vertex if at least one of its coordinates is in $\{1, m\}$, and that the vertex y is *antipodal* to x if $x_i + y_i = m + 1$ for every i. The following is thus a special case of Tucker's lemma:

Lemma 3.7 *Let f be a vertex coloring of $C = C(m,k)$ with colors from $\{-k,\ldots,-1,1,\ldots,k\}$ such that if x is a boundary vertex and y is antipodal to x, then $f(y) = -f(x)$. Then there exist two adjacent vertices u, u' with $f(u') = -f(u)$.*

To derive lemma 3.5 we argue as follows: Define a graph $\bar{C}$ on the vertex set $\{0,\ldots,m+1\}^k$ and the same adjacency relationship as C, viz. x is adjacent to $x+e$ for every $e \in \{0,1\}^k$. Lemma 3.7 is to be applied to $\bar{C}$, with the understanding that being a boundary vertex means having a coordinate which equals either 0 or $m+1$, and that vertices x and y are antipodal in $\bar{C}$ if $x_i + y_i = m + 1$ for all i. Now suppose we are given a vertex coloring of C with colors in $\{1,\ldots,k\}$. A coloring of $\bar{C}$ by $\{-k,\ldots,-1,1,\ldots,k\}$ is induced as follows: For a boundary vertex x of $\bar{C}$ let j be the first index for which $x_j \in \{0, m+1\}$. If $x_j = 0$, color x by $-j$, and if $x_j = m+1$, color it j. Now each vertex u with no 0 or $m+1$ is also a vertex of C and thus has a color j; if there is a path consisting of j-colored vertices connecting u to the set $\{x : x_j = 0\}$ then recolor u by $-j$, otherwise u's color remains unchanged.

Obviously, there cannot be two adjacent vertices colored j and $-j$, because the rule for sign change calls for replacing the j by $-j$. Lemma 3.7 may be invoked now to conclude that in this coloring of $\bar{C}$ there are two antipodal vertices whose colors do not sum to zero. Therefore some boundary vertex must have been recolored in the above recoloring process, which means that for some $1 \le j \le d$ there is a path of j colored vertices in C connecting a vertex with $x_j = 1$ to one with $x_j = m$, as required. ∎

Keeping the dimension k fixed and letting m tend to infinity, the following geometric theorem is derived:

Corollary 3.8 *Let the $k-$dimensional cube be covered by k closed sets. Then there is a connected component of one of these sets which intersects with two opposite facets of the cube.*

It would be reasonable to assume that this result is known to geometers/topologists, but we could not find it in the literature. If the closed sets are also assumed to be homotopically trivial, this corollary follows from the concept of category for topological spaces (cf. [Sp]).

4 A fast randomized distributed algorithm for low diameter decompositions

4.1 The algorithm

We would like to replace the sequential algorithm of section 2 by a fast distributed one. The sequential algorithm constructs the blocks iteratively, one at a time. The time that the algorithm takes is $O(\lambda t(n))$ where $t(n)$ is the worst case time for one iteration. In the algorithm as described $t(n)$ could be linear in n. In this section we show how to use randomization to replace the serial algorithm for the construction of a single block by a parallel distributed one which achieves the same trade-off as the serial algorithm between weak diameter and λ. The time to construct one block is reduced to $O(\log D)$ where D is the weak diameter, and thus we achieve total running time of $O(D\lambda)$. In particular, when we balance D and λ, we produce a $O(\log n)$-decomposition with weak diameter $O(\log n)$ that runs in $O(\log^2 n)$ time.

To construct a single block quickly, we'd like to parallelize the selection of safe radii. However, if we allow many vertices to choose a safe radius simultaneously the balls around the vertices may overlap significantly. In that case, there is no longer an obvious criterion for deciding which vertices are to be placed in the

block in such a way that the resulting block is guaranteed both to have small weak diameter and to contain a substantial fraction of the vertices. It is this difficulty that we must resolve.

We make the usual assumption that each vertex x has a unique integer ID_x. Of course, if such ID's are not provided then each vertex can select an ID uniformly at random from, for example, $\{1,2,...,n^2\}$ which guarantees that the ID's are unique with high probability; the algorithm can be modified to work under this assumption.

The algorithm for selecting a block out of a graph G, is called *Construct_Block*. First each vertex y selects an integer radius r_y at random (according to a distribution (given below) that is approximately geometric). It then broadcasts (ID_y, r_y) to all vertices that are within distance r_y of it. After collecting all such messages from other vertices, each vertex y selects the vertex $C(y)$ of highest ID from among the vertices whose broadcast it received in the first round (including itself), and joins the current block if $d(y, C(y)) < r_{C(y)}$ (note that it is necessarily the case that $d(y, C(y)) \leq r_{C(y)}$).

