Graph Connectivities, Network Coding, and Expander Graphs

Ho Yee Cheung, Lap Chi Lau, Kai Man Leung
The Chinese University of Hong Kong

Abstract— We present a new algebraic formulation to compute edge connectivities in a directed graph, using the ideas developed in network coding. This reduces the problem of computing edge connectivities to solving systems of linear equations, thus allowing us to use tools in linear algebra to design new algorithms. Using the algebraic formulation we obtain faster algorithms for computing single source edge connectivities and all pairs edge connectivities, in some settings the amortized time to compute the edge connectivity for one pair is sublinear. Through this connection, we have also found an interesting use of expanders and superconcentrators to design fast algorithms for some graph connectivity problems.

1. Introduction

Graph connectivity is a basic concept that measures the reliability and efficiency of a graph. The edge connectivity from vertex s to vertex t is defined as the size of a minimum s-t cut, or equivalently the maximum number of edge disjoint paths from s to t. Computing edge connectivities is a classical and well-studied problem in combinatorial optimization. Most known algorithms to solve this problem are based on network flow techniques (see e.g. [33]).

The fastest algorithm to compute the s-t edge connectivity in a simple directed graph is a $O(\min\{m^{1/2}, n^{2/3}\} \cdot m)$ time algorithm by Even and Tarjan [11], where m is the number of edges and n is the number of vertices. To compute the edge connectivities for many pairs, however, it is not known how to do it faster than computing the edge connectivity for each pair separately, even when the pairs share the source or the sink. For instance, it is not known how to compute all pairs edge connectivities faster than computing the s-t edge connectivity for $\Omega(n^2)$ pairs. This is in contrast to the problem in undirected graphs, where all pairs edge connectivities can be computed in $\tilde{O}(mn)$ time by constructing a Gomory-Hu tree [6].

Network coding is an innovative method to transmit information in a computer network. The fundamental result is a max-information-flow min-cut theorem for multicasting [1]: if the edge connectivity between the source vertex s to each sink vertex t_i is at least k, then one can transmit k units of information to all sink vertices simultaneously, by performing encoding and decoding at the vertices. An elegant algebraic framework has been developed to construct efficient network coding schemes for multicasting [28], [25].

We use the techniques developed in network coding to obtain a new algebraic formulation for computing edge connectivities. This reduces the problem of computing edge connectivities to solving systems of linear equations, and opens up new directions to design algorithms for the problem. One advantage is that the edge connectivities from a source vertex to all other vertices can be computed simultaneously (as in the max-information-flow min-cut theorem). This leads to faster algorithms for computing single source edge connectivities and all pairs edge connectivities, and in some settings the amortized time to compute the edge connectivity for one pair is sublinear. In the process we have also found an interesting use of expanders and superconcentrators to design fast algorithms for graph connectivity problems.

Research supported by HK RGC grant 413609 and 413411.

1.1. Our Results

Our new algebraic formulation for computing edge connectivities is inspired by the random linear coding algorithm [17] in constructing network codes. Let $G = (V, E)$ be a directed graph. Let s be the source vertex with out-degree d, with outgoing edges $e_1, \ldots, e_d$. For each edge $e \in E$ we associate a vector f_e of dimension d, where each entry in f_e is an element from a large enough finite field $\mathbb{F}$. The vectors are required to satisfy the following properties: (1) the vectors on $e_1, \ldots, e_d$ are linearly independent, and (2) the vector on an edge $e = (v, w)$ is a random linear combination of the incoming vectors of v. Once we obtain the vectors, we can compute the edge connectivities from the source vertex as follows.

Theorem 1.1 (Informal statement) *With high probability there is a unique solution to the vectors, and the edge connectivity from s to t is equal to the rank of the incoming vectors of t for any $t \in V - s$.*

See Figure 1.1 for an example and Theorem 2.1 for the formal statement. This formulation is previously known for directed acyclic graphs only [21], [34]. For general directed graphs, network coding schemes require convolution codes [10], [27], and it is not known how to use these to compute edge connectivities. Our contribution is to show that this simpler formulation can be used to compute edge connectivities.

The algebraic formulation reduces the problem of computing edge connectivities to solving systems of linear equations. We call the step to compute the vectors as the *encoding* step, and the step to compute the ranks as the *decoding* step. The formulation has the advantage that after the encoding

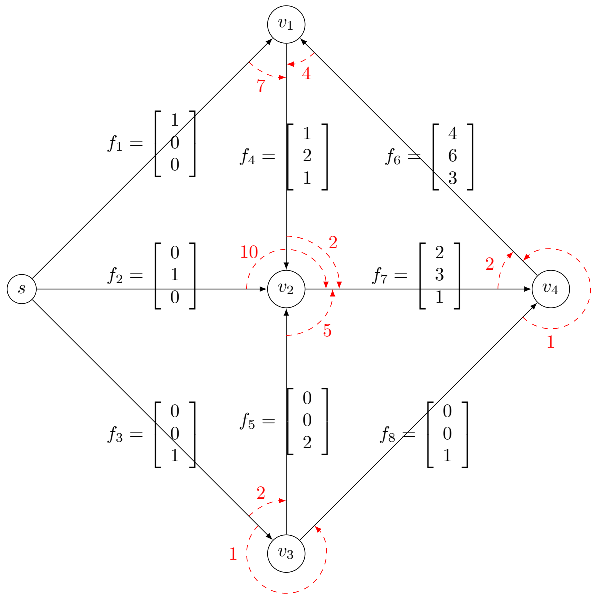

Figure 1.1: In this example three independent vectors f_1, f_2 and f_3 are sent from the source s. Other vectors are a linear combination of the incoming vectors, according to the random coefficients on the dotted lines, e.g. $f_4 = 7 \cdot f_1 + 4 \cdot f_6$ and $f_7 = 2 \cdot f_4 + 10 \cdot f_2 + 5 \cdot f_5$, etc. All operations are done in the field of size 11. To compute the edge connectivity from s to v_2, for instance, we compute the rank of (f_2, f_4, f_5) which is 3 in this example.

step has been done, the edge connectivities from the source vertex s to all other vertices can be computed readily. In the following we will first present algorithmic results on single source edge connectivities and then present algorithmic results on all pairs edge connectivities.

In directed acyclic graphs, the encoding step can be implemented directly without solving linear equations. By a simple transformation using superconcentrators [35], [18], we show how to implement the encoding step optimally. This implies a faster algorithm for computing single source edge connectivities in directed acyclic graphs.

Theorem 1.2 *Given a simple directed acyclic graph and a source vertex s with outdegree d, the encoding step can be done in $O(dm)$ time, and the edge connectivities from s to all vertices can be computed in $O(d^{\omega-1}m)$ time, where $\omega \approx 2.38$ is the matrix multiplication exponent.*

The algorithm runs in linear time when d is a constant, and by a simple reduction this can be used to return all vertices with edge connectivity at most d from the source s in linear time. We note that the encoding algorithm can also be used to improve the random linear coding algorithm [17] for network coding.

In some graphs the system of linear equations can be solved more efficiently. For instance, we can use a recent result of Alon and Yuster [3] to perform the encoding step faster in directed planar graphs with constant maximum degree. The best known algorithm to compute single source edge connectivities in directed planar graphs requires $O(n^2)$ time, by using a $O(n)$ time algorithm to compute s-t edge connectivity [8].

Theorem 1.3 *Given a simple directed planar graph with constant maximum degree and a source vertex s, the edge connectivities from s to all vertices can be computed in $O(n^{\omega/2})$ time.*

For general directed graphs, we show that all pairs edge connectivities can be computed in one matrix inverse time, instead of solving linear equations for each source vertex separately. The algorithm is faster when $m = O(n^{1.93})$, for example when $m = O(n)$ it takes $O(n^{\omega})$ time while the best known algorithm takes $O(n^{3.5})$ time.

