APPROX*, pages 36–46, 2003.

[15] Pierre Fraigniaud and Cyril Gavoille. Routing in trees. In 28^{th} *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 2076 of Lecture Notes in Computer Science, pages 757–772. Springer, July 2001.

[16] Philip Klein, Serge A. Plotkin, and Satish Rao. Excluded minors, network decomposition, and multicommodity flow. In 25^{th} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 682–690. ACM Press, 1993.

[17] Andrew Thomason. The extremal function for complete minors. *Journal of Combinatorial Theory, Series B*, 81(2):318–338, 2001.

[18] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. In 42^{nd} *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 242–251. IEEE Computer Society Press, October 2001.

[19] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In 33^{rd} *Annual ACM Symposium on Theory of Computing (STOC)*, pages 183–192, Hersonissos, Crete, Greece, July 2001.

[20] ttai Abraham, Cyril Gavoille, Andrew V. Goldberg, and Dahlia Malkhi. Routing in networks with low doubling dimension. In *ICDCS*, page 75, 2006.

A Name-Independent Routing

In this section we present the proof of Theorem 1.3

First let us outline the technique of hierarchical routing schemes introduced by Awerbuch and Peleg [10, 11]. Let us assume that there exist k and τ such that, for every $\Delta > 0$, the graph G has a (k, Δ) bounded sparse τ cover, and let $\mathcal{C}_\Delta$ denote this cover. Then, routing in G can be done by considering the family of covers $\mathcal{F} = \{\mathcal{C}_1, \ldots, \mathcal{C}_{2^i}, \ldots, \mathcal{C}_{2^{\lceil \log D \rceil}}\}$. More precisely, for each cover $\mathcal{C}_{2^i} \in \mathcal{F}$ and for each cluster $C \in \mathcal{C}$, we root a shortest path spanning tree T_C of $G[C]$, so of depth at most $k2^i$. Let us call $\mathcal{T}$ the collection of all these trees. Roughly speaking, the routing task for a source u consists in seeking the target v in each tree of $\mathcal{T}$ it belongs to.

Actually, u needs to seek v only in $\lceil \log D \rceil + 1$ trees in nondecreasing depth, each tree spanning the ball $B_G(u, 2^i)$ for some $i \in \{0, \ldots, \lceil \log D \rceil\}$. If each try can be done within a route of length proportional to the depth of the tree, then it is not difficult to check that the resulting stretch of the route from u to v is $O(k)$, the cluster covering $B_G(u, 2^i)$ being of diameter at most $k2^i$. Overall, if a tree routing scheme for seeking v can be implemented with M-bit routing tables, then the routing scheme for G uses at most $O(\tau \cdot \log D \cdot M)$ bits, each node participating in at most τ tree routings for each of the $O(\log D)$ covers of $\mathcal{F}$.

By Theorem 1.2, we have $\tau = r!2^{O(r)}$ and $k = O(r^2)$. Therefore Theorem 1.3 can be proved by designing a routing scheme with $M = O(\log^3 n / \log\log n)$-bit routing tables for seeking any destination in a tree along routes of length proportional to its depth. Unfortunately, such tree routing schemes cannot be applied as a black box and plugged to the hierarchical scheme for G. Indeed, routing from u to v in some partial tree T of G inevitably requires to visit some nodes outside T: forcing routing to use only edges of T is equivalent to routing in a weighted tree T' spanning G where edges of $T' \setminus T$ have some large costs. And we have seen that routing trees with edge weights in $\{1, k\}$ requires $\Omega((n \log n)^{1/k})$ bits for stretch $\Theta(k)$ [4]. It follows that routing in a cluster C will interact with nodes not in C. This is source of several complications since a node might be concerned with (and its routing table possibly charged for) the routing to some trees in which it does not belong to. Potential a node maybe a neighbor of all the $|\mathcal{T}| = \Omega(n)$ trees. To solve the problem, we will use a modified version of the single-source unweighted tree routing of [3] combined with the low density of minor-free graphs to balance the routing information.

An *L-error reporting* routing scheme for a subgraph C of G is a routing scheme such that, for all $u \in C$ and v of G: if $v \in C$, then the route from u to v has cost at most L, and if $v \notin C$, then the routing from u to v reports to u a *failure* mark in the header after a loop of cost at most L. In order to prove Theorem 1.3, our goal is to construct, for each depth-h tree $T \in \mathcal{T}$, a space efficient $O(h)$-error reporting routing scheme (cf. Lemma A.2 below).

A α-orientation of a graph is an orientation of its edges such that every node has out-degree at most α.

