Spanners in the Congested Model

by

Leonid Beranboim, Michael Elkin, and Tzalik Maimon

Technical Report #18-03

July 10, 2018

The Lynne and William Frankel Center for Computer Science,

Department of Computer Science,

Ben-Gurion University, Beer Sheva, Israel.

Spanners in the Congested Model

Leonid Beranboim Michael Elkin Tzalik Maimon

July 9, 2018

Abstract

The problems of connectivity and Minimum Spanning Tree (MST) have been well studied in the distributed setting. Recently, more interest has been found with these problems in the *Congested Clique* model as a consequence of Lenzen Routing [16]. The problems were also reduced as it was shown that solving the connectivity problem can help solve the MST problem (and obviously MST solves connectivity). Spanners can help with solving connectivity. In fact, spanners are interesting by their own merit. It was shown that a $(2k-1)$-*additive-spanner* has a lower bound on the size of $\Omega(n^{1+\frac{1}{k}})$ for $k > 2$ [23]. The lower bound for the multiplicative spanner is still a conjuncture made by Erdos [11] and is yet to be proven. Not much progress has been done to provide with a linear size spanner deterministically, not even in the Congested-Clique. Thus, we show in this paper some interesting results. The first one shows that the connectivity problem can be solved deterministically using a linear size spanner within constant running time on graphs with *bounded neighborhood independence*. Much more than this, our result works in the $\mathcal{CONGEST}$ model. It also immediately leads to a constant time deterministic solution for the connectivity problem in the Congested-Clique.
Our second result provides a linear size spanner in the $\mathcal{CONGEST}$ model for graphs with bounded *diversity*. Here too our result has constant running time and is deterministic. The difference is that the spanner we provide has also a small stretch which is a desired property from a spanner. More specifically, the spanner we provide has a constant stretch in bounded diversity graphs.

1 Introduction and Related Work

The distributed setting is a well studied setting in which we have a graph in which each vertex is treated as a processor and each edge is a communication line. In this setting there are several models which are of interests; The $\mathcal{LOCAL}$ model, where the running time is counted as the number of rounds of messages one needs to perform in order to achieve some task, and where the size of messages is not limited and local computations are not counted towards the running time; The $\mathcal{CONGEST}$ model, which is much like the $\mathcal{LOCAL}$ but with a limit on the size of each message that can be passed on each edge in the graph. This

limit is usually considered to be $O(\log n)$ bits; And the *Congested-Clique* model, which is sometimes referred to as *all-to-all communication*, is one where, like the $\mathcal{CONGEST}$, the size of messages is limited but each vertex is connected with a communication line to all other vertices in the graph, thus forming a clique. Sometimes the input in the Congested-Clique model is a subgraph of the clique on which one wishes to solve a certain problem. this model attracted much attention in recent years.

The problem of spanning a graph is a well-studied problem in the distributed setting, and a lot of research was done in the direction of solving spanners or the more lenient problem of connectivity [**?**]. A spanner is a subgraph H in which each vertex has a path to each of its neighbors in the input graph G. The connectivity problem, given a connected input graph $G = (V, E)$, and a subgraph $G' = (V, E' \subseteq E)$, is whether any two vertices in V are connected by a path in G'. An (α, β)-*spanner* H of a graph G is one where the distance between each two vertices v, u is such that $d_H(v, u) \leq \alpha \cdot d_G(v, u) + \beta$ where d is the distance between two vertices. One can consider a spanner where $\alpha = 1$, called an *additive spanner* or one where $\beta = 0$, called a *multiplicative spanner*.