Theorem 1.4 *Given a simple directed graph, the edge connectivities between all pairs of vertices can be computed in $O(m^{\omega})$ time where m is the number of edges in the graph.*

We show that the matrix inverse can be computed more efficiently for graphs that are "well-separable" (which will be defined in Section 4.2), including planar graphs, bounded genus graphs, fixed minor free graphs, etc. The resulting algorithm is faster than that for general graphs when the maximum degree is $O(\sqrt{n})$.

Theorem 1.5 *Given a simple well-separable directed graph with maximum degree d, the edge connectivities between all pairs of vertices can be computed in $O(d^{\omega-2}n^{\omega/2+1})$ time.*

The idea of transforming the graph using superconcentrators can also be used in other graph connectivity problems. In Section 5 we show how to use expanders and superconcentrators to speedup the algorithms for finding edge splitting-off operations preserving edge connectivities in undirected and directed graphs.

1.2. Related Work

The standard way to solve the s-t edge connectivity problem in directed graphs is by network flow techniques. For simple directed graphs, the best known algorithm is a $O(\min\{n^{2/3}, m^{1/2}\} \cdot m)$ time algorithm [11] by the blocking flow method. As mentioned previously, in directed graphs, it is not known how to compute all pairs edge connectivities faster than computing s-t edge connectivity for $\Omega(n^2)$ pairs separately. This is in contrast to the problem in undirected graphs, where all pairs edge connectivities can be computed in $\tilde{O}(mn)$ time by constructing a Gomory-Hu tree [6], much faster than computing edge connectivities for $\Omega(n^2)$ pairs.

There are improvements in special cases of the s-t edge connectivity problem in directed graphs. For bipartite matching, the best known algorithm is a $O(m\sqrt{n})$ time algorithm [19] by the blocking flow method, and a $O(n^{\omega})$ time algorithm [32], [16] by algebraic techniques. It is known that the bipartite matching problem is equivalent to the s-t vertex connectivity problem in directed graphs, and so the above results hold for the latter problem as well [9]. For simple undirected graphs, there is a $O(n^{3/2}\sqrt{m})$ time algorithm [15] and a $\tilde{O}(n^{2.2})$ time algorithm [23]. In directed planar graphs, there is an optimal $O(n)$ time algorithm for computing s-t edge connectivity [8].

For network coding, the max-information-flow min-cut theorem for multicasting is first proved by an information theoretical argument [1]. Later it is shown that linear network coding is enough to achieve the max-flow min-cut theorem for multicasting [28] and an algebraic framework is developed [25]. Then a polynomial time deterministic algorithm is obtained to construct optimal linear coding schemes for multicasting in directed acyclic graphs [22], and in general directed graphs using convolution codes [10], [27]. Subsequently a simple polynomial time randomized algorithm is obtained for constructing optimal linear coding schemes for multicasting [17], and our algorithm is based on this approach.

1.3. Techniques

The starting observation is that the random linear coding algorithm [17] for network coding does not require the knowledge of the graph topology, and it could be used to compute the edge connectivities from the source to the sinks. Actually this observation was already made for directed acyclic graphs in earlier work [21], [34]. For general directed graphs, however, network coding schemes are more complicated (even for random linear coding) as convolution codes are required [10], [27], and it is not known how to use these to compute edge connectivities. Our contribution is to show that a simpler formulation can be used to compute edge connectivities, which also allows us to design more efficient algorithms. The proof is based on the ideas developed in the random linear coding algorithm.

We show a simple transformation using expanders and superconcentrators [35], [18] to design fast algorithms for some graph connectivity problems. In directed graphs, the idea is to replace a vertex of indegree d and outdegree d by a superconcentrator with d inputs and d outputs. This reduces the maximum degree of the graph significantly, while preserving the edge connectivities and only increasing the number of vertices moderately. For random linear coding, using the direct algorithm in the resulting graph gives an optimal algorithm in the original graph. For edge splitting-off, using the straightfoward algorithm in the resulting graph gives a considerable improvement upon the same algorithm in the original graph. In undirected graphs, we can use constant degree expander graphs for the same purpose.

For all pairs edge connectivities, we observe that if we change the source vertex the system of linear equations is similar, and so we could compute the n single source edge connectivities in one matrix inverse time. For graphs with good separators, we show how to compute the inverse faster by a divide and conquer algorithm, using Schur's formula and the Sherman-Morrison-Woodbury formula, which may be of independent interest. We note that the algorithmic results on all pairs edge connectivities can also be derived from the matrix formulation given by Ingleton and Piff [20], [9] for vertex connectivities.

1.4. Organization

We present the algebraic formulation in Section 2 and prove a formal statement of Theorem 1.1. In Section 3 we present algorithms for single source edge connectivities and prove Theorem 1.2 and Theorem 1.3. In Section 4 we present algorithms for all pairs edge connectivities and prove Theorem 1.4 and Theorem 1.5. Finally we show the use of expanders and superconcentrators in the edge splitting-off problem in Section 5.

2. Algebraic Formulation

Throughout the paper we consider uncapacitated directed graphs where each edge is of the same capacity. We begin with some notation and definitions for graphs and matrices. In a directed graph $G = (V, E)$, an edge $e = (u, v)$ is directed from u to v, and we say that u is the tail of e and v is the head of e. For any vertex $v \in V$, we define $\delta^{in}(v) = \{uv \mid uv \in E\}$ as the set of incoming edges of v and $d^{in}(v) = |\delta^{in}(v)|$; similarly we define $\delta^{out}(v) = \{vw \mid vw \in E\}$ as the set of outgoing edges of v and $d^{out}(v) = |\delta^{out}(v)|$. For a subset $S \subseteq V$, we define $\delta^{in}(S) = \{uv \mid u \notin S, v \in S, uv \in E\}$ and $d^{in}(S) = |\delta^{in}(S)|$. Given a matrix M, the submatrix containing rows S and columns T is denoted by $M_{S,T}$. A submatrix containing all rows (or columns) is denoted by $M_{*,T}$ (or $M_{S,*}$), and an entry of M is denoted by $M_{i,j}$.

We define the algebraic formulation for computing graph connectivities formally. Given a directed graph $G = (V, E)$ and a specified source vertex s, we are interested in computing the edge connectivities from s to other vertices. Let $m = |E|$ and $E = \{e_1, e_2, \ldots, e_m\}$. Let $d = d^{out}(s)$ and $\delta^{out}(s) = \{e_1, \ldots, e_d\}$. Let $\mathbb{F}$ be a finite field. For each edge $e \in E$, we associate a *global encoding vector* $f_e \in \mathbb{F}^d$ of dimension d where each entry is in $\mathbb{F}$. We say a pair of edges e' and e are *adjacent* if the head of e' is the same as the tail of e, i.e. $e' = (u, v)$ and $e = (v, w)$ for some $v \in V$. For each pair of adjacent edges e' and e, we associate a *local encoding coefficient* $k_{e',e} \in \mathbb{F}$. Given the local encoding coefficients for all pairs of adjacent edges in G, we say that the global encoding vectors are a *network coding solution* if the following two sets of equations are satisfied:

1) For each edge $e_i \in \delta^{out}(s)$, we have $f_{e_i} = \sum_{e' \in \delta^{in}(s)} k_{e',e_i} \cdot f_{e'} + \vec{e}_i$, where $\vec{e}_i$ is the i-th vector in the standard basis.
2) For each edge $e = (v, w)$ with $v \neq s$, we have $f_e = \sum_{e' \in \delta^{in}(v)} k_{e',e} \cdot f_{e'}$.

