Distributed Random Digraph Transformations for Peer-to-Peer Networks

Peter Mahlmann*
Heinz Nixdorf Institute and
Computer Science Department
University of Paderborn
Germany
mahlmann@upb.de

Christian Schindelhauer*
Institute of Computer Science
University of Freiburg
Germany
schindel@informatik.uni-freiburg.de

ABSTRACT

We present a local random graph transformation for weakly connected multi-digraphs with regular out-degree which produces every such graph with equal probability. This operation, called Pointer-Push&Pull, changes only two neighboring edges. Such an operation is highly desirable for a peer-to-peer network to establish and maintain well connected expander graphs as reliable and robust network backbone. The Pointer-Push&Pull operation can be used in parallel without central coordination and each operation involves only two peers which have to exchange two messages, each carrying the information of one edge only.

We show that a series of random Pointer-Push&Pull operations eventually leads to a uniform probability distribution over all weakly connected out-regular multi-digraphs. Depending on the probabilities used in the operation this uniform probability distribution either refers to the set of all weakly connected out-regular multi-digraphs or to the set of all weakly connected out-regular edge-labeled multi-digraphs. In multi-digraphs multiple edges or self-loops may occur. In an out-regular digraph each node has the same number of outgoing edges.

For this, we investigate the Markov-Process defined by the Pointer-Push&Pull operation over the set of all weakly connected multi-digraphs. We show that a Pointer-Push&Pull operation — although preserving weak connectivity only — can reach every weakly connected multi-digraph. The main argument follows from the symmetry of the Markov-Process described by the Pointer-Push&Pull operation over the set of all weakly connected out-regular multi-digraphs.

*Partially supported by the DFG-Sonderforschungsbereich 376 and by the EU within 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems" (DELIS).

SPAA'06, July 30–August 2, 2006, Cambridge, Massachusetts, USA.

Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms, Network Problems; E.1 [**Data Structures**]: Graphs and Networks; C.2.4 [**Distributed Systems**]; G.3 [**Probability and Statistics**]: Markov Processes, Stochastic Processes

General Terms

Algorithms, Theory

Keywords

Peer-to-Peer Networks, Random Graphs, Expander Graphs, Distributed Algorithms

1. INTRODUCTION

Peer-to-peer networks continue pervading the Internet. Since the early start from the Napster network to the first true peer-to-peer network Gnutella [5] numerous approaches have been made both, theoretically and practically. In this paper, we concentrate on two important features necessary for successful peer-to-peer networks: Robust connectivity and randomness.

As an example consider the Gnutella network [5], which was the first peer-to-peer network using a random network structure. Studies reveal that Gnutella is not a truly random network, but a so-called Pareto (or power law) distributed graph [7, 15]. Compared to a random network, the degree distribution (i.e. the density function of neighbors) is skewed and also the diameter of the network is larger than in random networks with the same average out-degree. Gnutella would have been a better network if the peers would have been connected by a truly random network [11] (Clearly this does not solve the problem of efficient search which is the main problem of Gnutella [16]).

The idea of using random networks for peer-to-peer networks also appears in the peer-to-peer network design suite JXTA of Sun Microsystems. JXTA aims at connecting all peers by a random network to provide robustness, i.e. preventing peers or groups of peers from being disconnected. So, JXTA forms a design tool for peer-to-peer network applications like distributed search for web-sites [1]. The robustness and reliability of such random backbones is affirmed in [9, 10]. The usage of random networks can be further motivated by results of graph theory, which show that random graphs provide connectivity and expansion even for very small degrees [18]. Informally, the expansion property

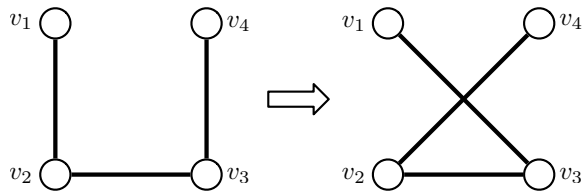

Figure 1: The 1-Flipper operation.

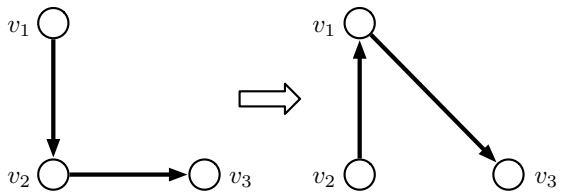

Figure 2: The Pointer-Push&Pull operation.

means that all partitions of the set of nodes of a graph have a number of edges on the cut which is proportional to the smaller set of the partitions. Such expander graphs have a lot of desirable properties, e.g. logarithmic diameter, large conductance, and short mixing times of random walks [4].