The publication of Lenzen Routing [16] brought much focus to the research of many problems in the Congested-Clique, among them the MST problem. Later, Hegeman et al. [14] obtained reductions between the MST problem and the connectivity problem in the congested clique, which result in $O(\log \log \log n)$-time randomized algorithms for these problems. Then, Korhonen [18] showed a deterministic variant to a randomized procedure used in [14]. However, due to certain assumptions required to employ this procedure, the deterministic running time becomes $O(\log \log n)$. This matches the time of the deterministic MST algorithm for congested cliques of Lotker et al. [**?**]. But still, to the best of our knowledge, no constant time deterministic algorithm exists for solving the connectivity problem in the Congested-Clique model. When considering randomized solutions the situation is much better. In fact, a randomized constant time solution for the MST problem itself was recently published by Jurdziński and Nowicki [15] which constitutes an optimal randomized solution in this model for both MST and connectivity. We note though that the problem of a small stretch spanner remains open. And yet the best deterministic result for MST (and connectivity) in the Congested-Clique is that of Lotker et al. [17]. We devise a simple deterministic solution for spanning a graph in the more general $\mathcal{CONGEST}$ model for graphs with bounded *neighborhood independence*. Specifically, our solution enjoys constant running time on such graphs. Roughly speaking, the neighborhood independence of a vertex v is the largest independent subset of vertices one can choose from the 1-hop neighborhood of v.

In this paper we focus on spanners. Specifically, multiplicative spanners which, in turn, help to solve the connectivity problem in the Congested-Clique. In general, spanners are used in many aspects of the distributed settings as they are a sparse counterpart of the original graph and yet retain certain traits. Thus

they are used in constructing network synchronizers [19], compact routing tables [1, 2, 21] and distance labeling schemes [22]. There are several constructions of additive and multiplicative spanners but yet none of them enjoy both linear size and a constant stretch, both important characteristics of a spanner and usually one can see a trade-off between the two.

A lower bound on the size of a $(2k-1)$-additive-spanner is known. Specifically, Woodruff [23] showed that such a spanner has a size of $\Omega(n^{1+\frac{1}{k}})$ thus ending the search for such linear size additive spanner for the general case. In the case of multiplicative spanners, there was already much research done. Elkin [10] showed a randomized algorithm for an $O(\log n)$-spanner of linear size with $O(\log^3 n)$ running time. Pettie [20] devised a randomized algorithm for an $O(2^{\log^* n}\log n)$-spanner with linear size with running time of $O(\log^{1+o(1)} n)$. A randomized result was achieved by Elkin and Neiman [12] where they constructed a $(2k-1)$-spanner of size $O(n^{1+1/k}/\epsilon)$ for some $\epsilon > 0$ with probability $1-\epsilon$ in $O(k)$ running time. Furthermore, their spanners are sparse when $k = \omega(\log n)$. Specifically, they achieved spanners of size $n(1+o(1))$ with probability $1-o(1)$ in that case.

Deterministic results were also achieved for multiplicative spanners. Derbel et al. [8] devised a construction of a $(2k-1)$-spanner with size $\mathrm{O}(kn^{1+1/k})$ and running time of $O(k)$. Derbel et al. [6] devised a construction of an $O(k^{\log 5})$-spanner with size $O(\log kn^{1+1/k})$ but with a running time of $O(n^{1/\sqrt{\log n}})$. Another result from the same authors [7] devised a k-round algorithm for constructing $(2k-1)$-spanners with optimal size. We note that all the mentioned deterministic results were achieved using message size of $O(n)$ or unbound. The situation of algorithms using small-sized messages is more complex. Barenboim et al. [3] showed a construction of an $O(\log^{k-1} n)$-spanner of size $O(n^{1+1/k})$ and running time of $O(\log^{k-1} n)$. Derbel, Mosbah and Zemmari [9] showed a $(2k-1)$-spanner of optimal size but in $O(n^{1-1/k})$ time. These results use messages of small size. Grossman and Parter [13] showed that a 3-multiplicative-spanner can be computed in $\tilde{O}(1)$ time in the $\mathcal{CONGEST}$ model with size $O(n^{3/2})$. Furthermore, this result works for the weighted case. Their more generalized result is a $(2k-1)$-spanner of size $O(n^{1+\frac{1}{k}})$ with running time of $O(\sqrt{n})$ for some constant $k > 2$ for the unweighted case. Their results are deterministic. Much earlier, Erdos [11] conjectured the same lower bound for the multiplicative spanner case which by many is regarded to be true although still remains unproven. If indeed the Erdos conjecture is correct, then this will also close the search for a linear size small stretch multiplicative spanner for the general case making the existence of such spanners in certain families of graphs that much more interesting.

