Algorithmica (2007) 47: 379–397
DOI: 10.1007/s00453-006-0160-2

Algorithmica

Drawing Power Law Graphs Using a Local/Global Decomposition[1]

Reid Andersen,[2] Fan Chung,[2] and Linyuan Lu[3]

Abstract. It has been noted that many realistic graphs have a power law degree distribution and exhibit the small-world phenomenon. We present drawing methods influenced by recent developments in the modeling of such graphs. Our main approach is to partition the edge set of a graph into "local" edges and "global" edges and to use a standard drawing method that allows us to give added importance to local edges. We show that our drawing method works well for graphs that contain underlying geometric graphs augmented with random edges, and we demonstrate the method on a few examples. We define edges to be local or global depending on the size of the maximum short flow between the edge's endpoints. Here, a short flow, or alternatively an ℓ-short flow, is one composed of paths whose length is at most some constant ℓ. We present fast approximation algorithms for the maximum short flow problem and for testing whether a short flow of a certain size exists between given vertices. Using these algorithms, we give an algorithm for computing approximate local subgraphs of a given graph. The drawing algorithm we present can be applied to general graphs, but it is particularly well suited for small-world networks with power law degree distribution.

Key Words. Power law graphs, Small-world phenomenon, Network flow, Graph drawing, Hybrid graphs.

1. Introduction.

Although graph theory has a history of more than 250 years, it was only recently observed that realistic graphs from many different areas satisfy the so-called "power law," where the fraction of nodes with degree k is proportional to $k^{-\beta}$ for some positive exponent β. Graphs with power law degree distribution are prevalent in the Internet, communication networks, social networks, and biological networks [1]–[13]. Many real examples of networks also exhibit a so-called "small-world phenomenon" consisting of two distinct properties—small average distance between nodes and a clustering effect where two nodes sharing a common neighbor are more likely to be adjacent. It was shown in [14] that a random power law graph has small average distance and small diameter. However, random power law graphs do not adequately capture the clustering effect.

In [15] the authors introduced a hybrid graph model where a random power law graph called the *global graph* is added to an underlying *local graph*. A local graph is a graph where the endpoints of each edge are highly locally connected. In particular, a graph is said to be (f, ℓ)-local if for every edge in the graph we can route a fractional flow of size f between the endpoints through paths of length at most ℓ. Examples of graphs that

[1] A conference version appeared in *Proceedings of the Twelfth Annual Symposium on Graph Drawing*, 2004. Fan Chung's research was supported in part by NSF Grants DMS 0100472 and ITR 0205061.

[2] Department of Mathematics, University of California, San Diego, La Jolla, CA 92093, USA. {randerse,fan}@ucsd.edu.
[3] Department of Mathematics, University of South Carolina, Columbia, SC 29208, USA. lu@maths.sc.edu.

Received December 6, 2004; revised November 5, 2006. Communicated by J. Pach and F. Shahrokhi.
Online publication February 13, 2007.

are (f, ℓ)-local for various parameters include d-dimensional grid graphs, hypercubes, and certain subgraphs of random geometric graphs. Given an arbitrary graph G and a choice of parameters f and ℓ, we define the local graph of G to be its largest (f, ℓ)-local subgraph, providing a way to partition G into local and global edges. The main result of [15] is that the local graph of a hybrid graph is equal to the original local graph up to a small error, indicating that local graphs are robust to the addition of random edges.

In this paper we demonstrate that partitioning a graph into local and global edges based on local connectivity can be done quickly and can be useful in graph drawing. To obtain local/global partitions for large graphs, we present an approximation algorithm for computing the local graph. We also give an algorithm for the problem of computing the maximum short flow between given vertices. The number of iterations required in our maximum short flow algorithm depends on the value f_{opt} of the maximum short flow. We also give an algorithm for the problem of local connectivity testing, where we wish to determine whether there exists a short flow of a certain size f_{test} between given vertices. The number of iterations required by our testing algorithm is determined by f_{test} and not f_{opt}, so testing local connectivity can often be done more quickly than computing the maximum short flow. Our algorithms are based on the maximum multicommodity flow algorithm of Garg and Könemann [16], which is based on the technique of multiplicative updates.

After obtaining a local/global partition, we use it by modifying a standard graph layout algorithm to emphasize local edges. This technique can be applied with many graph layout algorithms, including force-directed algorithms and spectral drawing methods. We demonstrate that this method produces improved drawings for graphs that contain underlying geometric graphs augmented with random edges. This is theoretically supported, since the robustness of the local graph implies that most of the edges in the underlying geometric graph will be classified as local, while most of the edges from the random graph will be classified as global.

We also present a drawing method based on a more sophisticated partition that reflects other aspects of the structure of power law graphs. For example, it was shown in [14] that a random power law graph has roughly an "octopus" shape, with a dense "core" and a myriad of attached "tentacles." While the core itself may be a dense graph that is not amenable to most drawing methods, our algorithm takes advantage of the local subgraph and the sparse tree-like structures in the tentacles, yielding improved drawings. This partition extends the class of graphs for which the local/global partition is useful to many realistic networks where some notion of distance between vertices is reflected in the edge set.

2. Preliminaries.

In [15] the authors introduced *local graphs*, which are graphs where the endpoints of each edge are highly locally connected by short flows. In this section we introduce short flows and define local graphs. We will also state a theorem showing that local graphs are robust to the addition of random edges.

Throughout the paper the graphs we consider are undirected and unweighted. Given a graph G, we let $d_G(u, v)$ denote the graph distance between vertices u and v in G. We will use the following notation for the neighborhoods of a vertex:

$$N_k(u) = \{v \in G \mid d_G(u, v) \leq k\}.$$

2.1. *Short Flows*. There are a number of ways to define local connectivity between two given vertices u and v. A natural approach is to consider the connectivity through paths whose length is at most some fixed constant ℓ, which we call *ℓ-short* paths or simply *short* paths when ℓ is understood. The path connectivity $a_\ell(u, v)$ is the maximum size of a collection of ℓ-short edge-disjoint u–v paths. The cut connectivity $c_\ell(u, v)$ is the minimum size of an *ℓ-short u–v cut*, a set of edges whose removal leaves no ℓ-short u–v paths. The analogous version of Menger's theorem does not hold when we restrict to short paths—namely, $a_\ell(u, v)$ and $c_\ell(u, v)$ are not necessarily equal. However, it is easy to see that the trivial relations $a_\ell(u, v) \leq c_\ell(u, v) \leq \ell \cdot a_\ell(u, v)$ still hold.

Both $a_\ell(u, v)$ and $c_\ell(u, v)$ are difficult to compute. In particular, computing the maximum number of short disjoint paths is $\mathcal{NP}$-hard if $\ell \geq 4$ [17]. In light of this result we will define local connectivity based on a fractional relaxation of the maximum short disjoint path problem, which we call the maximum ℓ-short flow.