According to [12], a graph transformation for the maintenance of random graphs should fulfill the following requirements to be suitable for the adaption in peer-to-peer networks:

Soundness: No transformation maps to graphs which are not in the domain space, e.g. the connectivity of the graph has to be preserved.

Generality: The random transformation process does not converge to a specific graph. All graphs are *reachable* and in the limit all graphs occur with non-zero probability. This requirement can be tightened to *uniform generality* where in the limit all graphs occur with the same probability.

Feasibility: The graph transformation can be described by a simple (distributed) routine. Its implementation in a distributed network should be straightforward.

Convergence rate: Only a small number of transformations is necessary to achieve an approximation of the ultimate distribution of all graphs.

Pandurangan, Rhagavan and Upfal [14] presented a first approach to build low-diameter networks with expansion property. Their approach ensures that the network is connected and has logarithmic diameter with high probability. An even simpler fully decentralized operation with guaranteed connectivity was presented with the 1-Flipper [12] — a sound, general and feasible random transformation for the domain of undirected connected regular graphs. The 1-Flipper essentially chooses a random path of length three and flips the two outer edges if the upcoming edges do not already exist (see Fig. 1). By definition the 1-Flipper operation preserves connectivity and regularity. Furthermore, the 1-Flipper, started with any regular connected undirected graph, in the limit constructs all such graphs with the same probability. Currently no tight bounds for the convergence rate of this process are known. However, if one extends the length of the path to $k \in \Theta(d^2 n^2 \log 1/\epsilon)$ edges, it is shown in [12] that the graph converges in $\mathcal{O}(dn)$ rounds to an expander graph (here n denotes the number of nodes and d the degree of the graph). In a recent work [3] it is claimed that the convergence rate of the 1-Flipper can be bounded by a high degree polynomial. However, we conjecture that the 1-Flipper operation converges in time $\mathcal{O}(dn \log n)$.

For the use in peer-to-peer networks, the 1-Flipper can be improved. First, the 1-Flipper involves the active participation of four peers. Second, the maintenance of undirected, or to be more precise bi-directed, graphs is costly and not necessary. In practice digraphs are completely sufficient. Peers do not need to know who is pointing towards them as long as the network is connected and robust. Note, that there are several peer-to-peer networks which use digraphs, see [17, 8, 13, 6] for example.

In this paper we concentrate on random graph transformations for weakly connected multi-digraphs and present the Pointer-Push&Pull operation. Its main advantage is simplicity: For a local update operation only two peers need to exchange two messages. Using Markov theory, we show that Pointer-Push&Pull operations will eventually generate all weakly connected out-regular multi-digraphs with the same probability. Interestingly this is not the case if Pointer-Push&Pull is applied to out-regular simple digraphs. Then we show that a slight change of the probability distribution for the choice of edges gives another terminal distribution over multi-digraphs with simple digraphs occurring with higher probability.

At last we discuss how to implement Pointer-Push&Pull in a distributed network and solve the problems arising by concurrent operations. Based on the Pointer-Push&Pull operation a prototypical peer-to-peer network has been implemented. This implementation as well as numerous simulations let us conjecture that the convergence rate towards the uniform probability distribution over all connected graphs is fast, i.e. we conjecture $\mathcal{O}(dn \log n)$ parallel operations. However, the question regarding convergence speed is open.

1.1 Notations

We now introduce some notations which are used throughout this paper. By $\log n = \log_2 n$ we denote the dual logarithm function. The term "with high probability" (w.h.p.) describes a probability $p \geq 1 - n^{-c}$ and "asymptotically almost surely" (a.a.s.) is a probability $p \geq 1 - o(1)$. Furthermore, we use the following definitions for several classes of directed graphs, which we will refer to as digraphs from now on.

Definition 1 (Simple Digraph)
A simple digraph $G = (V, E)$ is defined by a node set $V = \{1, \ldots, n\}$ and a set of directed edges $E = \{(u, v) : u, v \in V, u \neq v\}$.

A digraph is *strongly connected* if for all pairs of nodes there exists a directed path in E and *weakly connected* if there exists a path neglecting the direction of the edges for each pair of nodes. For $v \in V$ we define $N^+(v) = \{w : (v, w) \in E\}$ and $N^-(v) = \{u : (u, v) \in E\}$ to refer to the successor and predecessor nodes of v in G. A digraph is called d-regular if $\forall v \in V : |N^-(v)| = |N^+(v)| = d$. Furthermore, we say a digraph is d-out-regular if $\forall v \in V : |N^+(v)| = d$.