In this paper we show that a linear size multiplicative spanner not only exists but can also be computed in constant time in graphs with bounded *diversity* in the $\mathcal{CONGEST}$ model. The diversity of a vertex v is the number of maximal

cliques v belongs to in the input graph.

1.1 Our Contribution

In this paper we offer a couple of results, both for the $\mathcal{CONGEST}$ model and thus fitting to work in all three main models of the distributed setting.

The first result is the spanning of the input graph G within $O(K)$ running time using a spanner of size $O(Kn)$ where K denotes the neighborhood independence of G. Our result is deterministic. It is thus clear that for graphs with $K = O(1)$ we have a constant time algorithm that spans the graphs using a linear size spanner. This is helpful in the Congested-Clique model since one can send all the edges of such a spanner to a centralized vertex in the clique using Lenzen Routing and compute a spanning tree locally which is of much interest by itself.

The second result we present is where we wish to achieve a spanner of a bounded stretch. Specifically, we show that an $O(D)$-multiplicative-spanner of $O(D^2n)$ size can be computed deterministically within $O(D^2)$ running time where D denotes the diversity of the input graph. For graphs with bounded diversity, this offers a constant time constant stretch spanner of this type in the $\mathcal{CONGEST}$ model and the first such solution for any family of graphs. If Erdos conjecture is indeed true, and a lower bound of $\Omega(n^{1+\epsilon})$ for some $\epsilon > 0$ exists for all constant stretch multiplicative spanners, then finding families of graphs where it is possible to find linear size small stretch spanner is that much more interesting and our result is the first of this kind. We note that the family of bounded diversity graphs include the line-graphs and the unit-disc-graphs which play important role in the motivation of the distributed setting.

1.2 On Diversity

Diversity is a relatively new graph parameter, introduced in [4]. Although the well-defined parameter was presented in that paper, the concept was already known much earlier in the study of line-graphs in graph theory. It is long known that if G is a line-graph then there exists a clique cover of the vertices of G, such that each vertex belongs to at most 2 cliques. This has been the target of research in various papers in regard to graph th Barenboim, Elkin and Maimon [4] showed that diversity is a useful parameter for solving vertex-coloring and edge-coloring, two of the most well-studied symmetry breaking problems in the distributed setting. Recently, Barenboim and Maimon [5] showed that it is also useful for solving maximal matching and ruling-sets. Maximal matching is another one of the core symmetry breaking problems in this setting. In this paper we add spanners to the list of problems which diversity helps resolve. Furthermore, we do so in the bandwidth-restricted model of $\mathcal{CONGEST}$.

2 Preliminaries

An (α, β)**-spanner** is a subgraph H of the input graph G where for every two vertices v and u we have $d_H(v,u) \leq \alpha \cdot d_G(v,u) + \beta$.
The **neighborhood independence** of a vertex v is the largest independent subset of vertices one can choose from $\Gamma(v)$ where $\Gamma(v)$ is the 1-hop neighborhood of v. Naturally, the neighborhood independence of a graph G is the maximum between all neighborhood independences of all the vertices of G.
The **diversity** [1] of a vertex v is the number of maximal cliques v belongs to in the input graph. The diversity of a graph $G = (V, E)$ is defined as the maximum between diversities of all vertices in V.

3 Spanning a Graph in the $\mathcal{CONGEST}$ model

Let G be a graph in the $\mathcal{CONGEST}$ model which one wants to span with a spanning forest, a tree for each connected component. For graphs with neighborhood independence K we offer a deterministic $O(K)$-time algorithm for constructing a spanning subgraph of size $O(Kn)$. For graphs with constant neighborhood independence, this gives a simple linear size spanner which can be used to solve the connectivity problem in the Congested Clique.