An *ℓ-short flow* is a linear combination of ℓ-short paths where each edge has congestion at most 1, and we denote the maximum ℓ-short flow between u and v by $f_\ell(u, v)$. Finding $f_\ell(u, v)$ can be viewed as the linear programming relaxation of finding $a_\ell(u, v)$. If we let A be the incidence matrix where each column represents a short path from u to v and each row represents an edge in the graph, then

$$f_\ell(u, v) = \max_{\mathbf{x}}\{\vec{\mathbf{1}}^T\mathbf{x} \mid A\mathbf{x} \leq \vec{\mathbf{1}}, \mathbf{x} \geq \vec{\mathbf{0}}\}. \tag{1}$$

The linear programming dual of the maximum short flow problem is a fractional cut problem. A short fractional cut is a weight function $w\colon E \to R^+$ such that $\sum_{e\in P} w(e) \geq 1$ for every short u–v path P. The dual of the short maximum flow problem is the problem of finding a short fractional cut that minimizes $\sum_{e\in G} w(e)$. We let $w_\ell(u, v)$ denote the size of the minimum short fractional cut, and we note that LP duality implies

$$a_\ell(u, v) \leq f_\ell(u, v) = w_\ell(u, v) \leq c_\ell(u, v). \tag{2}$$

Since all the coefficients in the incidence matrix, cost vector, and constraint vector in the problem formulation (1) are nonnegative, the maximum short flow problem belongs to a special subset of linear programs called fractional packing problems, for which there exist general techniques that often lead to polynomial-time algorithms [18]. In Section 3 we present an algorithm for computing the maximum ℓ-short flow based on the multiplicative update technique.

2.2. *Local Graphs*. We are now ready to define local connectivity and then local graphs, based on the notion of short flow.

Definition 1. Two vertices u and v are (f, ℓ)-connected if $f_\ell(u, v) \geq f$.

Definition 2. A graph L is (f, ℓ)-local if for each edge (u, v) in L, the endpoints u and v are (f, ℓ)-connected in L.

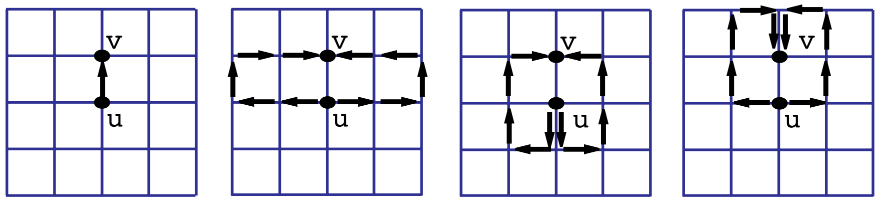

Fig. 1. A 5-short flow of size 4 between endpoints of a grid edge.

For example, the toroidal grid graph $C_n \times C_n$ is $(3, 3)$-local. This graph is also $(4, 5)$-local, which is slightly less obvious and highlights the difference between $a_\ell(u, v)$ and $f_\ell(u, v)$. A 5-short flow of size 4 between the endpoints of a grid edge is depicted in Figure 1. The flow consists of 1 unit routed on the path of length 1 in the leftmost grid and $\frac{1}{2}$ unit routed on each of the six paths shown in the remaining three grids.

The union of two (f, ℓ)-local graphs is (f, ℓ)-local, and so there is a unique largest (f, ℓ)-local subgraph of G, which we denote $L_{f,\ell}(G)$. In Section 4 we present algorithms for approximately computing $L_{f,\ell}(G)$. It is important to note that $L_{f,\ell}(G)$ is not necessarily connected.

2.3. *Mengerian Theorems for Short Paths.* It was mentioned previously that $a_\ell(u, v)$ and $c_\ell(u, v)$ are not necessarily equal. It is an open problem to determine how much these quantities can differ as a function of ℓ. It was shown by Lovász et al. [19] that $c_\ell(u, v) \leq \ell/2 \cdot a_\ell(u, v)$. Boyles and Exoo [20] have given a family of graphs where $c_\ell(u, v) \geq \ell/4 \cdot a_\ell(u, v)$. Their results were stated for the case of vertex-disjoint paths, but the results convert easily to the edge-disjoint case that we are considering. We now present a simple construction showing that $c_\ell(u, v) \geq \ell/3 \cdot a_\ell(u, v)$ for infinitely many ℓ, improving the best-known lower bound. An equivalent construction appeared in the thesis of Baier [21]. We conjecture that $c_\ell(u, v) \leq (\ell/3 + o(1)) \cdot a_\ell(u, v)$ for all graphs.

THEOREM 1. *For every ℓ there exists a graph such that $a_\ell(u, v) = 1$ and $c_\ell(u, v) \geq \ell/3$.*

PROOF. Consider a path $P = x_1 \cdots x_{2n}$ of length $2n - 1$ and let $u = x_1$ and $v = x_n$. We add $2n$ disjoint paths of length 2 between each pair of adjacent vertices x_i and x_{i+1}. Call this graph G_{2n}, and let $u = x_1$ and $v = x_n$.

We refer to the edges of P as base edges. It is easy to see that a u–v path using at most k base edges has length at least $(4n-2)-k$. If we set $\ell = 3n-2$, then every ℓ-short path must use at least n base edges so that its length is at most $(4n - 2) - n = 3n - 2 = \ell$. Since there are only $2n - 1$ base edges, any two ℓ-short paths must intersect in a base edge, so $a_\ell(u, v) = 1$. We now consider the size of a minimum ℓ-short cut C. Without loss of generality we can assume that C cuts only base edges. If C cuts $n - 1$ or fewer base edges, then n base edges remain and the path that proceeds from x_i to x_{i+1} by taking a base edge whenever possible has length at most $(4n - 2) - (n) = 3n - 2 = \ell$. Thus, $c_\ell(u, v) \geq n$, and so

$$c_\ell(u, v) = \frac{n}{3n-2} \cdot \ell \geq \frac{\ell}{3}. \qquad \square$$

It is not hard to modify the above construction to obtain examples for $\ell = 3n - 1$ and $\ell = 3n$ with $c_\ell(u, v) \geq \ell/3$. We can increase ℓ by 1 or 2 without changing $a_\ell(u, v)$ or $c_\ell(u, v)$ if we add a vertex x_0, let $u = x_0$, and connect x_0 to x_1 with $2n$ paths of length 1, or $2n$ paths of length 2.

2.4. *Random Graphs with Specified Expected Degree Sequence*. A random graph $G(\mathbf{w})$ with specified expected degree sequence $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ is formed by including each edge $v_i v_j$ independently with probability $p_{ij} = w_i w_j \rho$, where $\rho = (\sum w_i)^{-1}$. This includes self-loops of the form $v_i v_i$, which occur with probability $w_i^2 \rho$ as would be expected. We use the convention that a self-loop contributes only one to the degree of v_i. It is then easy to check that vertex v_i has expected degree w_i.