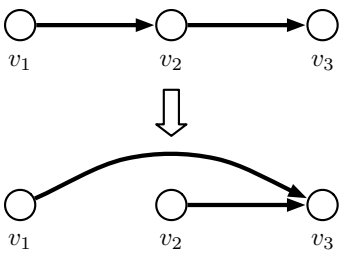

Figure 3: The Pointer-Push operation.

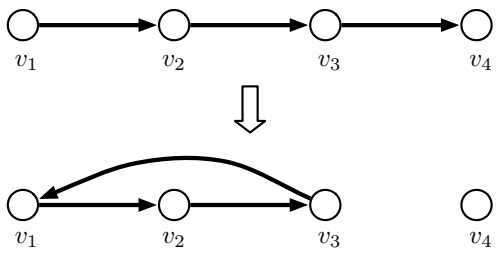

Figure 4: The Pointer-Pull operation.

The usage of a multiset for the set of edges in digraphs will allow us to define a more general model of digrahs. Therefore, we give a formal definition of multisets.

Definition 2 (Multiset)
A set E and a function $\#_E : E \to \mathbb{N}_0$, specifying the multiplicity of its elements, define a multiset. For $e \in E$ we write $e \in_k E$ if $\#_E(e) = k$. This implies $e \in_0 E$ for all $e \notin E$. The cardinality of a multiset E is defined as $|E| = \sum_{e \in E} \#_E(e)$.

For the subtraction on multisets we define $E' = E \setminus e$ such that $\#_{E'}(e) = \#_E(e) - 1$ if $\#_E(e) > 0$ and $\#_{E'}(e) = \#_E(e) = 0$ otherwise. The union of sets is defined analogously, i.e. $\#_{E'}(e) = \#_E(e)+1$. On basis of simple digraphs and multisets we can now define the more general model of multi-digraphs.

Definition 3 (Multi-Digraph)
A multi-digraph $G = (V, E, \#_E)$ is defined by a node set $V = \{1, \ldots, n\}$ and a multiset of directed edges $E = \{(u, v) : u, v \in V\}$ with $\#_E$ specifying the multiplicity of the edges.

In a multi-digraph self-loops (u, u) and multiple occurrence of edges are explicitly allowed, e.g. an edge (u, v) may occur twice. Analogous to simple digraphs, a multi-digraph is called d-regular if $\forall u \in V : \sum_{v \in V,(u,v) \in E} \#_E((u, v)) = \sum_{v \in V,(v,u) \in E} \#_E((v, u)) = d$ and called d-out-regular if $\forall u \in V : \sum_{v \in V,(u,v) \in E} \#_E((u, v)) = d$. Note, that if no slopes occur and the multiplicity of all edges is at most one then a multi-digraph describes a simple digraph. So, simple digraphs form a subset of multi-digraphs. As in case of simple digraphs we define the neighborhood of a node $v \in V$ as $N^+(v) = \{w : (v, w) \in E\}$. Note that $N^+(v)$ is not a multi-set and therefore the $|N^+(v)| < d$ is possible in a d-out-regular multi-digraph.

The operations we introduce in the following sections are so called *graph transformations*, i.e. they transform a graph of a specific domain to another graph of that domain. We now give a formal definition of a graph transformation.

Definition 4 (Graph Transformation)
A graph transformation is a random transition between the graphs of a specific domain $\mathcal{G}$, e.g. multi-digraphs. A graph $G \in \mathcal{G}$ is mapped to a set of other graphs in $\mathcal{G}$ such that

$$\sum_{G' \in \mathcal{G}} P[G \to G'] = 1,$$

where $G \to G'$ denotes that G is transformed to G'. A graph transformation describes a Markov chain, where the set of states equals the set of graphs in $\mathcal{G}$, i.e. each $G \in \mathcal{G}$ represents a state of the Markov chain. The transition matrix of the Markov chain is given by the transformation probabilities $P[G \to G']$ for all pairs $G, G' \in \mathcal{G}$.

2. THE POINTER-PUSH&PULL GRAPH TRANSFORMATION

Our goal is to design a graph transformation for connected digraphs which is sound, general and feasible. Since the model of multi-digraphs is less restrictive than the model of simple digraphs, we will first consider multi-digraphs and then show that our results can not be transfered to simple digraphs.