Each vertex initializes a set of vertices $L_v = \Gamma(v)$ and an empty set of edges $\hat{E}_v = \emptyset$. The sets $\hat{E}_v$ are going to be used for storing the solution. The algorithm proceeds to executing K iterations. In each iteration, each vertex v chooses the vertex with the highest ID, denoted as $c(v)$, out of L_v and adds the edge $(v, c(v))$ to $\hat{E}_v$. Then v sends $ID(c(v))$ to all of its neighbors. Each neighbor reports to v if it is connected to $c(v)$. For each neighbor u of v which is connected to $c(v)$, v removes u from L_v. v also removes $c(v)$ from L_v. This concludes the description of the algorithm. After K iterations, v returns $\hat{E}_v$ as the result.
Denote $G' = \{\cup \hat{E}_v | v \in G\}$. Since there are K iterations, each of which adds a single edge to $\hat{E}_v$ for each $v \in V$, we obtain $|G'| = O(Kn)$. The following lemma shows that G' spans G.

Lemma 3.1. *G' spans G.*

Proof. Let (v, u) be an edge in G. Denote $v = v_0, u = u_0$. If for some iteration $c(v_0) = u_0$ then it is clear that $(v_0, u_0) \in G'$. Otherwise, let v_1 be the vertex v_0 selects in some iteration that had u_0 removed from L_{v_0}. Thus, v_1 is a common neighbor of v_0 and u_0. Therefore, $ID(v_1) > ID(u_0)$ and $(v_0, v_1) \in G'$.

Now, either the edge $(v_1, u_0) \in \hat{E}_{u_0}$ and thus in G', or on some iteration u_0 selects a neighbor u_1 that had v_1 removed from L_{u_0}. Thus we have $ID(u_1) > ID(v_1) > ID(u_0)$, and $(u_0, u_1) \in G'$. Now, again, either the edge $(u_1, v_1) \in \hat{E}_{v_1}$ and thus in G', or on some iteration v_1 selects a neighbor v_2 that had u_1

[1] Diversity can be also defined with respect to a clique cover of a given graph. Then, the diversity of a vertex is the number of maximal cliques in the cover that the vertex belongs to. In this paper, however, we do not employ clique covers, and so the diversity is defined as the number of maximal cliques in the input graph.

removed from L_{v_1}. Thus we have $ID(v_2) > ID(u_1) > ID(v_1) > ID(u_0)$ and $(v_1, v_2) \in G'$. This cannot last infinitely. At some point the path of $v_0, v_1, \ldots$ and the path $u_0, u_1, \ldots$ either meet, or there is an edge $(v_i, u_i) \in G'$ or an edge $(u_i, v_{i+1}) \in G'$. In either case we have a path from v to u. Since this is true for every edge in G, we have that G' spans G. □

In the Congested Clique model, it is now possible for each vertex v to send $\hat{E}_v$ to some central vertex. One can thus conclude an $O(K)$ runtime deterministic algorithm for computing a spanning tree of a subgraph G in the Congested Clique. We note that, to the best of our knowledge, ours is the first result which does not depend on n and thus not growing in running time as the number of vertices grow. We summarize the results.

Theorem 3.2. *There is an $O(K)$ runtime deterministic algorithm for computing a spanning skeleton of size $O(Kn)$ in the $\mathcal{CONGEST}$ model.*

Corollary 3.3. *There is an $O(K)$ runtime deterministic algorithm for solving the connectivity problem in the congested clique model by finding a spanning tree of the input subgraph.*