We assume that $\max_i w_i^2 < \sum_k w_k$ so that $p_{ij} \leq 1$ for all i and j. This condition also implies that the sequence $\mathbf{w}$ would be graphical if each w_i were an integer [22]. The standard random graph $G(n, p)$ on n vertices with edge probability p is a special case of the $G(\mathbf{w})$ model where $\mathbf{w} = (pn, pn, \ldots, pn)$. For a subset S of vertices, we define

$$\mathrm{Vol}(S) = \sum_{v_i \in S} w_i \quad \text{and} \quad \mathrm{Vol}_k(S) = \sum_{v_i \in S} w_i^k.$$

We let d denote the average expected degree $\mathrm{Vol}(G)/n$, and let $\tilde{d}$ denote the second-order average expected degree $\mathrm{Vol}_2(G)/\mathrm{Vol}(G)$. We also let m denote the maximum weight among the w_i.

2.5. *Robustness of Local Graphs*. We now consider the robustness of a local graph L to the addition of random edges. In particular we consider a graph formed by taking the union of the edge sets of two graphs: a local graph L that is (f, ℓ)-local for some choice of parameters and a random graph R with specified expected degree sequence $G(\mathbf{w})$. We call such a graph a *hybrid graph* H and use the notation $H = L \cup R$.

The following theorem, proved by the authors in [15], describes when the largest local graph in the hybrid graph, $L_{f,\ell}(H)$, is a good approximation of the original local graph. The result depends mainly on the local graph parameters f and ℓ and the parameter $\tilde{d}$ in the random graph $G(\mathbf{w})$.

THEOREM 2 (Recovery Theorem). *Let $H = L \cup R$ be a hybrid graph where L is an (f, ℓ)-local graph with bounded maximum degree and where $R = G(\mathbf{w})$ is a random graph with average expected degree d, second-order average expected degree $\tilde{d}$, and maximum weight m. Let $L' = L_{f,\ell}(H)$. Let $\alpha > 0$ be some constant such that $\hat{d} = n^\alpha$ is an upper bound for $\tilde{d}$. If $\hat{d}$ satisfies*

$$\hat{d} \leq \left(\frac{nd}{m^2}\right)^{1/\ell} n^{-3/f\ell},$$

then with probability $1 - O(n^{-1})$:

1. *The expected number of edges in $L' \backslash L$ is $O(\tilde{d})$.*
2. *$d_{L'}(x, y) \geq (1/\ell) d_L(x, y)$ for every pair of vertices $x, y \in L$.*

The proof of this theorem is contained in [15]. In Section 3 we consider the problem of algorithmically recovering the local graph by computing $L_{f,\ell}(H)$.

3. Algorithms for Short Flow Problems.

Garg and Könemann [16] gave simple combinatorial algorithms for maximum multicommodity flow, maximum concurrent flow, and general fractional packing problems based on the technique of multiplicative updates.

We present a simple multiplicative update algorithm for the maximum short flow problem based on their techniques. The number of iterations required by our algorithm is smaller than the number required by the algorithms for maximum multicommodity flow or general fractional packing problems. This is because the number of iterations in our algorithm can be bounded in terms of $f_\ell(u, v)$ instead of by the number of edges in the graph.

Since we are computing ℓ-short flows between vertices u and v, our algorithms need to consider only those vertices and edges that are involved in some ℓ-short u–v path, and so it suffices to examine only the vertices in the induced subgraph on $N_{\lceil \ell/2 \rceil}(u) \cup N_{\lfloor \ell/2 \rfloor}(v)$. Whenever possible, we will state running times in terms of the number of edges in this graph instead of the number of edges in the entire graph.

For the applications in this paper we will be repeatedly testing whether two given vertices are (f, ℓ)-connected for some constants f and ℓ. We refer to this as the local connectivity testing problem. We combine our maximum short flow algorithm with a simple greedy algorithm to obtain an algorithm for local connectivity testing where the number of iterations required depends on the constant f rather than $f_\ell(u, v)$. Thus, in many cases we can test local connectivity more quickly than we can compute the maximum short flow.

3.1. *Computing the Maximum Short Flow.*

`Maximum Short Flow`. We are given as input a graph with two distinguised vertices u and v, a length ℓ, and parameter ε. The algorithm maintains a weight function w assigning nonnegative real weights to the edges of the input graph and a flow function f assigning integer amounts of flow to ℓ-short u–v paths. The weight function and flow function are updated at each iteration of the algorithm, and we let w_i and f_i refer to the weight and flow functions after i iterations. Initially w_0 assigns weight δ to every edge, where δ is a constant depending on ε that we will determine later. Initially f_0 is the empty flow assigning zero units of flow to every short path. The following steps constitute the ith iteration of the algorithm:

1. Compute the minimum-weight ℓ-short u–v path with respect to the current weight function w_{i-1}. Call this path p_i.
2. Obtain f_i from f_{i-1} by routing one additional unit of flow on p_i.
3. Obtain w_i from w_{i-1} by multiplying $w_{i-1}(e)$ by $(1 + \varepsilon)$ for each edge e in p_i.

The algorithm performs iterations until the first time step t is reached where $\alpha(t) \geq 1$. At that time, let κ be the maximum congestion on any edge in f_t. Multiply f_t by a factor of $1/\kappa$ to obtain a feasible ℓ-short flow $\hat{F}_\ell(u, v)$ of size $\hat{f} = t/\kappa$.

Finding a minimum-weight ℓ-short u–v path given a weight function w can be done easily by dynamic programming, as we will sketch here. Let $W(x, k)$ be the minimum weight of a path of length less than of equal to k from vertex u to vertex x, and let $P(x, k)$ be the predecessor of x on some path achieving this minimum. Initially set $W(x, 0) = \infty$ if $x \neq u$, and set $W(u, 0) = 0$. The values $W(x, k+1)$ and $P(x, k+1)$ can be computed from the values of $W(y, k)$ for all vertices y by the rule

$$W(x, k+1) = \min\left\{ W(x, k)\,,\ \min_{y \in N(x)} W(y, k) + w(x, y) \right\}.$$

When $W(x, k)$ and $P(x, k)$ are known for all $k \in [0, \ell]$, a minimum-weight path $x_1, \ldots, x_\ell$ can be obtained by setting $x_\ell = v$, then letting $x_{i-1} = P(x_i, i)$ until $x_1 = u$ is reached. This can be carried out in time $T_{\mathrm{mwp}} = \mathcal{O}(m\ell)$.

THEOREM 3. *The algorithm* `Maximum Short Flow` *produces a feasible flow* $\hat{F}_\ell(u, v)$ *of value* $\hat{f}_\ell(u, v)$ *with the approximation guarantee*

$$\frac{f_\ell(u, v)}{\hat{f}_\ell(u, v)} \leq (1 - \varepsilon)^{-2}.$$

The algorithm runs in time $(f_\ell(u, v)(1/\varepsilon^2)\ell \log \ell)T_{\mathrm{mwp}}$, *where* $T_{\mathrm{mwp}} = \mathcal{O}(m\ell)$ *is the time required to find a minimum-weight* ℓ*-short path*, *and* m *is the number of edges in the induced subgraph on* $N_{\lceil \ell/2 \rceil}(u) \cup N_{\lfloor \ell/2 \rfloor}(v)$.