The main theorem in this section is the following formal statement of Theorem 1.1.

Theorem 2.1 *Let $\mathbb{F}$ be a finite field with $|\mathbb{F}| = \Omega(m^{c+2})$ for some integer c. If we choose each local encoding coefficient independently and uniformly at random from $\mathbb{F}$, then with probability at least $1 - O(1/m^c)$:*

1) *There is a unique network coding solution for the global encoding vectors.*
2) *For $t \in V - s$, let $\delta^{in}(t) = \{a_1, \ldots, a_l\}$, the edge connectivity from s to t is equal to the rank of the matrix $(f_{a_1}, f_{a_2}, \ldots, f_{a_l})$.*

We will prove Theorem 2.1 in the remaining of this section. First we rewrite the requirements of a network coding solution in a matrix form. Let F be the $d \times m$ matrix $(f_{e_1}, \ldots, f_{e_m})$. Let K be the $m \times m$ matrix where $K_{i,j} = k_{e_i,e_j}$ when e_i and e_j are adjacent edges, and 0 otherwise. Let H_s be the $d \times m$ matrix $(\vec{e}_1, \ldots, \vec{e}_d, \vec{0}, \vec{0}, \ldots, \vec{0})$ where $\vec{0}$ denotes the all zero vector of dimension d. Then the network coding equations are equal to the matrix equation $F = FK + H_s$.

To prove the first part of Theorem 2.1, we will prove in the following lemma that $(I - K)$ is non-singular with high probability. Then the above equation can be rewritten as $F = H_s(I - K)^{-1}$, which implies that the global encoding vectors are uniquely determined.

Lemma 2.2 *Given the conditions in Theorem 2.1, the matrix $(I - K)$ is non-singular with probability at least $1 - O(1/m^{c+1})$.*

Proof: Since the diagonal entries of K are zero, the diagonal entries of $I - K$ are all one. By treating each entry of K as an indeterminant $K_{i,j}$, it follows that $\det(I - K) = 1 + p(\cdots, K_{i,j}, \cdots)$ where $p(\cdots, K_{i,j}, \cdots)$ is a polynomial of the indeterminates with total degree at most m. Note that $\det(I - K)$ is not a zero polynomial since there is a constant term. Hence, by the Schwartz-Zippel Lemma, if $|\mathbb{F}| = \Omega(m^{c+2})$ and each $K_{i,j}$ is a random element in $\mathbb{F}$, then $\det(I - K) = 0$ with probability at most $O(1/m^{c+1})$, proving the lemma. ∎

After we obtained the global encoding vectors, we would like to show that the edge connectivities can be determined from the ranks of these vectors. Consider a vertex $t \in V - s$. Let $\delta^{in}(t) = \{a_1, \ldots, a_l\}$ and let M_t be the $d \times l$ matrix $(f_{a_1}, f_{a_2}, \ldots, f_{a_l})$. Let the edge connectivity from s to t be $\lambda_{s,t}$. We prove in the following lemma that $\operatorname{rank}(M_t) = \lambda_{s,t}$ with high probability.

Lemma 2.3 *Given the conditions of Theorem 2.1, we have $\operatorname{rank}(M_t) = \lambda_{s,t}$ with probability at least $1 - O(1/m^{c+1})$.*

Proof: First we prove that $\operatorname{rank}(M_t) \leq \lambda_{s,t}$ with high probability. The plan is to show that the global encoding vector on each incoming edge of t is a linear combination of the global encoding vectors in a minimum s-t cut with high probability. Consider a minimum s-t cut $\delta^{in}(T)$ where $T \subset V$ with $s \notin T$ and $t \in T$ and $d^{in}(T) = \lambda_{s,t}$. Let $E' = \{e'_1, \ldots, e'_{m'}\}$ be the set of edges in E with their heads in T. Let $\lambda = \lambda_{s,t}$ and assume $\delta^{in}(T) = \{e'_1, \ldots, e'_\lambda\}$. Let F' be the $d \times m'$ matrix $(f_{e'_1}, \ldots, f_{e'_{m'}})$. Let K' be the $m' \times m'$ submatrix of K restricted to the edges in E'. Let H' be the $d \times m'$ matrix $(f_{e'_1}, \ldots, f_{e'_\lambda}, \vec{0}, \ldots, \vec{0})$. Then, by the network coding requirements, the matrices satisfy the equation $F' = F'K' + H'$. By the same argument as in Lemma 2.2, the matrix $(I - K')$ is nonsingular with probability at least $1 - O(1/m^{c+1})$. So the above matrix equation can be rewritten as $F' = H'(I - K')^{-1}$. This implies that every global encoding vector in F' is a linear combination of the global encoding vectors in H', which are the global encoding vectors in the cut $\delta^{in}(T)$. Therefore the $\operatorname{rank}(M_t) \leq d^{in}(T) = \lambda_{s,t}$.

Next we prove that $\operatorname{rank}(M_t) \geq \lambda_{s,t}$ with high probability. The plan is to show that there is a $\lambda \times \lambda$ submatrix M'_t of M_t such that $\det(M'_t)$ is a non-zero polynomial of the local encoding coefficients with small total degree. First we use the edge disjoint paths from s to t to define M'_t. Let $\lambda = \lambda_{s,t}$ and $P_1, \ldots, P_\lambda$ be a set of λ edge disjoint paths from s to t. Set $k_{e',e} = 1$ for every pair of adjacent edges $e', e \in P_i$ for every i, and set all other local encoding coefficients to be zero. Then each path sends a distinct unit vector of the standard basis to t, and thus M_t contains a $\lambda \times \lambda$ identity matrix as a submatrix. Call this $\lambda \times \lambda$ submatrix M'_t. Now we show that the $\det(M'_t)$ is a non-zero polynomial of the local encoding coefficients with small total degree. Recall that $F = H(I - K)^{-1}$. By considering the adjoint matrix of $I - K$, each entry of $(I - K)^{-1}$ is a polynomial of the local encoding coefficients with total degree m divided by $\det(I - K)$, and thus the same is true for each entry of F. Hence $\det(M'_t)$ is a degree λm polynomial of the local encoding coefficients divided by $(\det(I - K))^\lambda$. By using the edge disjoint paths $P_1, \ldots, P_\lambda$, we have shown that there is a choice of the local encoding coefficients so that $\operatorname{rank}(M'_t) = \lambda$. Thus $\det(M'_t)$ is a non-zero polynomial of the local encoding coefficients. Conditioned on the event that $\det(I - K)$ is nonzero, the probability that $\det(M'_t)$ is nonzero is at most $O(\lambda/m^{c+2}) \leq O(1/m^{c+1})$ by the Schwarz-Zippel lemma. By bounding the probability that $\det(I - K) = 0$ using Lemma 2.2, we obtain that the probability that $\det(M'_t) = 0$ is at most $O(1/m^{c+1})$. This implies that M'_t is a full rank submatrix with high probability, and thus $\operatorname{rank}(M_t) \geq rank(M'_t) = \lambda$. We conclude that $\operatorname{rank}(M_t) = \lambda_{s,t}$ with probability at least

$1 - O(1/m^{c+1})$ by the union bound. ■

Thus the probability that $\text{rank}(M_t) \neq \lambda_{s,t}$ for some $t \in V - s$ is at most $n \cdot O(1/m^{c+1}) \leq O(1/m^c)$ by the union bound, and this proves the second part of Theorem 2.1. Therefore, we only need to set c to be a constant to guarantee a high probability result, and so each field operation can be done in $O(\log m)$ time.