2.1 Multi-Digraphs

We start with some fundamental considerations about transformations of multi-digraphs. Considering a node v_1 of a multi-digraph G, there are basically two possibilities to change G:

- *Pointer-Push* — change the outgoing edges of the node, i.e. do a random walk v_1, v_2, v_3 in G and replace (v_1, v_2) with (v_1, v_3) (see Figure 3).
- *Pointer-Pull* — change the ingoing edges of the node, i.e. do a random walk v_1, v_2, v_3, v_4 in G and replace (v_3, v_4) with (v_3, v_1) (see Figure 4).

However, neither the Pointer-Push nor the Pointer-Pull operation meet our requirements for random graph transformations. Due to space restrictions we do not present the complete analysis of these operations here.

For the Pointer-Push operation it turns out that it is sound and feasible but not general, i.e. the graph will converge to a set of connected stars in the limit. This is because there is a non zero probability to generate a sink in the graph, i.e. a node with all edges pointing to itself. Once a node points to such a sink there is no possibility to remove this edge with a Pointer-Push operation anymore.

The situation of the Pointer-Pull operation is similar. The Pointer-Pull operation is not able to preserve connectivity in a graph and therefore is not sound. To see this, note that there is a non-zero probability for a node to create self-loops. Once all edges of a node are pointing to itself this node is disconnected from the rest of the graph. Without global knowledge such a situation is irrevocable and therefore the graph will consist only of disconnected nodes in the long run. A more detailed analysis of the Pointer-Push and the Pointer-Pull operation is presented in Appendix A.

We now show that a combination of these basic graph transformations, called Pointer-Push&Pull operation, overcomes the shortcomings of the Pointer-Push respectively Pointer-Pull operation. It is defined as follows.

Definition 5 (Pointer-Push&Pull Operation) *Let $G = (V, E, \#_E)$ be a d-out-regular multi-digraph and let nodes $v_1, v_2, v_3 \in V$ form a directed path $P = (v_1, v_2, v_3)$ in G. Then, the Pointer-Push&Pull operation $\mathcal{PP}_P$ transforms graph G to graph $\mathcal{PP}_P(G) = (V, E', \#_{E'})$ with*

$$E' = (E \setminus \{(v_1, v_2), (v_2, v_3)\}) \cup \{(v_1, v_3), (v_2, v_1)\} \ .$$

The Pointer-Push&Pull operation is illustrated in Figure 2. Note, that a Pointer-Push&Pull operation can be divided to a Pointer-Push operation and a Pointer-Pull operation. We start our analysis of this graph transformation with the following lemma.

Lemma 1 *The Pointer-Push&Pull operation is sound for the domain of weakly connected out-regular multi-digraphs.*

Proof. We have to show that the Pointer-Push&Pull operation preserves connectivity and out-degree of all nodes. For connectivity note that all participating nodes stay connected, what implies at least weak connectivity for the graph. Concerning the out-degree node v_1 as well as v_2 just replace one of their outgoing edges and therefore their outdegrees remain unchanged. □

Algorithm 1 Random Pointer-Push&Pull

$v_1 \leftarrow$ random node $\in V$
if random event with $p = \frac{|N^+(v_1)|}{d}$ occurs **then**
  $v_2 \leftarrow$ random node $\in N^+(v_1)$
  if random event with $p = \frac{|N^+(v_2)|}{d}$ occurs **then**
    $v_3 \leftarrow$ random node $\in N^+(v_2)$
    $E \leftarrow (E \setminus \{(v_1, v_2), (v_2, v_3)\}) \cup \{(v_1, v_3), (v_2, v_1)\}$

Algorithm 1 shows a randomized Pointer-Push&Pull operation. This random Pointer-Push&Pull operation chooses its starting node uniformly at random, then performs a random walk of length two with some probability. Recall that due to multi-edges $|N^+(v)|$ may be less than d in d-out-regular multi-digraphs. Therefore, a random Pointer-Push&Pull operation may cancel with a probability proportional to the number of multi-edges of v_1 and v_2. This specific definition of the random Pointer-Push&Pull operation is motivated by the following lemma.

Lemma 2 *The random Pointer-Push&Pull operation is symmetric for out-regular multi-digraphs, i.e. for two out-regular multi-digraphs G, G' the probability to transfom G to G' by a random Pointer-Push&Pull operation is the same as to transform G' to G by a random Pointer-Push&Pull operation, i.e.*

$$P[G \overset{\mathcal{PP}}{\to} G'] = P[G' \overset{\mathcal{PP}}{\to} G],$$

where $G \overset{\mathcal{PP}}{\to} G'$ denotes that G can be transformed to G' with a single Pointer-Push&Pull operation.