The analysis of the approximation guarantee for the maximum short flow algorithm is nearly identical to the analysis for the maximum multicommodity flow algorithm in Garg and Könemann [16], but the analysis of the number of iterations required to obtain the guarantee contains a significant difference. For completeness, we include the full analysis here.

PROOF OF THE APPROXIMATION GUARANTEE. Let $D(w) = \sum_e w(e)$ be the total amount of weight assigned by the weight function w and let $\alpha(w) = \min_p \sum_{e \in p} w(e)$ be the minimum weight on any ℓ-short path from the weight function w. Let $D(i) = D(w_i)$ and $\alpha(i) = \alpha(w_i)$. After each iteration $i \geq 1$,

$$\begin{aligned} D(i) &= \sum_e w_i(e) \\ &= \sum_e w_{i-1}(e) + \varepsilon \sum_{e \in p(i)} w(e) \\ &= D(i-1) + \varepsilon \cdot \alpha(i-1), \end{aligned}$$

and thus

$$D(i) = D(0) + \varepsilon \sum_{k=1}^{i} \alpha(k-1). \tag{3}$$

Notice that for any weight function w where $\alpha(w) \neq 0$, the scaled weight function $w \cdot 1/\alpha(w)$ is an ℓ-short fractional cut, since every ℓ-short path will have at least one unit of weight. Therefore,

$$f_\ell(u, v) = w_\ell(u, v) \leq \frac{D(w)}{\alpha(w)}. \tag{4}$$

Now consider the weight function $w_i - w_0$ where the initial weights δ are subtracted from each edge. We have $D(w_i - w_0) = D(i) - D(0)$ and $\alpha(w_i - w_0) \geq \alpha(i) - \delta\ell$. Applying (4) yields the inequality

$$f_\ell(u, v) \leq \frac{D(w_i - w_0)}{\alpha(w_i - w_0)} \leq \frac{D_i - D_0}{\alpha(i) - \delta\ell},$$

and so $D(i) - D(0) \geq f_\ell(u, v)(\alpha(i) - \delta\ell)$. We substitute this in (3) to obtain

$$\alpha(i) \leq \delta\ell + \frac{\varepsilon}{f_\ell(u, v)} \sum_{k=1}^{i} \alpha(k-1).$$

Rewriting the right side of this expression in terms of $\alpha(i-1)$ yields the recursive formula

$$\alpha(i) \leq \alpha(i-1)\left(1 + \frac{\varepsilon}{f_\ell(u, v)}\right).$$

Since $\alpha(0) \leq \delta\ell$ this implies

$$\alpha(i) \leq \alpha(0)\left(1 + \frac{\varepsilon}{f_\ell(u, v)}\right)^i \leq \delta\ell\left(1 + \frac{\varepsilon}{f_\ell(u, v)}\right)^i \leq \delta\ell e^{\varepsilon i / f_\ell(u,v)}.$$

On the other hand, we have the lower bound $\alpha(t) \geq 1$ by the stopping condition, which yields

$$1 \leq \alpha(t) \leq \delta\ell e^{\varepsilon t / f_\ell(u,v)},$$

and by taking the logarithm of both sides we obtain

$$(5) \qquad \frac{f_\ell(u, v)}{t} \leq \frac{\varepsilon}{\ln(1/\delta\ell)}.$$

Every edge e has $w_t(e) < 1 + \varepsilon$, since the last time e was increased it was on a path of length strictly less than 1, and its weight was increased by a factor of $1+\varepsilon$. Since $w_0(e) = \delta$, it follows that the weight of e can be increased at most $\log_{1+\varepsilon}((1+\varepsilon)/\delta)$ times, and thus the total flow through e is at most $\log_{1+\varepsilon}((1+\varepsilon)/\delta)$. Scaling the flow f_t by the maximum congestion yields a feasible flow $\hat{f}_\ell(u, v)$ of value at least $t/\log_{1+\varepsilon}((1+\varepsilon)/\delta)$. Using this value to bound the approximation ratio, we obtain

$$\frac{f_\ell(u, v)}{\hat{f}_\ell(u, v)} \leq \frac{f_\ell(u, v)}{t} \log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right).$$

Substitute the bound on $f_\ell(u, v)/t$ from (5) to obtain

$$\frac{f_\ell(u, v)}{\hat{f}_\ell(u, v)} \leq \frac{\varepsilon}{\ln(1/\delta\ell)} \log_{1+\varepsilon}\left(\frac{1+\varepsilon}{\delta}\right) = \frac{\varepsilon}{\ln(1+\varepsilon)} \frac{\ln((1+\varepsilon)/\delta)}{\ln(1/\delta\ell)}.$$

The ratio $\ln((1+\varepsilon)/\delta)/\ln(1/\delta\ell))$ is $(1-\varepsilon)^{-1}$ if we set $\delta = (1+\varepsilon)((1+\varepsilon)\ell)^{-1/\varepsilon}$. With this δ we have

$$\frac{f_\ell(u, v)}{\hat{f}_\ell(u, v)} \leq \frac{\varepsilon}{(1-\varepsilon)\ln(1+\varepsilon)} \leq \frac{\varepsilon}{(1-\varepsilon)(\varepsilon - \varepsilon^2/2)} \leq (1-\varepsilon)^{-2}.$$

Analysis of the running time. Each edge can have its weight multiplied by $(1+\varepsilon)$ at most $\log_{1+\varepsilon}((1+\varepsilon)/\delta)$ times until its weight reaches 1 and it cannot be further increased. If $C(u, v)$ is an ℓ-short cut, then at every time step some edge in $C(u, v)$ has its weight multiplied by $(1+\varepsilon)$. Thus, the algorithm performs at most $c_\ell(u, v)\log_{1+\varepsilon}((1+\varepsilon)/\delta)$ iterations before terminating. To obtain the stated approximation ratio, the initial weight δ is set to $(1+\varepsilon)((1+\varepsilon)\ell)^{-1/\varepsilon}$. Thus the number of iterations required is at most

$$c_\ell(u, v)\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta} \le c_\ell(u, v)\frac{1}{\varepsilon}\log_{1+\varepsilon}(1+\varepsilon)\ell \le c_\ell(u, v)\frac{2}{\varepsilon^2}\log_2 \ell. \tag{6}$$

The number of iterations can also be bounded in terms of the flow by applying the bound $c_\ell(u, v) \le \ell/2 \cdot f_\ell(u, v)$, which was mentioned in Section 2.1. The number of iterations required is then $(f_\ell(u, v)(1/\varepsilon^2)\ell\log_2 \ell)$. The result follows.

It should be noted that this bound on the number of iterations depends crucially on the assumption that all edge capacities are equal to 1. It is not hard to modify the algorithm to work in the case of arbitrary edge capacities (see [16]), but the running time may then depend on m instead of $f_\ell(u, v)$.