3. Single Source Edge Connectivities

The algebraic formulation can be used to compute the edge connectivities from one source vertex to all other vertices. In general the encoding step requires solving systems of linear equations with m variables and m equations. In this section we will show how to obtain faster algorithms for directed acyclic graphs and directed planar graphs.

3.1. Directed Acyclic Graphs

In directed acyclic graphs, one can compute the global encoding vectors directly by following the topological ordering of the graph. We can first preprocess the graph by a breadth first search to remove all vertices that are not connected from the source vertex. Then the source vertex s is the only vertex with indegree zero, and we set the global encoding vectors of its outgoing edges to be the vectors in the standard basis, as required by the first condition for the network coding solution. We assign random local encoding coefficients to each pair of adjacent edges. Then we process the remaining vertices following the topological ordering of the graph, so that when each vertex v is processed all the vectors of its incoming edges are already computed. For each outgoing edge of v, we compute its vector by taking a linear combination of the vectors of the incoming edges of v, according to the local encoding coefficients. It is easy to see that the global encoding vectors of all edges will be computed, and the resulting vectors satisfy the second requirement of the network coding solution.

We consider efficient implementations of this algorithm. Let d be the outdegree of the source vertex s. So each global encoding vector is of dimension d. For a vertex v with indegree $d^{in}(v)$, it requires $d \cdot d^{in}(v)$ arithmetic operations to compute the vector of one outgoing edge of v, and thus it requires $d \cdot d^{in}(v) \cdot d^{out}(v)$ operations to compute the vectors of all outgoing edges of v. Therefore the total encoding time of this straightforward implementation is $\sum_{v \in V-s} d \cdot d^{in}(v) \cdot d^{out}(v)$, and in worst case it requires $\Theta(dnm)$ steps. An improvement is to use a fast rectangular matrix multiplication algorithm to compute all the outgoing vectors of a vertex, but we will show how to do even faster. Observe that the encoding operation is much faster if the indegree of a vertex is a constant. So the idea is to transform the graph into a bounded degree graph, and it turns out that superconcentrators give us an optimal transformation for this purpose. In the following we will have a short introduction on expanders and superconcentrators, and then come back to the algorithm for directed acyclic graphs.

3.1.1. Expander Graphs and Superconcentrators: An expander graph is a sparse graph that exhibits strong connectivity properties. Given $S \subseteq V$, we define $N(S)$ as the set of vertices in $V - S$ with a neighbor in S.

Definition 3.1 *A graph $G = (V, E)$ is called an (n, d, c)-expander if it has n vertices, the maximum degree is d, and for all $S \subset V$ with $|S| \leq |V|/2$, we have $|N(S)| \geq c|S|$ and c is called the expansion of G.*

There are several explicit constructions for expander graphs with d and c constants (see [18]). Superconcentrators are first defined by Valiant [35] for studying the complexity of linear transformations.

Definition 3.2 ([18]) *Let $G = (V, E)$ be a directed graph and let I and O be two disjoint subsets of V. We say that G is a superconcentrator if for every k and every $S \subseteq I$ and $T \subseteq O$ with $|S| = |T| = k$, there exist k vertex disjoint paths from S to T.*

Valiant proved that there exist superconcentrators with $O(n)$ edges, see [18] for a simple recursive construction using bipartite expander graphs. There exist explicit constructions [12] of superconcentrators with the following properties: (1) there are $O(n)$ vertices and $O(n)$ edges, (2) the maximum indegree and the maximum outdegree are constants, (3) the graphs are directed acyclic. The construction in [12] can be implemented in $O(n)$ time for a superconcentrator with n inputs and n outputs.

3.1.2. Proof of Theorem 1.2: To improve the encoding time, we first transform the graph $G = (V, E)$ by replacing each vertex in $V - s$ by a superconcentrator. For each vertex v, let $d_v = \max\{d^{in}(v), d^{out}(v)\}$, we replace v by a superconcentrator Γ_v with $|I| = |O| = d_v$, and "rewire" each incoming edge of v to a distinct vertex in I and each outgoing edge of v from a distinct vertex in O. The total transformation time is $\sum_{v \in V} O(d_v) = \sum_{v \in V} O((d^{in}(v) + d^{out}(v))) = O(m)$ where $n = |V|$ and $m = |E|$.

Call the graph after the transformation G'. A key point is that the edge connectivity from the source s to a vertex t in G is equal to the edge connectivity from the source s to Γ_t in G' by thinking of Γ_t as a node. To see this, any set of edge disjoint paths from s to t in G corresponds to a set of edge disjoint paths from s to Γ_t in G', because paths sharing a vertex v in G can be routed using the disjoint paths guaranteed by the superconcentrator Γ_v in G'.

By the properties described in Section 3.1.1, G' is a directed acyclic graph with constant maximum indegree. Thus the global encoding vector of an edge in G' can be computed in $O(d)$ time, where d is the outdegree of the source vertex s. Since Γ_v has $O(d_v)$ edges, the global encoding vectors of all the edges of Γ_v can be computed in $O(d \cdot d_v)$ time. Therefore, all the global encoding vectors can be computed in $\sum_{v\in V} O(d \cdot d_v) = \sum_{v\in V} O(d \cdot (d^{in}(v) + d^{out}(v))) = O(dm)$ time. This proves the first part of Theorem 1.2.

The encoding time is optimal since writing down all the global encoding vectors already takes $O(dm)$ time. It is surprising that the vectors of all the outgoing edges of Γ_v in G' can be computed in $O(d \cdot d_v)$ time, the same time complexity as computing the vector of just one outgoing edge of v in G. This can be used to improve the random linear coding algorithm [17] for network coding.

To compute the edge connectivity from s to Γ_t in G', by Theorem 2.1 we can compute the rank of the incoming vectors of Γ_t in G'. For a rectangular matrix of size $a \times b$, its rank can be computed in $O(ab^{\omega-1})$ time. So the total decoding time is $\sum_v O(d^{in}(v) \cdot d^{\omega-1}) = O(m \cdot d^{\omega-1})$. This proves the second part of Theorem 1.2.

Our algorithm is faster than running the Even-Tarjan algorithm for $\Omega(n)$ times for all n and m. When d is small we can compute single source edge connectivites in linear time. Let $\lambda_{s,t}$ be the edge connectivity from s to t. When the outdegree of s is unbounded, this can be used to obtain an algorithm to compute $\min\{\lambda_{s,t}, d\}$ for all t in linear time for constant d, by adding a new source s' with d outgoing edges to s and compute the edge connectivities from s'.

3.2. Directed Planar Graphs

One way to compute the global encoding vectors is to solve the system of linear equations $F(I - K) = H_s$. For some graphs this system of linear equations can be solved more efficiently. The result in this section can be applied to more general classes of graphs than the class of directed planar graphs.

We say an undirected graph $G = (V, E)$ has a $(f(n), \alpha)$-separation, if V can be partitioned into three parts, X, Y, Z such that $|X \cup Z| \leq \alpha|V|$, $|Y \cup Z| \leq \alpha|V|$, $|Z| \leq f(n)$, and no edges have endpoints in both X and Y. A class of graphs is hereditary if it is closed under taking subgraphs (e.g. planar graphs). For a hereditary class of graphs, if there is always a $(f(n), \alpha)$-separation for any n vertex graph in this class, then one can recursively separate the separated parts X and Y until the separated pieces are small enough. This yields a $(f(n), \alpha)$-weak separator tree (see [3] for a formal definition). It is known that planar graphs, bounded genus graphs and fixed minor free graphs have a $(\sqrt{n}, 2/3)$-separation [30], [14], [2], and thus a $(\sqrt{n}, 2/3)$ weak separator tree.