Proof. Let $P = (u, v, w)$ be the path of the Pointer-Push&Pull operation transforming G to G'. To transform G' back to G we need to apply a Pointer-Push&Pull operation $\mathcal{PP}_{P'}$ with $P' = (v, u, w)$. Note, that $\mathcal{PP}_{P'}$ is the only possiblity to transform G' back to G with a single operation. As noted above, the random Pointer-Push&Pull operation chooses its starting node uniformly at random and then chooses a neighboring node for two times with probability $p = 1/d$ each. This implies $P[G \overset{\mathcal{PP}}{\to} G'] = P[G' \overset{\mathcal{PP}}{\to} G] = 1/(nd^2)$. □

The following lemma names the set of graphs which can be reached by applying random Pointer-Push&Pull operations to an arbitrary starting graph repeatedly.

Lemma 3 *A series of random Pointer-Push&Pull operations can transform a graph to any other graph within the domain of out-regular weakly connected multi-digraphs.*

Proof. We show that any weakly connected d-out-regular multi-digraph $G = (V, E, \#_E)$ with $V = \{v_1, \ldots, v_n\}$ can be transformed to a canonical graph $G_C = (V, E_C, \#_{E_C})$ with

$$E_C = \{(v_1, v_1), (v_2, v_1), \ldots, (v_n, v_1)\} \ ,$$

and $\forall e \in E_C : \#_{E_C}(e) = d$, i.e. all edges are pointing to v_1. Then, the theorem follows since each Pointer-Push&Pull operation is reversible so that two arbitrary graphs can be transformed to each other using G_C as an intermediate state. We start with an arbitrary weakly connected d-out-regular multi-digraph G and increase the indegree of v_1 iteratively. This can be done by repeatedly applying the at each time first applicable of the following six transformations:

Case 1: v_1 has at least one edge (v_1, v_j) with $j \neq 1$

- **Case 1.1:** v_j has an edge (v_j, v_k) with $k \neq 1$ and $k \neq j$. Apply a Pointer-Push&Pull operation to the path $P = (v_1, v_j, v_k)$ (see Figure 5(a)).
- **Case 1.2:** v_j has an edge (v_j, v_1). Apply a Pointer-Push&Pull operation with $P = (v_1, v_j, v_1)$ (see Figure 5(b)).
- **Case 1.3:** v_j has an edge (v_j, v_j). Apply a Pointer-Push&Pull operation with $P = (v_1, v_j, v_j)$ (see Figure 5(c)).

Case 2: v_1 has no edge (v_1, v_j) with $j \neq 1$ and therefore has an edge (v_1, v_1)

- **Case 2.1:** There is a node v_j pointing to v_1 and a node v_k, with $k \neq j$, pointing to v_j. Apply Pointer-Push&Pull operations to the paths $P_1 = (v_j, v_1, v_1)$, $P_2 = (v_k, v_j, v_1)$, $P_3 = (v_1, v_j, v_k)$, and $P_4 = (v_1, v_k, v_1)$ (see Figure 6).
- **Case 2.2:** There is a node v_j pointing both to v_1 and v_k, $k \neq 1$. Furthermore, v_k points to a node v_l with $l \neq 1$. If we apply a Pointer-Push&Pull operation to the path $P_1 = (v_j, v_k, v_l)$, then we reach the start configuration of case 2.1 and can continue with the operations described there (see Figure 7).
- **Case 2.3:** There is a node v_j pointing to v_1 and itself. Apply Pointer-Push&Pull operations to the paths $P_1 = (v_j, v_1, v_1)$, $P_2 = (v_1, v_j, v_j)$, and $P_3 = (v_1, v_j, v_1)$ (see Figure 8).