3.2. *Testing Local Connectivity*

`Approximate Local Connectivity Test`. We are given as input a graph with distinguised vertices u and v. We wish to accept if u and v are (f, ℓ)-connected and reject otherwise. The algorithm greedily chooses a collection A of short disjoint paths until either A has size f or A is maximal. If A reaches size f, then the algorithm accepts. Otherwise, the algorithm computes $\hat{f}_\ell(u, v)$ using `Maximum Short Flow`, and it accepts if and only if $\hat{f}_\ell(u, v) \ge (1-\varepsilon)^2 f$.

THEOREM 4. *The algorithm* `Approximate Local Connectivity Test` *accepts all vertex pairs that are (f, ℓ)-connected, and it rejects all vertex pairs that are not $((1-\varepsilon)^2 f, \ell)$-connected. The algorithm runs in time $(f_\ell(u, v)(1/\varepsilon^2)\ell\log\ell)T_{\mathrm{mwp}}$, where $T_{\mathrm{mwp}} = \mathcal{O}(m\ell)$ is the time required to find a minimum-weight ℓ-short path and m is the number of edges in the induced subgraph on $N_{\lceil \ell/2\rceil}(u) \cup N_{\lfloor \ell/2\rfloor}(v)$.*

PROOF. If A reaches size f then we correctly accept, since $f_\ell(u, v) \ge f$. If A is maximal then the edges used in A form an ℓ-short u–v cut, since every short path must intersect some path in A. This cut has fewer than $f\ell$ edges, since A contains fewer than f paths, which are all ℓ-short. This cut implies that $c_\ell(u, v) \le f\ell$, and combining this with (6) implies that the algorithm `Max Short Flow` computes a $(1-\varepsilon)^2$-approximation $\hat{f}_\ell(u, v)$ in $((2/\varepsilon^2)f\ell\log\ell)$ iterations. At this point, if $f_\ell(u, v) \ge f$, then $\hat{f} \ge (1-\varepsilon)^2 f$ by the approximation guarantee, and so we accept. If $f_\ell(u, v) < (1-\varepsilon)^2 f$, then $\hat{f}_\ell(u, v) < f_\ell(u, v) < (1-\varepsilon)^2 f$, and so we reject. The running time is dominated by the time to run `Max Short Flow`, and the result follows. □

4. Separating Local and Global Edges.

In this section we present approximation algorithms for computing and appraximating $L_{f,\ell}(G)$, the largest (f, ℓ)-local subgraph in G. These algorithms for computing $L_{f,\ell}(G)$ will be used to create our graph decompositions in Section 5.

4.1. *The Extract Algorithm.* We first present a simple greedy algorithm `Extract` for computing $L_{f,\ell}(G)$. This basic version of the `Extract` algorithm appeared previously in [15].

`Extract`(f, ℓ). We are given an input graph G and parameters f and ℓ. Initially, set $H = G$. For each edge $e = (u, v)$ in H, check if u and v are (f, ℓ)-connected in H. If not, remove edge e from H. Repeat until no further edges can be removed, then output H.

THEOREM 5. *For any graph G and any (f, ℓ), the algorithm* `Extract`(f, ℓ) *returns $L_{f,\ell}(G)$, the unique maximal (f, ℓ)-connected subgraph of G.*

PROOF. Let e_i be the ith edge removed during the running of the `Extract` algorithm, and let $L = G\backslash\{e_1, \dots e_k\}$ be the graph output by the algorithm. The first edge e_1 removed by the algorithm is not (f, ℓ)-connected in G, and therefore it is not contained in any (f, l)-local subgraph of G. Assume for the sake of induction that each of the edges $e_1 \cdots e_i$ in not contained in any (f, l)-local subgraph of G. The next edge removed, e_{i+1}, is not contained in any (f, l)-local subgraph of $G\backslash\{e_1 \cdots e_i\}$ and is therefore not contained in any (f, l)-local subgraph of G. By induction, each edge not in L is not contained in any (f, ℓ)-connected subgraph of G. Since the algorithm cannot remove any additional edges from L, each edge in L is (f, ℓ)-connected in L, and so L is (f, l)-local. Therefore, L is the unique maximal (f, l)-local subgraph of G. □

The immediate corollary below describes what happens if we replace the exact local connectivity testing in `Extract` with an ε-approximate local connectivity testing algorithm.

COROLLARY 1. *If the algorithm* `Extract` *uses an ε-approximate local connectivity testing algorithm which accepts all vertex pairs that are (f, ℓ)-connected and rejects all vertex pairs that are not* $((1-\varepsilon)f, \ell)$*-connected, then* `Extract` *returns a graph $\hat{L}$ such that $L_{f,\ell}(G) \subseteq \hat{L} \subseteq L_{f(1-\varepsilon),\ell}(G)$.*

4.2. *An Approximate Extract Algorithm.* The algorithm `Extract` performs $O(m^2)$ local connectivity tests, where m is the number of edges in G. The number of local connectivity tests used can be reduced by using a standard random sampling approach if we are willing to accept local graphs that are approximations in an additional sense. We say $\hat{L}$ is an α-approximate local subgraph of G with respect to some deterministic connectivity test T if $\hat{L}$ contains every edge of G that passes the test T and if at most an α-fraction of the edges in L fail the test T. The following algorithm returns an α-approximate local graph for a given local connectivity test with probability $1 - \delta$:

```
Approximate Extract

We are given as input a graph $G$, a local connectivity test $T$, and parameters $\alpha$
    and $\delta$.
Pick an edge $e = (u, v)$ from $G$ uniformly at random.
    If $u$ and $v$ fail the test $T$, remove $(u, v)$ from $G$.
Repeat until no edge is removed for $(1/\alpha)\log(m/\delta)$ consecutive attempts,
then output the current graph.
```

Since at most m edges are removed from G and there are at most $(1/\alpha)\log(m/\delta)$ attempted removals for every edge removed, `Approximate Extract` performs at most $(m/\alpha)\log(m/\delta)$ local connectivity tests.

THEOREM 6. *With probability at least* $1 - \delta$, *the algorithm* `Approximate Extract` *returns an* α*-approximate local subgraph of* G *with respect to the local connectivity test* T.

PROOF. We say the algorithm is in phase i if $i - 1$ edges $e_1 \cdots e_{i-1}$ have been removed so far. Say that the algorithm has reached phase i. If the algorithm halts before phase $i + 1$ and outputs a graph that is not α-approximate for T, then $(1/\alpha)\log(m/\delta)$ edges were tested and passed the local connectivity test in phase i, but at least an α-fraction of the edges remaining in phase i do not pass the local connectivity test. The probability that this occurs is bounded by

$$(1-\alpha)^{(1/\alpha)\log(m/\delta)} \le e^{-\log(m/\delta)} \le \frac{\delta}{m}.$$

Since there are at most m phases of the algorithm, the probability that this occurs in any phase is at most δ. Note that we never remove an edge that passes the local connectivity test. The result follows.

COROLLARY 2. *If we run the algorithm* `Approximate Extract` *with an* ε*-approximate connectivity test, then with probability* $1 - \delta$ *we obtain a graph* $\hat{L}$ *which contains* $L_{f,\ell}$ *and where at most an* α*-fraction of edges are not* $(f(1-\varepsilon), \ell)$*-connected in* $\hat{L}$.