Given $Ax = b$ where A is an $n \times n$ matrix, the underlying graph of A is an undirected graph with n vertices where there is an edge ij if $A_{i,j} \neq 0$ or $A_{j,i} \neq 0$. The nested dissection method of Lipton, Rose and Tarjan [29] shows that if A is a symmetric positive definite matrix and the underlying graph has an $(O(n^\beta), 2/3)$-weak separator tree then $Ax = b$ can be solved in $O(n^{\omega\beta})$ time. In our setting, however, the matrix $(I - K)$ is not symmetric and its elements are from a finite field. Interestingly, Alon and Yuster [3] extended the nested dissection method very recently to solve system of linear equations $Ax = b$ over any finite field and for any matrix (not necessarily symmetric) whose underlying graph has an $(O(n^\beta), \alpha)$-weak separator tree, and this is exactly what we need. In the following a family of graphs is δ-sparse if any n-vertex graph in this family has at most δn edges.

Theorem 3.3 (Alon, Yuster [3]) *Let $\mathcal{F}$ be a δ-sparse family of graphs with an $O(n^\gamma)$ time algorithm to find an $(O(n^\beta), \alpha)$-weak separator tree where α is a constant smaller than one. Given a system of linear equations $Ax = b$ where $A \in \mathbb{F}^{n\times n}$ is non-singular, $b \in \mathbb{F}^n$ and the underlying graph of A is in $\mathcal{F}$, there is a randomized algorithm that finds the unique solution of the system in $O(n^{\omega\beta} + n^\gamma + n \log n)$ time.*

We are interested in computing the network coding solution for a directed planar graph $G = (V, E)$ with constant maximum degree. If the source vertex has outdegree d, then the global encoding vectors are of dimension d, and the equation $F(I - K) = H_s$ can be solved by d systems of linear equations of the form $(I - K)^T x = h$ where h is the transpose of a row of H_s. The underlying graph of $(I-K)$ is not a planar graph, but is the line graph of a planar graph. Nevertheless, if the maximum degree of G is a constant, then an $(O(\sqrt{n}), 2/3)$-separation in G corresponds to an $(O(\sqrt{n}), \alpha)$-separation in the line graph for α a constant smaller than one, by taking the edges with one endpoint in the separator in G as a separator of the line graph. Since a separator can be found in linear time in planar graphs [30], by applying Theorem 3.3 with $\gamma = 1$, $\beta = 1/2$ and δ a constant (since the maximum degree is a constant), we obtain an $O(n^{\omega/2})$ time randomized algorithm for solving one system of linear equation $(I - K)^T x = h$. The total encoding time is also $O(n^{\omega/2})$ as d is a constant. To compute the edge connectivity from s to a vertex t, by Theorem 2.1 we can just compute the rank of a $d \times l$ matrix where $l \leq d$, and this can be done in constant time. Therefore the total decoding time is $O(n)$. This proves Theorem 1.3.

The same result holds for bounded genus graphs with constant maximum degree, since a $(O(\sqrt{n}), 2/3)$ separator can be found in linear time [14]. For fixed minor free graphs with constant maximum degree, we can use Alon and Yuster [3] result to obtain a $O(n^{3\omega/(3+\omega)}) = O(n^{1.33})$ time algorithm. Very recently Kawarabayashi and Reed [24] gave an $O(n^{1+\epsilon})$ time algorithm for finding a separator in fixed minor-free graphs for any $\epsilon > 0$, and this can be used to improve the running time for bounded degree fixed minor free graphs to $O(n^{\omega/2})$.

4. All Pairs Edge Connectivities

In this section we show how to compute all pairs edge connectivities in general directed graphs, and then show how to improve the running time for graphs with good separators.

4.1. General Directed Graphs

To solve $F = FK + H_s$, one can compute the inverse of $(I-K)^{-1}$ and get $F = H_s(I-K)^{-1}$. It takes $O(m^\omega)$ time to compute $(I-K)^{-1}$ since the matrix $(I-K)$ is of size $m \times m$, but we observe that all pairs edge connectivities can be computed readily once we have $(I-K)^{-1}$. In our setup, H_s is a $d^{out}(s) \times m$ matrix with a $d^{out}(s) \times d^{out}(s)$ identity matrix in the columns corresponding to $\delta^{out}(s)$. So F is just equal to $((I-K)^{-1})_{\delta^{out}(s),*}$ up to permuting rows. Therefore $(I-K)^{-1}$ contains all the global encoding vectors for all source vertices, and thus the total encoding time for all pairs is $O(m^\omega)$.

To compute the edge connectivity from s to t, by Theorem 2.1 we compute the rank of $F_{*,\delta^{in}(t)}$ and this is just equal to the rank of $((I-K)^{-1})_{\delta^{out}(s),\delta^{in}(t)}$. As shown in Section 3.1, given a vertex s, computing the ranks of $((I-K)^{-1})_{\delta^{out}(s),\delta^{in}(t)}$ for all t can be done in $O(m \cdot (d^{out}(s))^{\omega-1})$ time. So the total decoding time is $\sum_{s \in V} O(m \cdot (d^{out}(s))^{\omega-1})$. Since $d^{out}(s) \leq n$ in a simple graph, $\sum_{s\in V} O((d^{out}(s))^{\omega-1})$ is at most $O(\frac{m}{n} \cdot n^{\omega-1})$. This implies that the total decoding time is at most $O(m^2 n^{\omega-2})$, which is $O(m^\omega)$ since $m \geq n$. This proves Theorem 1.4.

Our algorithm is faster than running Even-Tarjan algorithm for $\Omega(n^2)$ pairs as long as $m = O(n^{1.93})$. For $m = O(n)$ our algorithm runs in $O(n^\omega)$ time while running Even-Tarjan algorithm for $\Omega(n^2)$ pairs takes $O(n^{3.5})$ time. We note that Theorem 1.4 and Theorem 1.5 can also be derived from the matrix formulation of Ingleton and Piff [20], [9] for vertex connectivities. By transforming to line graphs, the resulting matrix is similar to $(I-K)^{-1}$.

4.2. Directed Graphs with Good Separators

We show a faster method to compute $(I-K)^{-1}$ when its underlying graph has a weak separator tree (see Section 3.2). Let

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

where A and D are square matrices. If A is nonsingular, $S = D - CA^{-1}B$ is called the Schur complement of A.

Lemma 4.1 (Schur's formula [37]) *Let M and A, B, C, D be matrices as defined above. Then* $\det(M) = \det(A) \times \det(S)$. *If A and S are nonsingular then*

$$M^{-1} = \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}$$

Our observation is that for graphs with good separators, we can find a partition of $(I-K)$ such that B and C are of low rank, so that we can compute $(I-K)^{-1}$ faster by a divide and conquer algorithm.

The rest of this section is organized as follows. In Section 4.2.1, we will show that if a matrix is "well-separable" (to be defined in the next section), then its inverse can be computed more efficiently. Then in Section 4.2.2, we show that the matrix $I - K$ for graphs with good separators is well-separable, and conclude that the edge connectivities in such graphs can be computed efficiently.

4.2.1. Inverse of Well-Separable Matrix: Let M be an $n \times n$ matrix as defined above. We say M is (r, α)-*well-separable* (α is a constant smaller than one) if M is invertible, both A and D are (r, α)-well-separable square matrix with dimension no more than αn, and both B and C are of rank no more than r. In this section we are going to show that the inverse of a well-separable matrix can be computed in $O(r^{\omega-2}n^2)$ time.