Lemma A.1. *Any graph excluding a $K_{r,r}$ minor has an $\Theta(r\sqrt{\log r})$-orientation that can be computed in linear time.*

Proof. Graph excluding a fixed minor are closed under taking induced subgraphs. It is known that the n-node graphs excluding a K_r minor have no more that $f(r) \cdot n$ edges [17] where $f(r) = \Theta(r\sqrt{\log r})$. Therefore, an $f(r)$-orientation can be easily obtained in linear time by pruning the graph by its minimum degree node. Now, the family of graphs excluding a K_{2r} minor contains all the graphs excluding a $K_{r,r}$ minor. Thus any n-node graph excluding a $K_{r,r}$ minor has no more than $f(2r) \cdot n$ edges, and so an $\Theta(r\sqrt{\log r})$-orientation computable in linear time. □

The *sparsity* of a collection of trees of a graph G is the maximum number of trees a node of G belongs to. The key lemma is the following:

Lemma A.2. *Let $\mathcal{T}$ be a collection of trees of sparsity σ in an n-node graph G with an α-orientation. Then, one can construct in polynomial time for each node of G a routing table of $O(\sigma \cdot \log n \cdot (\alpha + \log^2 n / \log\log n))$ bits, such that each depth-h tree of $\mathcal{T}$ has a $8h$-error reporting routing scheme using $O(\log n)$-bit headers.*

The proof of Theorem 1.3 is completed thanks to Lemma A.2 by observing that the sparsity of the collection $\mathcal{T}$ is $\sigma = O(r!2^{O(r)} \cdot \log D)$, and that $\alpha = O(r\sqrt{\log r})$ (Lemma A.1), so providing a $O(r!2^{O(r)} \cdot \log D \cdot \log^3 n / \log\log n)$-bit routing scheme for G as claimed.

Proof. Let $\mathcal{U}$ be the set of names for G, $|\mathcal{U}| \leq n^{O(1)}$. Consider a depth-$h$ tree T of $\mathcal{T}$, and let n be the number of nodes of T, and s be a root for T of minimum eccentricity. Consider a source u of T, and let v be any node of G. For simplicity, we confuse the nodes with their names, that is we assume that $u, v \in \mathcal{U}$.

Let $k : \mathcal{U} \to \mathcal{P}$ be some universal hashing function mapping the names of $\mathcal{U}$ to $\mathcal{P} = \{1, \ldots, p\}$ where p is some prime such that $n \leq p < 2n$. Such function h can be implemented with a degree-$O(\log n)$ polynomial of the field $\mathbb{Z}_p$ such that there are at most $O(\log n)$ collisions [13], thus using $O(\log^2 n)$ bits.

The outline of the L-error reporting scheme for T from u to v is the following:

1. Node u hashes v in $k(v) \in \mathcal{P}$.
2. Node u routes a message to a node w of T, thanks to a labeled tree routing scheme $\mathcal{L}_1$, whose label in $\mathcal{L}_1$ is precisely $k(v)$.
3. Node w is in charge of the labels, for a tree routing scheme $\mathcal{L}_2$, of all the nodes z of T such that $k(z) = k(v)$.
4. If v is not one of such node z of T, then a failure mark is sent back from w to u using $\mathcal{L}_1$.
5. Otherwise, w route to v using a routing label in tree routing scheme $\mathcal{L}_2$.

More precisely, the tree routing schemes $\mathcal{L}_1, \mathcal{L}_2$ are based on a specific port labeling of T, say a virtual port labeling. They requires that ports number i of u leads to the ith heaviest child of u in T (the port to the parent of u is fixed to 0). So, given the label of the current node x and the label of the destination y, say $\ell_2(x)$ and $\ell_2(y)$, the scheme $\mathcal{L}_2$ computes the *virtual* port number i of the edge on the path from x to y in T. For the scheme $\mathcal{L}_2$, we will use the label tree routing scheme of [15], that uses $O(\log n)$-bit labels.

Unfortunately, *real* port numbers of u are fixed and range in $\{1, \ldots, \deg_G(u)\}$, and $\deg_G(u) \neq \deg_T(u)$ in general. To overcome this problem we distribute a translating table from virtual to real port numbers over all the neighbors of u, potentially charging some nodes of G that are not in T. To take care of the over load of some nodes, we use the α-orientation of G.

Let $\mathrm{port}(x, y)$ be the real port number of the edge from x to y in G, and consider a node z with directed neighbors (according to the α-orientation) $z^1, \ldots, z^\alpha$, and let z_i be the ith heaviest child of z.