4.3. *Heuristic Versions of the Extract Algorithm.* In some applications it may be sufficient to create a local/global partition by defining L be the edges that are (f, ℓ)-connected in G. In this case L will not necessarily be an (f, ℓ) local graph, but the subgraph L still has the robustness properties described in Section 2.5. This approach eliminates the recursive part of `Extract` and requires only m local connectivity tests. It should be considered for large graphs.

An alternative to computing the maximum short flow between u and v is to compute the max flow between u and v within the incuded subgraph on $N_{\ell/2}(u) \cup N_{\ell/2}(v)$. The size of this flow can be significantly larger than the maximum ℓ-short flow between u and v, and we do not currently have any theoretical guarantees for local graphs based on this notion of flow. However, the time savings could be significant due to the high quality max flow implementations that are available.

For a few limited choices of parameters, testing local connectivity can be solved more easily. With the parameters $(f = 2, \ell)$, testing local connectivity can be accomplished by determining the distance between u and v in $G\backslash(uv)$. For the parameters $(f, \ell = 3)$, testing local connectivity can be accomplished by computing a maximum bipartite matching.

To choose parameters in practice we generally pick a small fixed value of ℓ between 3 and 6. We then start with $f = 3$ and try increasing f until a large number of edges are classified as global, but the components in the local graph are not too small.

5. Applying Local/Global Partitions in Graph Drawing.

We combine a local/global partition with a standard graph layout method to produce improved drawings for a variety of graphs. The method we use is the *neato* software package from AT&T [23], which implements a version of the Kamada–Kawai algorithm [24]. Our basic approach is to partition the edge set of the graph into local and global edges and assign shorter target lengths to local edges. This partitioning method can conceivably be combined with any layout method that allows the user to specify either the target lengths or spring constants of edges.

To provide motivation, we first demonstrate this approach on a few examples from the hybrid model. We then present a more complicated partition and length assignment rule to be used on real-world examples, and we show the results for certain biological networks and collaboration graphs.

5.1. *A Simple Partition and Motivating Examples from the Hybrid Model.*

Here we demonstrate the application of a straightforward local/global partition to a few examples from the hybrid model. To obtain the results depicted in the figures, we apply the `Extract` algorithm with appropriate parameters f and ℓ to an input graph H to obtain the local subgraph L, and let $G = H\backslash L$ be the global subgraph. We assign target length 1 to edges in L, and target length 100 to edges in G. In the figures the local edges are darker.

All the examples described in this section consist of an underlying local graph augmented with random edges, as in the hybrid model. In all the examples, the underlying graph is highly locally connected, and the Kamada–Kawai algorithm on its own will produce good drawings of the local graph. However, the Kamada–Kawai algorithm will not necessarily produce good drawings of the augmented graphs, even though the local graph may still be easily recoverable from the augmented graph with the `Extract` algorithm. Combined with our partitioning scheme, the Kamada–Kawai algorithm produces drawings of the augmented graph similar to those of the local graph. This is arguably the best that can be hoped for with a graph from the hybrid model.

The first example is a modified random geometric graph $G(n, d, p)$. A random geometric graph is created by choosing n points $x_1 \cdots x_n$ uniformly from the unit square and creating an edge between each pair $x_i x_j$ if and only if $d(x_i, x_j) \leq d$. We then augment

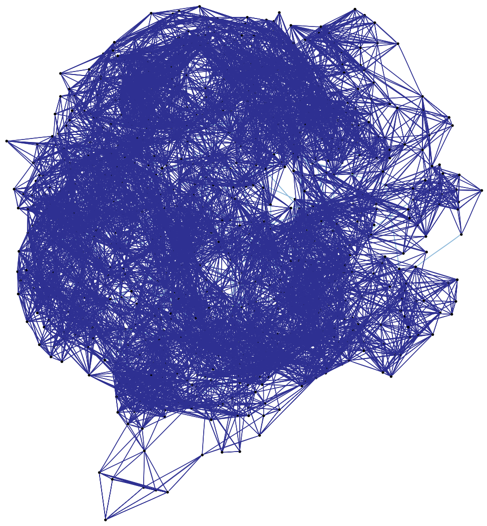

Fig. 2. A modified geometric graph G(n,d,p) with all target lengths equal.

this graph with the edge set of a random graph $G(n, p)$. We also consider the hypercube Q_n and the grid graph $P_n \times P_n$ augmented with a random graph $G(n, p)$. Note that Q_n is an $(n, 3)$-local graph. The grid graph $P_n \times P_n$ is a $(1, 3)$-local graph, and all edges not on the border are $(3, 3)$-connected. The infinite grid graph and the toroidal grid graph $C_n \times C_n$ are $(3, 3)$-connected. The random geometric graph is displayed in Figures 2 and 3 with the local edges darker than the global edges. The other examples are included in Section 6.

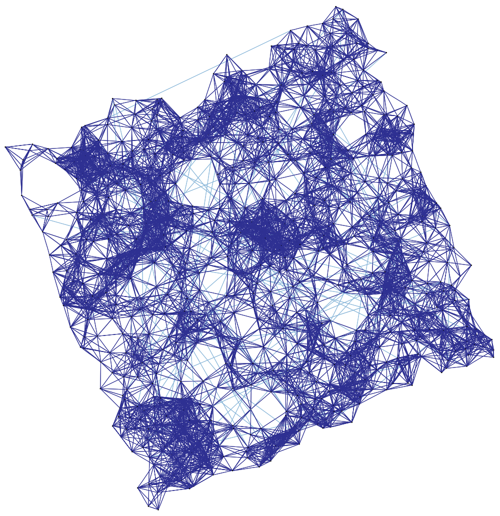

Fig. 3. The same graph with target lengths set by the local/global partition, using $(f = 3, \ell = 3)$.

5.2. *A More Sophisticated Partition for Real-World Examples.* The simple local/global partition $H = [L, G]$ and length assignment scheme presented in the previous section will not produce desirable results for real examples. We present a different partition which attempts to address the main problems that arise. Some of the modifications are motivated solely by practical concerns in order to produce reasonable drawings while still incorporating the local/global partition, but many of the obstacles to adapting the local/global partition to real examples are predicted by models of random power law graphs.

We expect a random power law graph to have an "octopus" structure. In particular, a random power law graph with exponent β, where $2 < \beta < 3$, contains a dense subgraph, called the "core," with $n^{c/\log\log n}$ vertices. Almost all vertices are within distance $\log\log n$ of the core although there are vertices at distance $\log n$ from the core [14]. This octopus structure is also found in real examples, for example the Yahoo Instant Messenger Graph [25]. We also expect a random power law graph to have sparse tree-like components. In the simple local/global partition, edges in these sparse components are likely to be classified as global edges. We wish to treat these edges differently from global edges in the denser parts of the graph, so the first step in forming our modified partition will be to separate the graph into two pieces using the k-core decomposition. The k-core of a graph is the unique maximal induced subgraph where every vertex has degree at least k. It has been studied in the context of random graphs [26] and has also been used in graph drawing applications [27]. We separate the edges of G into two sets, *tentacle* edges T and *core* edges K, by letting K be the k-core of G for some small value of k and letting $T = G \backslash K$.