To compute M^{-1}, we first compute the inverses of A and D recursively. We will compute M^{-1} using Schur's formula. A key step is to use the Sherman-Morrison-Woodbury formula to compute S^{-1} efficiently, instead of first computing $D - CA^{-1}B$ and then $(D - CA^{-1}B)^{-1}$. The Sherman-Morrison-Woodbury formula tells us how to compute the $(D + UV^T)^{-1}$ efficiently if D^{-1} is known and U, V are low rank matrices.

Lemma 4.2 (Sherman-Morrion-Woodbury [36]) *Let M be an $n \times n$ matrix, U be an $n \times k$ matrix, and V be an $n \times k$ matrix. Suppose that M is nonsingular. Then*

- *$M + UV^T$ is nonsingular if and only if $I + V^T M^{-1} U$ is nonsingular.*
- *if $M + UV^T$ is nonsingular, then $(M + UV^T)^{-1} = M^{-1} - M^{-1}U(I + V^T M^{-1}U)^{-1}V^T M^{-1}$.*

Firstly, since $\text{rank}(B), \text{rank}(C) \leq r$, we can write $B = B_u B_v^T$ and $C = C_u C_v^T$ where B_u, B_v, C_u and C_v are size $O(n) \times r$ matrices. This can be done in $O(r^{\omega-2}n^2)$ time.

Claim 4.3 *If we have A^{-1} and D^{-1}, then we can compute $S^{-1} = (D - CA^{-1}B)^{-1}$ in $O(r^{\omega-2}n^2)$ time.*

Proof: By Schur's formula $S = D - CA^{-1}B$ is non-singular if M^{-1} and A^{-1} exist. The proof consists of two steps. We will first write $CA^{-1}B$ as a product of two low rank matrices. Then we use the Sherman-Morrison-Woodbury formula to compute S^{-1} efficiently. In the following, rectangular matrix multiplications will be done by dividing each matrix into submatrices of size $r \times r$. Multiplications between these submatrices can be done in $O(r^\omega)$ time.

First we consider $CA^{-1}B$, which is equal to $(CA^{-1}B_u)B_v^T$. $C(A^{-1}B_u)$ is of size $O(n) \times r$ and can be computed using $O(n^2/r^2)$ submatrices multiplications of size $r \times r$. By putting $U = CA^{-1}B_u$ and $V = B_v$, we have $CA^{-1}B = UV^T$ where U and V are of size $O(n) \times r$.

Now we can use Sherman-Morrison-Woodbury formula to compute $(D + UV^T)^{-1}$ efficiently since D^{-1} is known and U, V are low rank matrices. By the formula,

$$S^{-1} = D^{-1} - D^{-1}U(I + V^T D^{-1}U)^{-1}V^T D^{-1}.$$

Similar to the above $V^T D^{-1}U$ can be computed using $O(n^2/r^2)$ submatrix multiplications. Since S is non-singular, $(I + V^T D^{-1}U)^{-1}$ exists by Lemma 4.2, and it can be computed in one submatrix multiplication time as $I + V^T D^{-1}U$ is of size $r \times r$. Finally, since $(D^{-1}U)^T$ and $(V^T D^{-1})$ are $r \times O(n)$ matrices, we can compute $(D^{-1}U)(I + V^T D^{-1}U)^{-1}(V^T D^{-1})$ using $O(n^2/r^2)$ submatrix multiplications. Hence the overall time complexity is $O(r^\omega \cdot n^2/r^2) = O(r^{\omega-2}n^2)$. ■

Using Claim 4.3, we can now compute M^{-1} using Schur's formula efficiently.

Claim 4.4 *If we have A^{-1}, D^{-1} and S^{-1}, then we can compute M^{-1} in $O(r^{\omega-2}n^2)$ time.*

Proof: By Schur's formula, to compute M^{-1}, we need to compute $A^{-1}BS^{-1}CA^{-1}$, $A^{-1}BS^{-1}$ and $S^{-1}CA^{-1}$. These multiplications all involve some $O(n) \times r$ matrix B_u, B_v, C_u or C_v follows:

- $A^{-1}BS^{-1}CA^{-1} = (A^{-1}B_u)(B_v^T S^{-1}CA^{-1})$
- $A^{-1}BS^{-1} = (A^{-1}B_u)(B_v^T S^{-1})$
- $S^{-1}CA^{-1} = (S^{-1}C_u)(C_v^T A^{-1})$

All steps during computation of these products involve a $O(n) \times r$ matrix. Hence they all only take $O(n^2/r^2)$ submatrix multiplications to compute. Therefore the total time complexity is $O(r^\omega \cdot n^2/r^2) = O(r^{\omega-2}n^2)$. ■

By Claim 4.3 and Claim 4.4, we can compute M^{-1} in $O(r^{\omega-2}n^2)$ time if we are given A^{-1}, D^{-1} and also low rank decompositions of B and C. We can then use a recursive algorithm to compute M^{-1}.

Theorem 4.5 *If an $n \times n$ matrix M is (r, α)-well-separable, and the separation can be found in $O(n^\gamma)$ time, then M^{-1} can be computed in $O(n^\gamma + r^{\omega-2}n^2)$ time.*

Proof: First we analyse the time to compute M^{-1}. We use a $O(n^\gamma)$ time algorithm to find an (r, α)-well-separable partition of M. By the property of the partition, both B and C are matrices of rank at most r. Then we can write $B = B_u B_v^T$ and $C = C_u C_v^T$ in $O(r^{\omega-2}n^2)$ time where B_u, B_v, C_u and C_v are all $O(n) \times r$ matrices. We then compute A^{-1} and D^{-1} recursively, as A and D by definition are also (r, α)-separable. Using these inverses we can apply Claim 4.3 to compute S^{-1}, then apply Claim 4.4 to compute M^{-1} using A^{-1}, D^{-1} and S^{-1}. Thus, given A^{-1} and D^{-1}, we can compute M^{-1} in $O(r^{\omega-2}n^2)$ time. Let $f(n)$ be the time to compute M^{-1} of size $n \times n$. Then $f(n) = f(\alpha n) + f((1-\alpha)n) + O(n^\gamma) + O(r^{\omega-2}n^2)$, and it can be shown by induction that $f(n) = O(n^\gamma + r^{\omega-2}n^2)$. ■

4.2.2. Proof of Theorem 1.5: In this section we show that all pairs edge connectivities in planar graphs, bounded genus graphs, and fixed minor free graphs can be computed in $O(d^{\omega-2}n^{\omega/2+1})$ time.

We will first see that the underlining matrix $I-K$ for these graphs are well separable. Thus we can apply Theorem 4.5 together with the fact the these graphs have $O(n)$ edges, to obtain a $O(d^{\omega-2}n^{\omega/2+1})$ time algorithm to compute $(I - K)^{-1}$. Finally we show that the time to compute the required ranks of all submatrices in $(I-K)^{-1}$ is $O(d^{\omega-2}n^{\omega/2+1})$.