4 A Small Size Small Stretch Spanner in Bounded Diversity Graphs

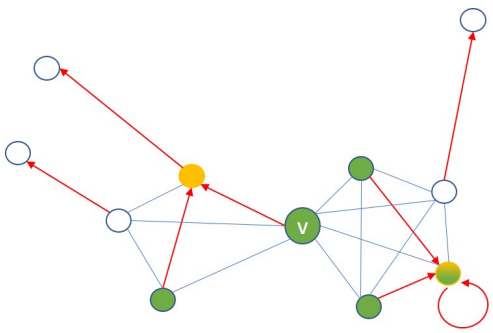
It is also of importance to find a spanning subgraph of a bounded stretch. Even though spanners have been well-studied, even in the congested-clique model, to the best of our knowledge, there is no known algorithm for constructing a spanner of linear size and a small stretch for any family of graphs deterministically. The algorithm we devised in the previous section did not guaranteed a bounded stretch.
A lower bound on the size of a $(2k-1)$-spanner is conjectured for the general case by Erdos [11] and was proven for the additive spanner case [23]. Specifically, the lower bound states that there are graphs for which such spanner requires $\Theta(n^{1+\frac{1}{k}})$ edges. This makes finding families of graphs which allow spanners of linear size and small stretch of much interest.

We start with a simple result to show that in the $\mathcal{LOCAL}$ model diversity is strongly tied to spanning. One can achieve an $O(Dn)$ size 2-spanner within 1 round of communication. This is done as follows. Each vertex shares its 1-hop neighborhood with all its neighbors. Thus, each vertex can compute locally what are the maximal cliques it belongs to and what is the highest ID vertex in each such clique (which we call the *master* of the clique). Each vertex then connects only to the master in each clique. If (v, u) is an edge in G then v and u are connected with a path at most 2 through the master of the clique containing the edge (v, u) (if there are more than one such clique then v and u are surely connected through the master with the highest ID among all the

masters of these cliques). The size of such spanner is $O(Dn)$ as each vertex is connected to at most D masters.

Theorem 4.1. *In the $\mathcal{LOCAL}$ model one can computed a linear size 2-spanner within 1 round of communication in graphs with constant diversity.*

We now show the more interesting result which is that for graphs with diversity D one can find a $(D+1)$-stretch spanner of size $O(D^2n)$ within $O(D^2)$ running time in the $\mathcal{CONGEST}$ model. For simplification, we first show our algorithm in the Congested Clique and later we will show that the all-to-all communication is not required.
We begin each iteration with each vertex v choosing a neighbor with the highest ID, denote $c(v)$, from all vertices it is yet to be connected to in the solution. v sends $c(v)$ to all vertices in G. Let u be a neighbor of v and $c(u)$ the chosen vertex of u. We denote $C = \{c(u) \mid \{u, c(u)\} \in \Gamma(v)\}$ to be all the choices v receives from its neighbors in G such that they, the choices, are also neighbors of v in G. Let P denote all neighbors of v which made the choices in C. That is $P = \{u \mid c(u) \in C\}$. In other words, if a neighbor w of v chose a vertex $c(w)$ which is not neighbor of v, that is $c(w) \notin \Gamma(v)$, then w is not in P. We would now like to connect v to all vertices in P with path of size at most $i+1$ where i denotes the current iteration. We will show that v needs only to choose D vertices out of $C \cup P$ to achieve this. After choosing a subset $\hat{E}$, v broadcasts $\hat{E}$ to all vertices in G. All vertices maintain their own copy of the result so far, denoted R, which represents all the edges accumulated so far. It is this R that each vertex uses to choose a small subset of $C \cup P$. This completes the description of the algorithm. Its pseudocode appears below. We prove its correctness in the following lemmas.



Arrows shows choices. Set C in Orange. Set P in Green.

Algorithm 1 CongestSpanner(G, D)

$S = \Gamma(v)$
$R = \emptyset$
for D iterations **do**
  Choose the highest ID vertex $c(v)$ in S
  Add the edge $(v, c(v))$ to R.
  Send $ID(c(v))$ to all neighbors.
  Remove $c(v)$ from S.
  $C =$ all choices $c(u)$ sent to v by its neighbors such that $c(u) \in \Gamma(v)$.
  $P =$ all neighbors u of v which have a choice in C.
  Choose the smallest subset $T \subseteq C \cup P$ with the following condition: Let $E(T)$ be the set of edges between v and the vertices in T. $E(T) \cup R$ connects v to all vertices in P.
  Send the chosen subset of T to all vertices in G. Add T to R. This requires $|T|$ rounds.
  Remove all vertices in P from S.
end for
return R.