We then divide the core into local and global edges with the `Extract` algorithm to obtain a partition of K into L and G. In real examples, unlike in the examples from the hybrid model, the local graph L is likely to be disconnected with a large number of connected components. We further divide the global edges into two sets, depending on whether the endpoints lie in a single local component or have endpoints in different local components. Edges whose endpoints lie in the same local component of L are placed into the set S of *shortcut* edges. We set the target lengths of shortcut edges very high, since they are the analogues of the global edges in the examples from the hybrid model. Those global edges whose endpoints lie in different connected components of the local graph L are placed into the set C of *connector* edges. It is crucial to set the target lengths of the connector edges to ensure that the local components are separated in the resulting drawing but are not placed extremely far apart. We set the target lengths of the connector edges to moderate values that depend on the sizes of the components being bridged.

The result is a partition of the edges of G into four sets: tentacle, local, shortcut, and connector edges. We use the following length assignment: The edges in T are given target length 1. The edges in L are given target length c, some small constant which determines the relative size of the tentacles. The shortcut edges in S are given target length $100 \cdot c$. A connector edge in C bridging components of size a and b is assigned target length $(ab)^{(1/4)} \cdot c$.

6. Examples.

We have implemented the `Extract` algorithm and produced several examples using both the simple $[L, G]$ partition and the $[T, L, C, S]$ partition (Figures 4

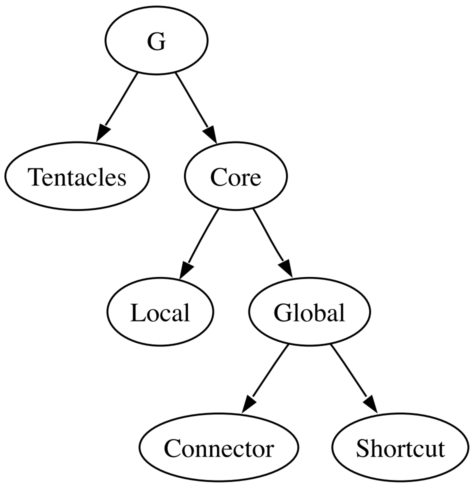

Fig. 4. The $[T, L, C, S]$ partition.

and 5). Figures 6–9 are the grid and hypercube examples described in Section 5.1. In these figures the edges in L are drawn in darker blue (or black), and the edges of G in red (or gray).

Grossman [28] has graciously provided data from a collaboration graph of the second kind, where each vertex represents an author and each edge represents a joint paper with two authors. The graph in Figures 12 and 13 is a random induced subgraph on the vertices in the collaboration graph with degree at least 18. The graph in Figures 10 and 11 is a subgraph of a biological network that encodes protein–DNA interactions. In these figures the local edges are drawn darker than the other edges.

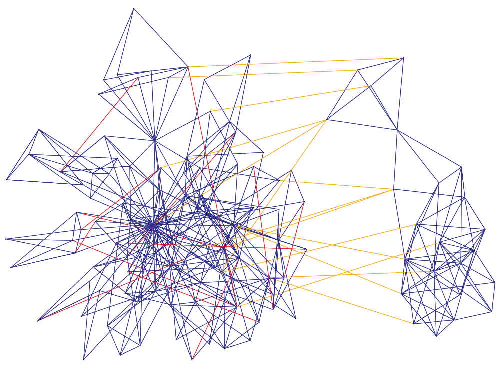

Fig. 5. Two local components in the collaboration graph with shortcut edges in red and connector edges in orange.

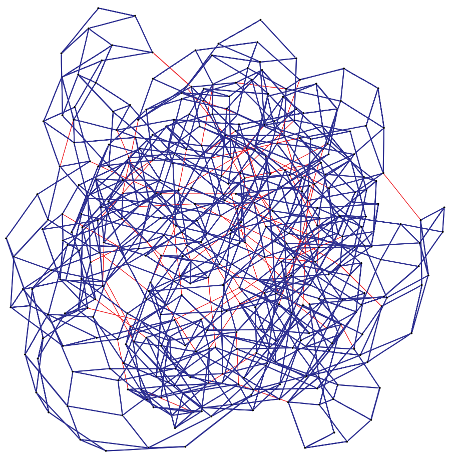

Fig. 6. The grid graph $P_{20} \times P_{20}$ plus a random graph, with all target lengths equal.

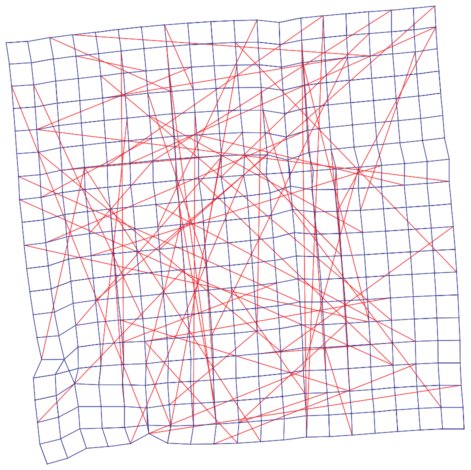

Fig. 7. The same graph with target lengths set by the local/global partition, using $(f = 1, \ell = 3)$.

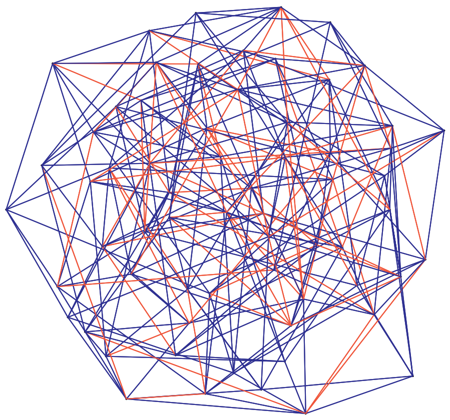

Fig. 8. The hypercube Q_6 plus a random graph, with all target lengths equal.

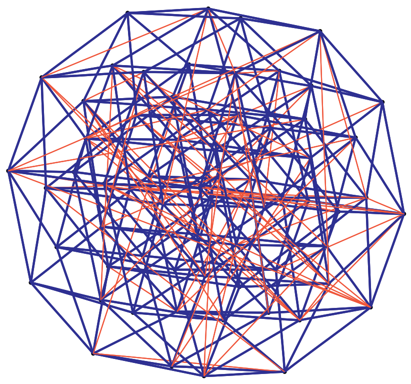

Fig. 9. The same graph with target lengths set by the local/global partition, using $(f = 6, \ell = 3)$.