Following this scheme, the starting graph G will be transformed to G_C in the limit. To see this, note that G_C is the only possible graph with indegree dn for node v_1. Furthermore, the six cases cover all possible arrangements of edges, so that always one of the cases can be applied unless G_C is reached already. □

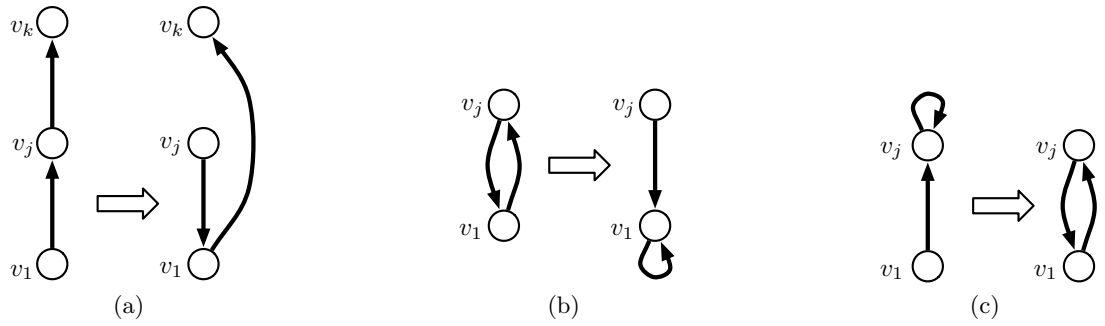

Figure 5: Cases 1.1 (a), 1.2 (b), and 1.3 (c) of Lemma 3 (from left to right).

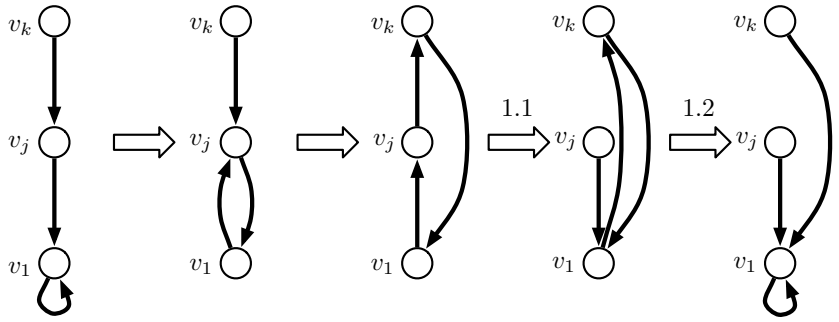

Figure 6: Case 2.1 of Lemma 3.

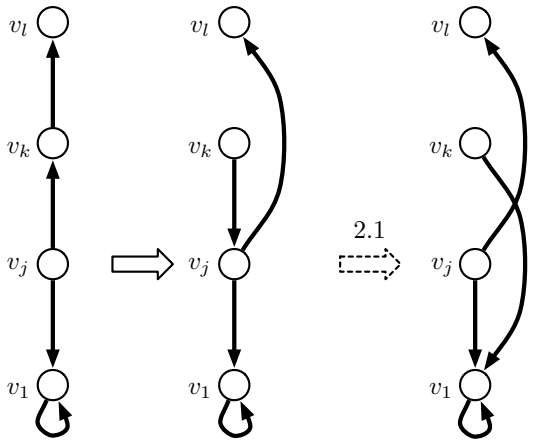

Figure 7: Case 2.2 of Lemma 3.

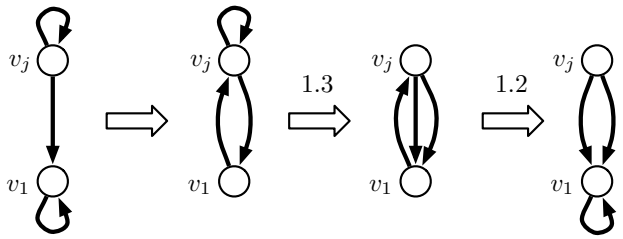

Figure 8: Case 2.3 of Lemma 3.

As a side effect, the proof of Lemma 3 delivers an upper bound: Starting with an arbitrary d-out-regular weakly connected multi-digraph, at most $10nd$ Pointer-Push&Pull operations are needed to reach any other d-out-regular weakly connected multi-digraph.

Combining our results, we are now able to show that the Pointer-Push&Pull operation provides uniform generality within the domain of d-out-regular weakly connected multi-digraphs.

Theorem 1 *Let G' be an arbitrary d-out-regular weakly connected multi-digraph with n nodes. Then, applying random Pointer-Push&Pull operations repeatedly to G' will construct every d-out-regular weakly connected multi-digraph with the same probability in the limit, i.e.*

$$\lim_{t\to\infty} P[G' \xrightarrow{t} G] = \frac{1}{|\mathcal{MDG}_{n,d}|},$$

where $\mathcal{MDG}_{n,d}$ denotes the set of all d-out-regular weakly connected multi-digraphs with n nodes.