For the port translation, node z stores: 1) the real port number of its parent for each tree of $\mathcal{T}$ it belongs to; 2) $\mathrm{port}(z, z^i)$ for each $i \in \{1, \ldots, \alpha\}$, and for each tree of $\mathcal{T}$ node z belongs to; 3) $\mathrm{port}(z^i, z^i_p)$, where $p = \mathrm{port}(z^i, z)$, for each $i \in \{1, \ldots, \alpha\}$, and for each tree of $\mathcal{T}$ node z^i belongs to. We check that overall the memory requirements for the port translation is $O(\sigma \cdot \alpha \cdot \log n)$ bits per node of G.

The port translation in a node x of T is performed as follows: if the virtual port is 0, then the real port of the parent of T is returned. If the virtual port is one of the real port to $x^1, \ldots, x^\alpha$, then the real port is returned. Otherwise, if the virtual port is p, then a message from x on port p, specifying the tree T of $\mathcal{T}$, is sent, and let z be its neighbor. Note that this edge is incoming in x in the orientation, and outgoing from z. Therefore, z knows the real port number of the pth child of x in the tree T. In other words, the routing from x to its parent is done in 1 step, whereas for a child it is done in at most three steps.

The L-error reporting routing scheme from u to v in T has the following performances: 1) if $v \notin T$, then the route has length $L \leq d_T(u, s) + 3d_T(s, w) + d_T(w, s) + 3d_T(s, u) \leq 8h$. 2) if $v \in T$, then the route has length $L \leq d_T(u, s) + 3d_T(s, w) + d_T(w, s) + 3d_T(s, v) \leq 8h$. Therefore, the scheme is $8h$-error reporting.

It remains to describe the tree routing scheme $\mathcal{L}_1$ (see [3] for details). The scheme is based on two numbers, $c(x)$ and $q(x)$, we assign with each node x of T. The first, called the *charge of* x, represents the total number of values $k(v)$ mapped on the nodes of T_x, the subtree T of root x. (So for the root s, $c(s) = |\mathcal{P}| = p$). The second one denotes the number of values $k(v)$ that are mapped to node x. These two numbers satisfy that, for every x, $c(x) = \sum_{y \in T_x} q(y)$.

Given the numbers $c(x)$ and $q(x)$ one can then route through a *modified* DFS number $f(x)$ associated with each x and defined by: for the root $f(s) := 1$, and $f(x_i) := f(x) + q(x) + \sum_{j<i} c(x_j)$, where x_i is

the ith child of x. (This matches to the standard DFS numbering if $q(x) = 1$ for every x.)

Now the routing is done similarly to Interval Routing Scheme. Let w be the node in charge of $k(v)$. Assume that w is a descendant of some node x, initially $x = s$. It is easy to see that:

1. either $k(v) \in [f(x), f(x) + q(x))$, and $w = x$, i.e., the key of v is stored by x;
2. or w is a descendant of x_i where $k(v) \in [f(x_i), f(x_{i+1}))$, and thus the routing in x must answer port i.

So the routing from x to $k(v)$ is well defined if x is aware of $f(x)$, $q(x)$, and of the vector $\vec{c}(x) = (c(x_1), c(x_2), \dots)$ of charges of x's children. Indeed the numbers $f(x_i)$ and $f(x_{i+1})$ can be computed from $f(x)$, $q(x)$, and from $\vec{c}(x)$.

It is proved in [3] that $c(x)$ and $q(x)$ can be polynomially computed in a particular way so that $q(x) = O(\log n / \log\log n)$ and $\vec{c}(x)$ contains at most $O(\log^2 n / \log\log n)$ distinct values, and so coded with $O(\log^3 n / \log\log n)$ bits.

A node x belonging to T stores $q(x)$, $c(x)$, $\vec{c}(x)$, and all the labels and names for which x is in charge: this is $O(q(x)\log n)$ values, since there are $O(\log n)$ nodes of T that can collide in the same node x. Labels and names are on $O(\log n)$ bits, therefore the storage for $\mathcal{L}_1$ in x is $O(\log^3 n / \log\log n)$ bits for T.

Thus a node of G stores $O(\sigma \cdot \log^3 n / \log\log n)$ bits for $\mathcal{L}_1$ plus $O(\sigma \cdot \alpha \cdot \log n)$ bits for the port translation table and the scheme $\mathcal{L}_2$. The hashing functions for each tree represents $O(\sigma \cdot \log^2 n)$ bits. In total, a node of G stores $O(\sigma \cdot \log n(\alpha + \log^2 n / \log\log n))$ bits as claimed.