A fixed minor free graph G (and its subgraph) has $O(n)$ edges and a $(O(\sqrt{n}), 2/3)$-separation (recall the definition from Section 3.2), we claim that the matrix $I-K$ of G is $O(d\sqrt{n}, \alpha)$-well-separable for some constant $\alpha < 1$. To show that $I-K$ is well-separable, we can use a separator to divide the graph into three parts X, Y, Z such that $|X \cup Z| \leq 2n/3$, $|Y \cup Z| \leq 2n/3$ and $|Z| = O(\sqrt{n})$, and there are no edges between X and Y. Let E_1 be the set of edges with at least one endpoint in X and let E_2 be $E - E_1$. We partition $I - K$ as follows.

$$\begin{array}{c c} & \begin{array}{cc} E_1 & E_2 \end{array} \\ \begin{array}{c} E_1 \\ E_2 \end{array} & \begin{pmatrix} A & B \\ C & D \end{pmatrix} \end{array}$$

Since $|Y| = \Omega(n)$ and each vertex in Y is of degree at least one, we have $|E_1| = \Omega(n)$ and $|E_2| = \Omega(n)$. Also we have $|E_1| = O(n)$ and $|E_2| = O(n)$ since $m = O(n)$, and thus $|E_1|, |E_2| \leq \alpha|E|$ for some constant $\alpha < 1$. Let F_1 be the subset of E_1 with one endpoint in Z, and define F_2 similarly. Then any edge in $E_1 - F_1$ does not share an endpoint with any edge in $E_2 - F_2$. Since $|Z| = O(\sqrt{n})$ and each vertex is of degree at most d, there are at most $O(d\sqrt{n})$ edges with one endpoint in Z, and thus $|F_1| = O(d\sqrt{n})$ and $|F_2| = O(d\sqrt{n})$. Therefore, submatrices B and C have $O(d\sqrt{n})$ non-zero rows and columns, and thus are of rank at most $O(d\sqrt{n})$. Also $I-K$ must be invertible because $I-K$ has ones on its diagonal, and thus $\det(I-K)$ is a non-zero polynomial. Matrices A and D correspond to matrices $I-K$ for subgraphs of G, and by the inductive hypothesis A and D are $(O(d\sqrt{n}), \alpha)$-well-separable. Hence we can conclude that $I - K$ is $(O(d\sqrt{n}), \alpha)$-well-separable.

Now it remains to analyze the probability that all $I - K$, A and D remain invertible after substituting random values to $k_{i,j}$. Note that $\det(I - K)$ is a degree m non-zero polynomial, because $I - K$ has ones on its diagonal. Thus, by the Schwartz-Zippel Lemma, $I - K$ is invertible with probability at least $1 - m/|\mathbb{F}|$. Since A and D correspond to matrices $I - K$ for subgraphs of G, we can apply the same argument to A and D recursively, and all the required matrices are invertible with probability at least $1 - O(m \log m)/|\mathbb{F}|$ by the union bound.

As a result $I-K$ is $(O(d\sqrt{n}), \alpha)$-well-separable. We can now apply Theorem 4.5 with $\gamma = 1.5$ [2]. So $(I-K)^{-1}$ can be computed in $O(n^{1.5} + (d\sqrt{n})^{\omega-2}n^2) = O(d^{\omega-2}n^{\omega/2+1})$

time. For the decoding time, by a similar argument as in Section 4.1, the total decoding time is $\sum_{s \in V} O(m \cdot (d^{out}(s))^{\omega-1})$. Since $m = O(n)$ and $d^{out}(s) \leq d$, this is at most $O(n \cdot \frac{n}{d} \cdot d^{\omega-1}) = O(n^2 d^{\omega-2})$, which is dominated by the encoding time. Thus we obtain the following corollary. This is faster than Theorem 1.4 when $d = O(\sqrt{n})$

Corollary 4.6 *All pairs edge connectivities can be computed in $O(d^{\omega-2} n^{\omega/2+1})$ time for any directed fixed minor free graph with maximum degree d.*

5. Edge Splitting-Off

In this section we show that expanders and superconcentrators can be applied to design fast algorithms for another well-studied graph connectivity problem.

Splitting-off a pair of edges (ux, xv) means deleting these two edges and adding a new edge uv if $u \neq v$. Note that the above definition works for both undirected and directed graphs. The content of edge splitting off theorems is to prove the existence of one pair of edges (ux, xv) so that its splitting-off preserves the edge connectivities for all pairs of vertices. These results are a powerful tool for proving theorems and developing algorithms for many graph connectivity problems, including connectivity augmentation problems, network design problems, tree packing problems and graph orientation problems (see the references in [26]).

Faster algorithms for the splitting-off operation can be used to obtain faster algorithms for many graph connectivity problems. There are several existing algorithms for this task (e.g. [13], [5], [7]) in undirected and directed graphs, but most algorithms only preserve global edge connectivity (i.e. the value of the global min-cut). Our results in this section apply to the general setting where all pairs edge connectivities are preserved.

In undirected graphs, Mader proved that there is a "good" pair of edges in almost all situations.

Theorem 5.1 (Mader [31]) *Let $G = (V, E)$ be an undirected graph and $x \in V$. If there is no cut edge incident to x and $d(x) \neq 3$, then there exists an edge pair (yx, xz) so that its splitting-off preserves the edge connectivity for every pair of vertices $a, b \in V - x$.*

There is a similar theorem for Eulerian directed graphs where for every vertex its indegree is equal to its outdegree.

Theorem 5.2 (Bang-Jensen, Frank and Jackson [4]) *Let $G = (V, E)$ be an Eulerian directed graph and $x \in V$. Then there exists an edge pair (yx, xz) so that its splitting-off preserves the edge connectivity for every ordered pair of vertices $a, b \in V - x$.*

In undirected graphs, when $d(x)$ is even, Mader's theorem can be repeatedly applied until x is of degree zero. In directed graphs, Theorem 5.2 can also be repeatedly applied until x is of degree zero. We call this a complete splitting-off at x. Our goal is to design a fast algorithm to completely split-off x. A straightforward algorithm is to try every pair of edges on x, and check whether the edge connectivities decrease for some pairs after its splitting-off. In the worst case this requires $O((d(x))^2)$ attempts to completely split-off x. We show how to use expanders and superconcentrators to completely split-off x in $O(d(x))$ attempts in undirected and directed graphs respectively.

In undirected graphs, it is proved in [26] that $O(d(x))$ attempts are enough to completely split-off x, using a structural theorem of mincuts. We show how to get the same result immediately using expanders. We replace x by a constant degree expander graph H_x with $O(d(x))$ vertices, and "rewire" each edge of x to a distinct vertex in H_x.

The graph H_x is required to satisfy the following propery: for every subset $S \subseteq V(H_x)$ with $|S| \leq |V(H_x)|/2$, it holds that $d(S) \geq |S|$ where $d(S)$ is the number of edges with exactly one endpoint in S. By Menger's theorem, it follows that for every $S, T \subseteq V(H_x)$ with $|S| = |T|$, there are $|S|$ edge disjoint paths between S and T in H_x. Hence the edge connectivity between every pair of vertices $a, b \in V - x$ in the resulting graph is the same as in the original graph. We can add extra edges to make sure that every vertex in H_x is of even degree. Then every vertex in H_x can be completely split-off by Mader's theorem. Since every vertex in H_x is of constant degree, we can completely split-off one vertex in H_x in $O(1)$ attempts by the straightforward algorithm, and thus we can completely split-off all vertices in H_x in $O(d(x))$ attempts since H_x has only $O(d(x))$ vertices. After we completely split-off all vertices in H_x, this is the same as completely split-off x in the original graph and the edge connectivity between every pair is preserved.