Proof. Consider the Markov chain over the set of d-out-regular weakly connected multi-digraphs described by the random Pointer-Push&Pull operation. Lemma 2 implies that the transition matrix of the Markov chain is symmetric and therefore doubly stochastic. Lemma 3 shows that every state of the Markov chain is reachable. Furthermore, there is a non-zero probability to transform a graph to itself. This implies that some diagonal entries of the transition matrix are non-zero. Using these three properties of the transition matrix the theorem follows by applying essential results of Markov theory. □

2.2 Edge Labeled Multi-Digraphs

We have seen that the Pointer-Push&Pull operation generates out-regular weakly connected multi-digraphs with uniform probability. We now show that a slight change of the probability distribution for the choice of edges gives another terminal distribution over multi-digraphs with simple digraphs occurring with higher probability. To achieve this, we modify the Pointer-Push&Pull operation to work with out-regular edge labeled multi-digraphs, which we formally define as follows.

Definition 6 (Edge Labeled Multi-Digraph)
An edge labeled d-out-regular multi-digraph $G^* = (V, E^*)$ is defined by a node set $V = \{1, \ldots, n\}$ and an edge set $E^* = \{(u, v, i) : u, v \in V, i \in \{1, \ldots, d\}\}$, where i specifies the label of an edge. Here, we restrict to the labels $1, \ldots, d$ and unique labels for each outgoing edge of a node, i.e. $\forall u \in V, \forall i \in \{1, \ldots, d\} : \exists v \in V : (u, v, i) \in E^*$.

In d-out-regular edge labeled multi-digraphs we use the notation $N^+(v, i)$, $i \in \{1, \ldots, d\}$, $v \in V$ to refer to v's neighbor due to the i-th labeled edge. In addition we use $E^-(v) = \{(w, v, i) \in E^*\}$ to refer to the set of v's ingoing edges. Recall that $N^+(v, i) = N^+(v, j)$, $i \neq j$ is possible in a multi-digraph. Algorithm 2 shows the random Pointer-Push&Pull operation, modified for edge labeled multi-digraphs. For the sake of clarity and to simplify notation, we will refer to this algorithm as *labeled*-Pointer-Push&Pull and use *unlabeled*-Pointer-Push&Pull to refer to the previous section Pointer-Push&Pull algorithm.

Algorithm 2 Random labeled-Pointer-Push&Pull

$v_1 \leftarrow$ random node $\in V$
$i \leftarrow$ random number $\in \{1, ..., d\}$
$v_2 \leftarrow N^+(v_1, i)$
$j \leftarrow$ random number $\in \{1, ..., d\}$
$v_3 \leftarrow N^+(v_2, j)$
$E^* \leftarrow (E^* \setminus \{(v_1, v_2, i), (v_2, v_3, j)\})$
$\quad \cup \{(v_1, v_3, i), (v_2, v_1, j)\}$

The proof for preserving connectivity and out-degrees of the unlabeled-Pointer-Push&Pull can be transfered directly to the labeled-Pointer-Push&Pull operation. We now show that the labeled-Pointer-Push&Pull operation is symmetric within the domain of out-regular weakly connected edge labeled multi-digraphs.

Lemma 4 *The labeled-Pointer-Push&Pull operation $\mathcal{PP}^*$ is symmetric within the domain of out-regular weakly connected edge labeled multi-digraphs. That is, for two arbitrary graphs G_1^*, G_2^* of this domain*

$$P[G_1^* \xrightarrow{\mathcal{PP}^*} G_2^*] = P[G_2^* \xrightarrow{\mathcal{PP}^*} G_1^*].$$

Proof. According to Algorithm 2 G_1^* and G_2^* differ in exactly two edges. More precisely G_1^* has edges (v_1, v_2, i), (v_2, v_3, j) and G_2^* has edges $(v_1, v_3, i), (v_2, v_1, j)$. The only way to transform G_2^* back to G_1^* with a single operation is a labeled-Pointer-Push&Pull operation using the edges (v_2, v_1, j) and (v_1, v_3, i). Now observe that the starting node of the labeled-Pointer-Push&Pull operation is chosen uniformly at random, i.e. with $p = 1/n$. So, the algorithm does a random walk by choosing two edges uniformly at random, implying $P[G_1^* \xrightarrow{\mathcal{PP}^*} G_2^*] = P[G_2^* \xrightarrow{\mathcal{PP}^*} G_1^*] = 1/(nd^2)$. □

Similar as in case of the unlabeled-Pointer-Push&Pull operation, every out-regular weakly connected edge labeled multi-digraph can be reached by a series of labeled-Pointer-Push&Pull operations. This can be shown by transferring the proof of Lemma 3 to edge labeled multi-digraphs. To see this, note that using the proof of Lemma 3 any starting graph will be transformed to the canonical graph G_C — regardless of its labeling. Furthermore, the labeling in G_C can be neglected, since all edges point to the same node. Therefore, the proof also holds for the labeled-Pointer-Push&Pull operation.