Lemma 4.2. *Let u, w be two neighbors that select $c(u), c(w)$, respectively, such that $u, w, c(u), c(w)$ belong to the same clique. Then, if $c(u) \neq c(w)$ there is a path in R between u and w of length $i + 1$ in the end of iteration i.*

Proof. The proof is by induction on the iteration i.
Base ($i = 1$): in the first iteration, if two vertices in the same clique make choices within that clique, then both select the vertex with the maximum ID in that clique. Consequently, $c(u) = c(w)$ and u is connected to w in R by a path of length 2. **Step:** In iteration i, suppose that u, w make choices in the same clique, but u selects a neighbor $c(u)$ with a smaller ID than that of the selection $c(w)$ of w. This means that u already has a path in R to $c(w)$. This path was computed in a previous iteration, and so its length is at most i, by induction hypothesis. Hence, by the end of iteration i, the vertex u is connected to w through $c(w)$ by a path of length at most $i + 1$. □

Lemma 4.3. *At each iteration i, there is a subset in $C \cup P$ of size at most D which v can choose that connects v to each $u \in P$ with a path of size at most $i + 1$.*

Proof. Denote $L = G(C \cup P)$ as the induced subgraph of G over the subset of vertices $C \cup P$ in iteration i. The diversity of L is less than or equals that of G since each distinct clique in L is a sub-clique of a distinct clique in G. Thus, v can belong to at most D distinct maximal cliques in L.
Let Q be a clique containing v. Denote all pairs of neighbors $u, c(u)$ which are in Q as $(u_1, c(u_1), \ldots, (u_q, c(u_q)))$. Note that P consists of $u_1, u_2, \ldots, u_q$. W.L.O.G, assume that $ID(c(u_1)) \leq \ldots \leq ID(c(u_q))$. Then by Lemma 4.2, for each $1 \leq j \leq q$ there is a path of length at most i between u_j and $c(u_q)$. Thus

by connecting v to $c(u_q)$ we obtain paths from v to all the vertices $u_1, \ldots, u_q$ of length at most $i+1$ each. This means we require only a single additional edge, $(v, c(u_q))$, per clique which v belongs to in L. Hence at most D edges per vertex per iteration. □

Since Lemma 4.3 ensures the existence of a subset of small size which connects v to all of its neighbors in P, surely v can find this subset locally and use it or maybe a smaller size subset. (Recall that in each iteration, v is aware of the entire solution R for G in that stage, since the up-to-date information about R is made available to all vertices in the congested clique.)

In the next lemmas we shows that each iteration of our algorithm makes a certain distinct clique of a vertex v to be spanned by a subgraph of bounded diameter. Consequently, within D iteration each vertex has paths of bounded length in R to all its neighbors in G. We prove this by analyzing the clique of v that contains the edge $(v, c(v))$ of the choice of v in that iteration.

Lemma 4.4. *Let Q be a maximal clique containing the edge $(v, c(v))$ in iteration i. Then, within D rounds (performed during the execution of line 11), in the output R there is a path from v to any vertex in Q of length at most $i+1$.*

Proof. Let u be a vertex in Q. Since $(v, c(v)) \in Q$ then u is connected to v in G and also is connected to $c(v)$ in G. Hence, $v \in P$ as computed by u. By Lemma 4.3, u connects to all vertices in P. Specifically, u makes sure to connect to v with a path of length at most $i+1$. □

Lemma 4.5. *For $i = 1, 2, \ldots, D$, within i iterations (of the outer loop), there are paths in the output R between v and all vertices of i cliques containing v.*