The distribution by which each vertex x selects its radius r_x is a truncated geometric distribution, which is defined in terms of two parameters, p and B:

$$Pr(r_x = j) = p^j(1-p)$$

for $j = 0, \ldots, B-1$, and

$$Pr(r_x = B) = p^B.$$

While the above algorithm is conceptually quite simple, there are some subtleties involved in implementing the above algorithm efficiently in an asynchronous distributed network, i.e., accomplishing the broadcast in such a way that each vertex knows when it has received all the broadcasts that it will get and can thus select $C(y)$. This can be done using standard synchronization techniques, which are discussed in the final subsection of this section.

4.2 Proof of correctness

The key properties of this algorithm are summarized by:

Lemma 4.1 *Suppose Construct_Block is applied to a graph G with at most n vertices. Let S be the set of vertices comprising the block selected. Then:*

1. *The set of selected vertices has weak diameter at most $2B$.*
2. *For each vertex y of G, the probability that it belongs to S is at least $p(1-p^B)^n$.*

If we apply Construct_Block iteratively to decompose the entire graph (using the same values of p and B at each iteration), then the first part of the lemma guarantees that the weak diameter of the resulting decomposition is at most $2B$. The second part of the lemma implies that if $q = p(1-p^B)^n$ then for each vertex x, the probability that x is not assigned to one of the first i blocks is at most $(1-q)^i$ and thus the probability that some vertex is unassigned after i iterations is at most $n(1-q)^i$. By selecting B to be $\frac{\log n + \omega(n)}{\log(1/p)}$ where ω is any function tending to infinity with n, it is easily verified that $q = p(1+o(1))$ where the little oh term depends only on the choice of ω. Thus with high probability, the number of iterations (and hence the number of colors) does not exceed $(1+o(1))\frac{\log n}{\log(1/(1-p))}$. The result is a λ-decomposition with diameter D where the expressions for λ and D in terms of p and n are essentially the same as obtained for the sequential algorithm, and therefore the trade-off is the same.

It remains to prove the lemma. The first part of the lemma follows easily from:

Claim. For each connected subset T of S, $C(y)$ is the same vertex x for all $y \in T$.

By the claim, any two vertices in T are at distance at most B from x and thus at most $2B$ from each other.

We prove the claim by contradiction; if it is false then there are two adjacent vertices

y and z belonging to S with $C(y) \neq C(z)$. Without loss of generality, the ID of $C(y)$ exceeds that of $C(z)$. By the definition of S, y is in S implies $r_{C(y)} > d(C(y), y)$. Since y and z are neighbors, $r_{C(y)} \geq d(C(y), z)$ and thus z received the broadcast sent by $C(y)$ in the first round. This contradicts the fact that $C(z)$ is the vertex of highest ID whose broadcast reached z.

We now proceed to the proof of the second part of the lemma. We fix a vertex y and estimate the probability that it is assigned to S. We can bound this probability as follows:

$$Pr(y \in S) \geq \sum_{z|d(z,y)<B} Pr(y \in S | C(y) = z) Pr(C(y) = z)$$

For a given vertex z, define the following three events:

$D_z : r_z \geq d(z, y)$

$E_z : r_z > d(z, y)$

F_z: For every vertex w with ID higher than z, $r_{C(w)} < d(w, y)$.

Then for z such that $d(z, y) < B$ we can rewrite $Pr(y \in S | C(y) = z)$ as:

$$\begin{aligned} &Pr(E_z \wedge F_z | D_z \wedge F_z) \\ &\quad = Pr(E_z \wedge F_z)/Pr(D_z \wedge F_z) \\ &\quad = Pr(E_z)/Pr(D_z) \\ &\quad = p, \end{aligned}$$

where the first equality follows from the fact that D_z implies E_z, the second equality follows from the fact that F_z is independent of both D_z and E_z, and the third follows from the definition of the distribution on selected radii which implies that $Pr(E_z) = p^{d(z,y)+1}$ and $Pr(D_z) = p^{d(z,y)}$. Thus,

$$Pr(y \in S) \geq p \sum_{z|d(z,y)<B} Pr(C(y) = z)$$

$$\begin{aligned} &\geq pPr(d(C(y), y) < B) \\ &\geq pPr(r_z \neq B, \forall z) \\ &\geq p(1 - p^B)^n. \end{aligned}$$

4.3 Implementation details

Implementing the above algorithm requires that each vertex y send (ID_y, r_y) to every vertex z that is within distance r_y of y and also that each vertex y be able to detect that it has received all such messages that are intended for it (so that it can correctly select $C(y)$. Below we sketch how to use a standard synchronization technique to achieve this. The resulting algorithm has the disadvantage that it requires that nodes send very long messages. We will then indicate how to modify the algorithm to eliminate this disadvantage.