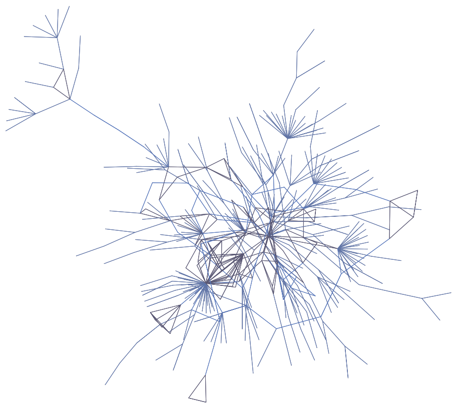

Fig. 10. Subgraph of a biological network, with all target lengths equal.

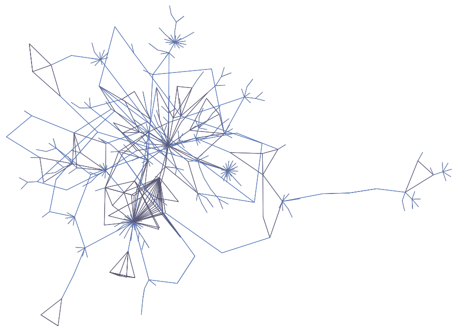

Fig. 11. Subgraph of a biological network, using $[T, L, S, C]$ partition.

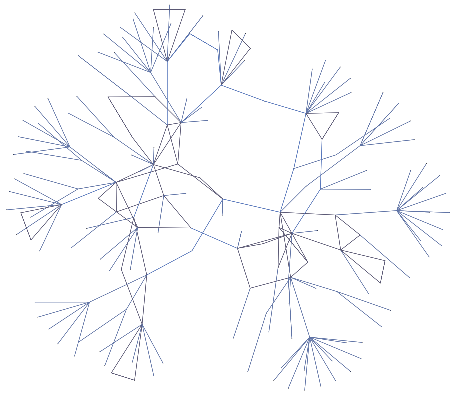

Fig. 12. A subgraph of the collaboration graph, with all target lengths equal.

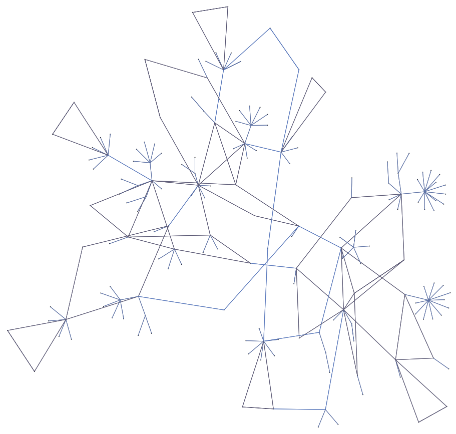

Fig. 13. A subgraph of the collaboration graph, using $[T, L, S, C]$ partition.

References

[1] L. A. Adamic and B. A. Huberman, Growth dynamics of the World Wide Web, *Nature* **401**, September 9, 1999, p. 131.

[2] W. Aiello, F. Chung and L. Lu, A random graph model for massive graphs, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, 2000, pp. 171–180.

[3] R. B. R. Azevedo and A. M. Leroi, A power law for cells, *Proc. Natl. Acad. Sci. USA* **98**(10) (2001), 5699–5704.

[4] A. Barabási and R. Albert, Emergence of scaling in random networks, *Science* **286** (1999), 509–512.

[5] A. Barabási, R. Albert and H. Jeong, Scale-free characteristics of random networks: the topology of the world wide web, *Physica A* **272** (1999), 173–187.

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tompkins and J. Wiener, Graph structure in the Web, *Proceedings of the WWW9 Conference*, May, 2000, Amsterdam. Paper version appeared in *Computer Networks* **33**(1–6) (2000), 309–321.

[7] K. Calvert, M. Doar and E. Zegura, Modeling Internet topology. *IEEE Communications Magazine* **35**(6) (1997), 160–163.

[8] C. Cooper and A. Frieze, On a general model of web graphs, *Random Structures and Algorithms* **22** (2003), 311–335.

[9] M. Faloutsos, P. Faloutsos and C. Faloutsos, On power-law relationships of the Internet topology, *Proceedings of the ACM SIGCOM Conference*, Cambridge, MA, 1999.

[10] S. Jain and S. Krishna, A model for the emergence of cooperation, interdependence, and structure in evolving networks, *Proc. Natl. Acad. Sci. USA* **98**(2) (2001), 543–547.

[11] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, The web as a graph: measurements, models and methods, *Proceedings of the International Conference on Combinatorics and Computing*, 1999.

[12] S. R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, Extracting large-scale knowledge bases from the web, *Proceedings of the 25th VLDB Conference*, Edinburgh, 1999.

[13] M. E. J. Newman, The structure of scientific collaboration networks, *Proc. Natl. Acad. Sci. USA* **98**(2) (2001), 404–409.

[14] F. Chung and L. Lu, Average distances in random graphs with given expected degree sequences, *Proceedings of National Academy of Science*, **99** (2002).

[15] R. Andersen, F. Chung and L. Lu, Analyzing the small world phenomenon using a hybrid model with local network flow, *Proceedings of the Third Workshop on Algorithms and Models for the Web-Graph*, 2004.

[16] N. Garg and J. Könemann, Faster and simpler algorithms for multicommodity flow and other fractional packing problems, Technical Report, Max-Planck-Institut fur Informatik, Saarbrucken, Germany (1997).

[17] A. Itai, Y. Perl and Y. Shiloach, The complexity of finding maximum disjoint paths with length constraints, *Networks* **12** (1982).

[18] S. Plotkin, D. B. Shmoys and E. Tardos, Fast approximation algorithms for fractional packing and covering problems, *FOCS*, 1991, pp. 495–504.

[19] L. Lovász, V. Neumann-Lara and M. Plummer, Mengerian theorems for paths of bounded length, *Periodica Mathematica Hungaria* **9** (1978).

[20] S. Boyles and G. Exoo, On line disjoint paths of bounded length, *Discrete Mathematics* **44** (1983).

[21] G. Baier, Flows with Path Restrictions, Ph.D. thesis, Technische Universität Berlin, 2003.

[22] P. Erdős and T. Gallai, Gráfok előírt fokú pontokkal (Graphs with points of prescribed degrees, in Hungarian), *Matematikai Lapok* **11** (1961), 264–274.

[23] S. C. North, Drawing graphs with NEATO, *NEATO User Manual*, April 26, 2004.

[24] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs, *Information Processing Letters* **31**(1) (1989), 7–15.

[25] K. J. Lang, *Fixing Two Weaknesses of the Spectral Method*, Advances in Neural Information Processing Systems 18, MIT Press, Cambridge, MA, 2006.

[26] B. Bollobás, The evolution of sparse random graphs, in *Graph Theory and Combinatorics*, *Proceedings of the Cambridge Combinatorial Conference in Honour of Paul Erdős*, Academic Press, New York, 1984, pp. 35–57.

[27] M. Baur, U. Brandes, M. Gaertler and D. Wagner, Drawing the AS graph in 2.5 dimensions, Presented at the 12th International Symposium on Graph Drawing, 2004.

[28] J. Grossman, P. Ion and R. De Castro, Facts about Erdős numbers and the collaboration graph, http://www.oakland.edu/∼grossman/trivia.html.