Proof. Denote $c_i(v)$ as the choice v makes at iteration i. Denote $Q_1, \ldots, Q_{i-1}$ the maximal cliques which contain the edges $(v, c_1(v)), \ldots, (v, c_{i-1}(v))$ respectively. By Lemma 4.4, at each iteration of the outer loop the output contains a path between v and all of its neighbors in at least one maximal clique containing v. At each following iteration, v chooses the vertex with the highest ID $c(v)$ such that v is not yet connected to in the output. Hence, the edge $(v, c_i(v))$ cannot belong to any of the cliques $Q_1, \ldots, Q_{i-1}$ since v already knows of a connection to all vertices in these cliques. Thus, a different distinct maximal clique Q_i must contain the edge $(v, c_i(v))$. Again, by Lemma 4.4, the entire clique Q_i will be connected to v at iteration i. Hence, within i iterations (of the outer loop), there are paths in the output between v and all vertices of i distinct maximal cliques containing v. Specifically, after D such iterations, v will be left with no maximal cliques to choose a neighbor from, and will be connected to all of its neighbors. □

Since at each inner round each vertex adds at most D edges to the solution and since there are D outer iterations, we conclude with the following result.

Theorem 4.6. *There is an $O(D^2)$ running time deterministic algorithm for computing a spanner of size $O(D^2 n)$ and stretch at most $D+1$.*

Now we note that the information used by a vertex v in order to decide on a subset $\hat{E}$ of the graph L is weather or not a neighbor u which sent a choice $c(u)$ is already connected to some vertex $c(w)$, a vertex chosen by w, another neighbor of v, in a clique in L that contains the vertices $u, c(u), w, c(w)$ and v. In the Congested Clique model, this knowledge can be aggregated at v in each iteration. This is not necessary as v can query u about the reason why it chose $c(u)$ instead $c(w)$ in case their IDs are different; is it because u is not connected to $c(w)$ in G or is it because u is already connected to $c(w)$ in the output. To achieve this, each vertex is required to register locally with which neighbors it is already connected to in the output. Unlike before, at each iteration i, R will contain the neighbors v is connected to in the output with a path at most $i+1$. Another difference is that v is now required to build up the subset $\hat{E}$ edge by edge for $O(D)$ rounds depending on the responses it gets from querying. (Note that we refer to this inner loop as rounds and to the outer loop as iterations.) It will first choose the vertex with highest ID from C, $c(w)$, and then query all vertices in P whether they are connected to $c(w)$ already. For all those who are, v will remove their choices from C, register them in R as connected to v with a path of length at most $i+1$ as v adds the edge $(v, c(w))$ to $\hat{E}$, and choose the next available vertex with the highest ID in C. v needs to repeat this for at most D rounds until we can be sure it is connected to all vertices in P with paths of length at most $i+1$ in accordance to Lemma 4.3. We provide the pseudocode of this variant for the $\mathcal{CONGEST}$ model in Algorithm 2 below. We summarize this discussion with the next theorem.

Theorem 4.7. *There is an $O(D^2)$ runtime deterministic algorithm for computing a spanner of size $O(D^2 n)$ and stretch at most $D+1$ in the $\mathcal{CONGEST}$ model.*

Algorithm 2 CongestSpanner(G, D)

$S = \Gamma(v)$
$R = \emptyset$
for D iterations **do**
  Choose the highest ID vertex $c(v)$ in S
  Add the edge $(v, c(v))$ to R.
  Send $ID(c(v))$ to all neighbors.
  Remove $c(v)$ from S.
  $C =$ all choices $c(u)$ sent to v by its neighbors such that $c(u) \in \Gamma(v)$.
  $P =$ all neighbors u of v which have a choice in C.
  for D rounds **do**
    Choose the highest ID vertex w from C.
    Remove w from S and from C.
    Add (v, w) to R.
    Query all vertices in P if they already registered w as connected to them in their solution.
    For each neighbor u which replied 'yes', remove u from S and from P. Register that v is connected to u.
    $C =$ all choices of vertices which are still in P.
    Register that v is connected to w in the solution R.
  end for
end for
return R.

4.1 Conclusion

As we showed in this paper, diversity helps to efficiently find a linear size 2-spanner in the $\mathcal{LOCAL}$ model as well as a linear size small stretch spanner in the $\mathcal{CONGEST}$ model for graphs with bounded diversity. As noted these results are added to the efficiency of distributed algorithms for graphs with bounded diversity for well-studied problems.