To synchronize the network we require that each node proceed in a sequence of communication *steps*. During the step i, the node sends one message to each of its neighbors and receives one message from each of its neighbors. The node can not send out any step $i+1$ messages until it has sent and received all of its step i messages. Notice that not all nodes will begin step 1 at the same time, and, indeed, a node may not begin step 1 until after it has received some step 1 message from another node. However, this method ensures that two neighboring nodes are never more than one step apart.

The algorithm works by having each node y build a sequence of sets $S[i]_y$, for $0 \leq i \leq B$, where $S[i]_y$ consists of all pairs (ID_z, r_z) for nodes z for which $d(z, y) = i \leq r_z$. These sets are constructed in B steps, with the set $S[i]_y$ being constructed during the i^{th} step of the algorithm. Before step 1, $S[0]_y$ consists of the singleton (ID_y, r_y). Having constructed set $S[i-1]_y$ prior to step i, node y defines $T[i]_y$ to be the subset of $S[i-1]_y$ consisting of those pairs (ID_x, r_x) with $r_x \geq i$ and sends the set $T[i]_y$ to each of its neighbors during step i. Once it has received $T[i]_z$ from each of its neighbors z, it defines $S[i]_y$ to be the union of

$T[i]_z$ over all its neighbors z, minus the union of $S[j]_y$ over all $j < i$. It is easy to show by induction that the sets $S[i]_y$ are as required. After B steps, the node then selects $C(y)$ to be the node of maximum ID among all of the sets $S[i]_y$.

The drawback to this algorithm is that the sets $T[i]_y$ transmitted in step i can grow very large, requiring very large messages. One way to reduce the message size is to realize that if two pairs (ID_x, r_x) and (ID_w, r_w) are both in $S[i-1]_y$ and $r_x = r_w$, then y need only put the one with the larger ID (say ID_x) in $T[i-1]_y$. Thus at each step, the message sent by a node contains at most B pairs, one for each possible value of r_x. It can be shown easily by induction that the maximum element in the union of $S[j]_y$ over $j \leq i$ in the modified algorithm is the same as in the unmodified algorithm.

Finally, the messages can be shortened further by modifying the algorithm as follows. At any time, node y remembers only the pair (ID_x, r_x) for which ID_x is maximum among all pairs that it has received thus far. At step i, the node sends this pair to each of its neighbors if $r_x \geq i$, otherwise it sends a null message to its neighbors. The node $C(y)$ is then the node stored after step B.

This algorithm does not produce the same result as the algorithms presented above, i.e., $C(y)$ is not necessarily the vertex of maximum ID such that $r_{C(y)} \geq d(C(y), y)$. Instead, $C(y)$ is the vertex of maximum ID such that there is a path of length at most $r_{C(y)}$ from $C(y)$ to y such that for each vertex w on the path, $C(y)$ is the vertex of highest ID in the ball of radius $d(w, C(y))$ around w. The proof of lemma 4.1 can be easily modified to work for this definition of $C(y)$.

References

[AR] Y. Afek and M. Ricklin, *Sparser: A paradigm for running distributed algorithms*, manuscript, Tel-Aviv University.

[ABI] N. Alon, L. Babai, and A. Itai. *A fast and simple randomized parallel algorithm for the maximal independent set problem*, J. of Algorithms 7(1986), 567-583.

[A] B. Awerbuch, *Complexity of network synchronization*, J. ACM 32(1985), 804-823.

[AGLP] B. Awerbuch, M. Luby, A. Goldberg and S. Plotkin, *Network decomposition and locality in distributed computation*, Proc. 30th IEEE Symp. on Foundations of Comp. Sci.(1989) 364-369.

[AP] B. Awerbuch and D. Peleg, *Sparse Partitions*, Proc. 31st IEEE Symp. on Foundations of Comp. Sci. (1990) 503-513.

[BM] I. F. Blake and R. C. Mullin: *An Introduction to Algebraic and Combinatorial Coding Theory*, Academic Press, New York, 1976.

[B] B. Bollobas, *Extremal graph theory*, Academic Press.

[CV] R. Cole and U. Vishkin, *Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms*, Proc. 18th ACM Symp. on Theory of Computing (1986) 206-219.

[FT] R. M. Freund and M. J. Todd, *A constructive proof of Tucker's combinatorial lemma*, J. Comb. Theory A 30(1981) 321-325.

[GPS] A.V. Goldberg, S.V. Plotkin and G.E. Shannon, *Parallel Symmetry-Breaking in Sparse Graphs*, SIAM J. Disc. Math. 1 (1989) 434-446.