We check that all the headers are on $O(\log n)$ bits, completing the proof of the lemma. $\square$

B Figures

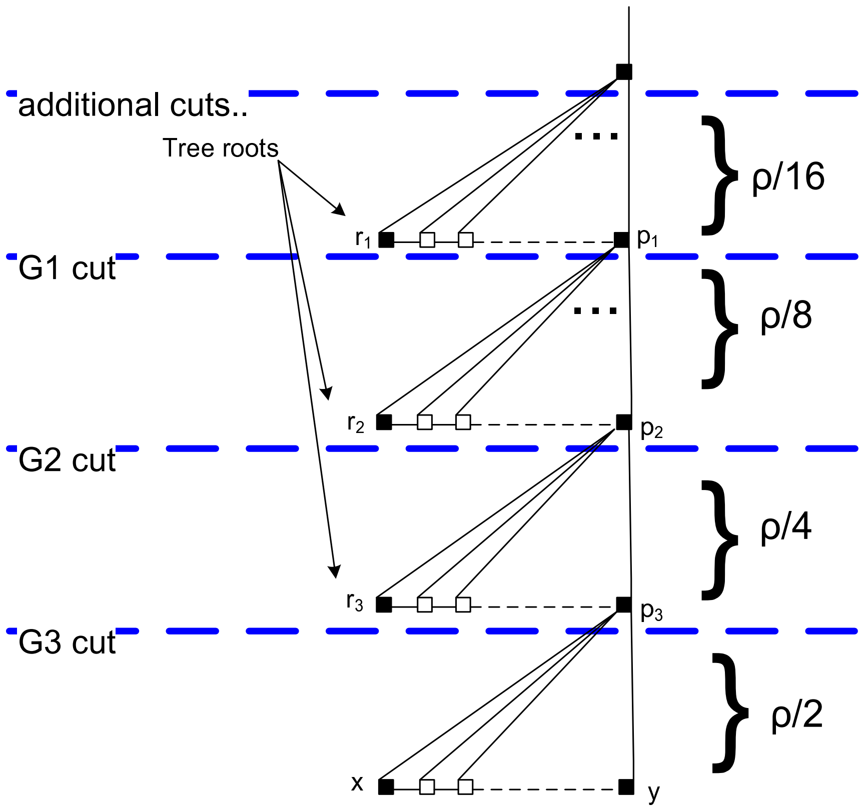

Figure 1: Example graph in which KPR partition has arbitrarily large diameter

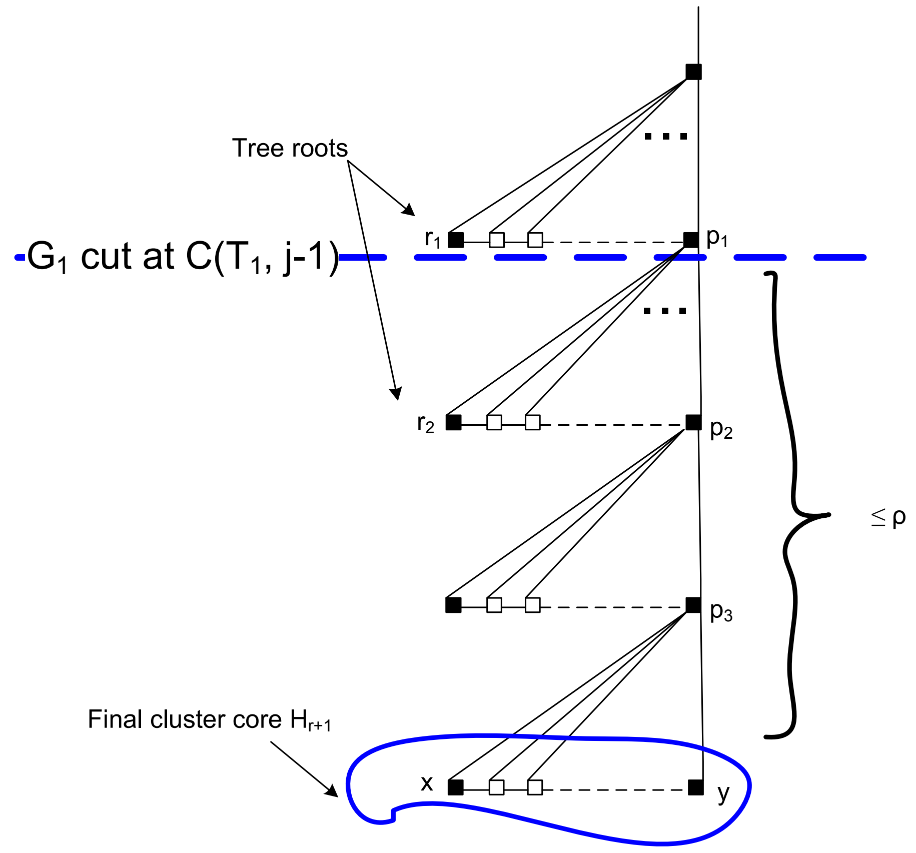

Figure 2: A revisit of the graph in which KPR partition has arbitrarily large diameter; the construction above has small strong diameter.