Hence we believe that bounded diversity graphs are indeed an interesting family of graphs as this family is not a trivial one and yet has deterministic poly-logarithmic solutions for many of the well-studied problems in the distributed setting. Currently, out of the four major symmetry breaking problems in this setting, only MIS (Maximal Independent Set) remains open without such a solution for graphs with bounded diversity. We believe it would be of interest to find such a solution.

References

[1] I. Abraham, C. Gavoille, D. Malkhi On space-stretch trade-offs: upper bounds. *SPAA '06*, 217-224, 2006.

[2] I. Abraham, C. Gavoille, D. Malkhi, N. Nisan, M. Thorup Compact name-independent routing with minimum stretch. *SPAA '04*, 20-24, 2004.

[3] L. Barenboim, M. Elkin, C. Gavoille. A fast network-decomposition algorithm and its applications to constant-time distributed computation. *SIROCCO '15*, 209-223, 2015.

[4] L. Barenboim, M. Elkin, T. Maimon. Deterministic Distributed $(\Delta+o(\Delta))$-Edge-Coloring, and Vertex-Coloring of Graphs with Bounded Diversity. *PODC '17*, 175-184, 2017.

[5] L. Barenboim, T. Maimon Distributed Symmetry Breaking in Graphs with Bounded Diversity. To appear in IPDPS 2018.

[6] B. Derbel, C. Gavoille, D. Peleg Deterministic distributed construction of linear stretch spanners in polylogarithmic time. *DISC '07*, 179-192, 2007.

[7] B. Derbel, C. Gavoille, D. Peleg, L. Viennot On the locality of distributed sparse spanner construction. *PODC '08*, 273–282, 2008.

[8] B. Derbel, C. Gavoille, D. Peleg, L. Viennot Local computation of nearly additive spanners. *DISC '09*, 176–190, 2009.

[9] B. Derbel, M. Mosbah, A. Zemmari Sublinear fully distributed partition with applications. *Theory of Computing Systems '10*, 47(2):368–404, 2010.

[10] M. Elkin Computing almost shortest paths. *ACM Algorithms1(2) '05*, 283-323, 2005.

[11] P. Erdos Extremal problems in graph theory. *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963)*, 2936, 1963.

[12] M. Elkin, O. Neiman Efficient Algorithms for Constructing Very Sparse Spanners and Emulators. *SODA '17*, 652-669, 2017.

[13] O. Grossman, M. Parter Improved Deterministic Distributed Construction of Spanners. *DISC '17*, 24:1-24:16, 2017.

[14] J.W. Hegeman, G. Pandurangan, S.V. Pemmaraju, V.B. Sardeshmukh, M. Scquizzato Toward optimal bounds in the congested clique: Graph connectivity and MST. *PODC '15*, 91–100, 2015.

[15] T. Jurdzinski, K. Nowicki MST in $O(1)$ Rounds of the Congested Clique. https://arxiv.org/abs/1707.08484, 2017.

[16] C. Lenzen Optimal Deterministic Routing and Sorting on the Congested Clique. *PODC '13*, 42–50, 2013.

[17] Z. Lotker, E. Pavlov, B. Patt-Shamir, D. Peleg MST construction in $O(\log\log n)$ communication rounds. *SPAA '03*, 94-100, 2003.

[18] J.H. Korhonen Deterministic MST Sparsification in the Congested Clique. https://arxiv.org/pdf/1605.02022.pdf, 2016.

[19] D. Peleg Distributed computing: a locality-sensitive approach. *SIAM 2000*.

[20] S. Pettie Distributed algorithms for ultrasparse spanners and linear size skeletons. *PODC '08*, 253-262, 2008.

[21] M. Thorup, U. Zwick Compact routing schemes. *SPAA '01*, 1-10, 2001.

[22] M. Thorup, U. Zwick Approximate distance oracles. *ACM 52(1) '05*, 1-24, 2005.

[23] D.P. Woodruff Lower Bounds for Additive Spanners, Emulators, and More. *FOCS '06*, 389-398, 2006.