For the algorithm to be efficient, we need to show that the expander graph H_x can be constructed efficiently, and here we give an example of one such construction. For a d-regular graph $G = (V, E)$, the edge expansion of a set $S \subseteq |V|/2$ is defined as $h(S) = d(S)/(d \cdot |S|)$. The requirement that $d(S) \geq |S|$ for every set S with $|S| \leq |V|/2$ is equivalent to the requirement that $h(S) \geq 1/d$ for every set S with $|S| \leq |V|/2$. Let $h(G) = \min_{S:|S| \leq |V|/2} h(S)$. By Cheeger's inequality, $h(G) \geq (1 - \lambda_2)/2$ where λ_2 is the second largest eigenvalue in the normalized adjacency matrix $A(G)/d$. So it suffices to construct a graph with $\lambda_2 \leq 1 - 2/d$. Consider the graph $G_p = (V_p, E_p)$ in which $V_p = \{0, \ldots, p-1\}$. For $a \in V_p - \{0\}$, the vertex a is connected to $a + 1 \mod p$, to $a - 1 \mod p$ and to its multiplicative inverse $a^{-1} \mod p$. The vertex 0 is connected to 1, to $p - 1$ and has a self-loop. This graph G_p is 3-regular, and it is known that $\lambda_2 < 0.9999$ (see [18], Section 11.1.2). By taking the 8-th power of G_p, denoted by G_p^8, one can verify that $\lambda_2^8 \leq 1 - 2/d^8$. So G_p^8 satisfies the property that $d(S) \geq |S|$ for every S with $|S| \leq |V|/2$, and is of constant degree. Since G_p can be constructed in

$O(p \log p)$ time, G_p^8 can also be constructed in $O(p \log p)$ time. Therefore, for a vertex x with degree $d(x)$, we can set H_x to be G_p^8 for $p = O(d(x))$.

In directed graphs, we can completely split-off a vertex in $O(d^{in}(x))$ attempts using superconcentrators. The argument is very similar to the undirected case and is omitted.

In general, expander graphs and superconcentrators can be used to reduce the maximum degree significantly while preserving the edge connectivities and only increasing the number of vertices moderately. This may be used to reduce the running time of an algorithm that has a super-linear dependency on the maximum degree. We believe that these reductions will find further applications for other graph connectivity problems.

Acknowledgement

We thank Andrej Bogdanov for his help on expanders and superconcentrators, and Nick Harvey for useful comments on the paper.

References

[1] R. Ahlswede, N. Cai, S.R. Li, and R.W. Yeung. *Network Information Flow*. IEEE Transactions on Information Theory, 46(4):1204–1216, 2000.

[2] N. Alon, P. Seymour, R. Thomas. *A Separator Theorem for Nonplanar Graphs*. Journal of the AMS, 3(4):801–808, 1990.

[3] N. Alon, R. Yuster. *Solving Linear Systems Through Nested Dissection*. FOCS 2010, 225–234.

[4] J. Bang-Jensen, A. Frank, and B. Jackson. *Preserving and increasing local edge-connectivity in mixed graphs*. SIAM Journal on Discrete Mathematics, 8(2):155-178, 1995.

[5] A. A. Benczúr and D. R. Karger. *Augmenting undirected edge connectivity in $O(n^2)$ time*. J. Algorithms, 37(1):2-36, 2000.

[6] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. *An $\tilde{O}(mn)$ Gomory-Hu tree construction algorithm for unweighted graphs*. STOC 2007, 605–614.

[7] A. Bhalgat, R. Hariharan, T. Kavitha, and D. Panigrahi. *Fast edge splitting and Edmonds' arborescence construction for unweighted graphs*. SODA 2008, 455–464.

[8] U. Brandes, D. Wagner. *A linear time algorithm for the arc disjoint Menger problem in planar directed graphs*. ESA 1997, 64–77.

[9] J. Cheriyan. *Randomized $\tilde{O}(M(|V|))$ Algorithms for Problems in Matching Theory*. SIAM J. Computing, 26(6):1635–1655, 1997.

[10] E. Erez, M. Feder. *Convolutional network codes*. ISIT, 146, 2004.

[11] S. Even, R.E. Tarjan. *Network Flow and Testing Graph Connectivity*. SIAM J. Computing, 4(4), 507-518, 1975.

[12] O. Gabber, Z. Galil. *Explicit Constructions of Linear-Sized Superconcentrators*. Journal of Computer and System Sciences, 22:407–420, 1981.

[13] H. N. Gabow. Efficient splitting off algorithms for graphs. STOC 1994, 696–705.

[14] J.R. Gilbert, J.P. Hutchinson, and R.E. Tarjan. *A Separator Theorem for Graphs of Bounded Genus*. J. Algorithms, 5:391–407, 1984.

[15] A.V. Goldberg, S. Rao. *Flows in undirected unit capacity networks*. FOCS 1997, 32-34.

[16] N. Harvey. *Algebraic Algorithms for Matching and Matroid Problems*. SIAM J. Computing, 39:679-702, 2009.

[17] T. Ho, M. Médard, R. Koetter, D.R. Karger, M. Effros, J. Shi, B. Leong. *A Random Linear Network Coding Approach to Multicast*. IEEE Transactions on Information Theory 52, 4413–4430, 2006.

[18] S. Hoory, N. Linial, A. Wigderson. *Expander Graphs and their Applications*. Bulletin of the AMS, 43:439-561, 2006.

[19] J. Hopcroft, R.M. Karp. *An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs*. SIAM J. Computing, 2:225-231, 1973.

[20] A.W. Ingleton, M.J. Piff. *Gammoids and transversal matroids*. J. Combin. Theory Ser. B, 15, 51–68, 1973.

[21] S. Jaggi. *Design and analysis of network codes*. PhD thesis, California Institute of Technology, 2006.

[22] S. Jaggi, P. Sanders, P.A. Chou, M. Effros, S. Egner, K. Jain, L.M.G.M. Tolhuizen. *Polynomial time algorithms for multicast network code construction*. IEEE Transactions on Information Theory 51, 1973–1982, 2005.

[23] D.R. Karger, M.S. Levine. *Finding maximum flows in undirected graphs seems easier than bipartite matching*. STOC 1998, 69–78.

[24] K. Kawarabayashi, B. Reed. *A separator theorem in minor-closed classes*. FOCS 2010.

[25] R. Koetter, M. Médard. *An algebraic approach to network coding*. IEEE/ACM Transactions on Network, 11(5):782–795, 2003.

[26] L.C. Lau, C.K. Yung. *Efficient edge splitting-off algorithms maintaining all-pairs edge-connectivities*. IPCO 2010, 96–109.

[27] S.-Y.R. Li, R.W. Yeung. *On Convolutional Network Coding*. ISIT 2006, 1743–1747.

[28] S.-Y.R. Li, R.W. Yeung, and N. Cai. *Linear Network Coding*. IEEE Transactions on Information Theory, 49(2):371–381, 2003.

[29] R.J. Lipton, D.J. Rose, and R.E. Tarjan. *Generalized Nested Dissection*. SIAM J. Numerical Analysis, 16(2):346–358, 1979.

[30] R.J. Lipton, R.E. Tarjan. *A separator theorem for planar graphs*. SIAM J. Applied Mathematics, 36:177–189, 1979.

[31] W. Mader. *A reduction method for edge-connectivity in graphs*. Annals of Discrete Mathematics, 3:145–164, 1978.

[32] M. Mucha, P. Sankowski. *Maximum matchings via Gaussian elimination*. FOCS 2004, 248-255.

[33] A. Schrijver. Combinatorial Optimization. Springer, 2003.

[34] A.L. Toledo, X. Wang. *Efficient multipath in sensor networks using diffusion and network coding*. 40th Annual Conference on Information Sciences and Systems, 2006.

[35] L. Valiant. *On Non-linear Lower Bounds in Computational Complexity*. STOC 1975.

[36] M.A. Woodbury. *Inverting modified matrices*. Memorandum Report 42, Statistical Research Group, Princeton University, 1950.

[37] F. Zhang. *The Schur Complement and Its Applications*. Springer, 2005.