Finally, these results lead to the following theorem showing that the labeled-Pointer-Push&Pull operation provides uniform generality within the domain of out-regular weakly connected edge labeled multi-digraph.

Theorem 2 *Let G_0^* be an d-out-regular weakly connected edge labeled multi-digraph with n nodes. Then, applying random labeled-Pointer-Push&Pull operations repeatedly to the graph will construct every d-out-regular weakly connected edge labeled multi-digraph with the same probability in the limit, i.e.*

$$\lim_{t\to\infty} P[G_0^* \xrightarrow{t} G^*] = \frac{1}{|\mathcal{MDG}^*_{n,d}|},$$

where $\mathcal{MDG}^*_{n,d}$ denotes the set of all d-out-regular weakly connected edge labeled multi-digraphs.

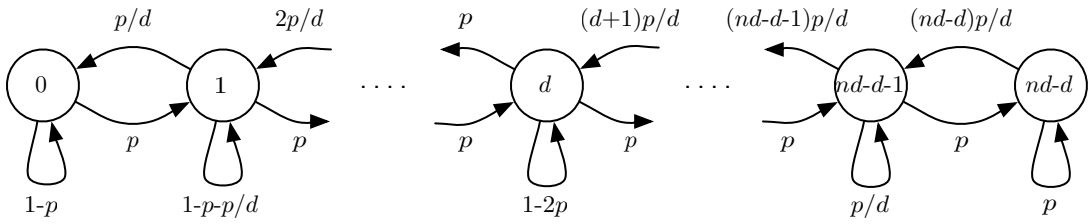

Figure 9: A Markov chain modeling the change of a node's indegree.

PROOF. The proof is essentially the same as the proof of Theorem 1. The transition matrix of the Markov chain described by the labeled-Pointer-Push&Pull operation over all $G^* \in \mathcal{MDG}^*_{n,d}$ is doubly stochastic and every state is reachable. Furthermore the labeled-Pointer-Push&Pull operation does not change the edge set if the operation chooses a self-loop for two times, implying non-zero diagonal entries. Therefore, the theorem follows by essential results of Markov theory. □

At first glance this result does not seem to be more powerful than the uniform generation of d-out-regular weakly connected multi-digraphs, shown by Theorem 1. Note however, that each d-out-regular edge labeled multi-digraph can be transformed to a d-out-regular (unlabeled) multi-digraph G. For this, the set of triples in E^* is transformed to a multi-set by omitting the labels, i.e. the third element of each triple. This leads to the following definition of an equivalence class.

Definition 7 (Equivalence Class)
The set of d-out-regular edge labeled multi-digraphs describing a d-out-regular (unlabeled) multi-digraph G, when omitting the edge labels, is called equivalence class of G: $[G]$.

Now, we can transfer Theorem 2 to the domain of unlabeled multi-digraphs using equivalence classes.

Theorem 3 *Let G be a d-out-regular weakly connected multi-digraph with n nodes. Then, applying random labeled-Pointer-Push&Pull operations to the graph repeatedly will construct every d-out-regular weakly connected multi-digraph G with probability $\sim \frac{1}{|[G]|}$.*

It remains to analyze the size of the equivalence classes, i.e. how many ways are there to label the edges of a d-out-regular multi-digraph with labels chosen according to Defintion 6. Straight forward combinatorics leads to the following sizes for equivalence classes in dependency of the number of multi edges in a graph.

Lemma 5 *Let G be a d-out-regular multi-digraph. Then, the size of the equivalence class of G is given by*

$$|[G]| = \prod_{u \in V(G)} \frac{d!}{M_0(u)! \prod_{i=1}^{d} (i!)^{M_i(u)}} ,$$

with $M_0(u)$ denoting the number of slopes, $M_1(u)$ denoting the number of single edges, $M_2(u)$ denoting the number of double edges, etc.

In other words, Lemma 5 shows: The lower the number of multi-edges in G is, the larger is the cardinality of its equivalence class $[G]$. This again implies that the labeled-Pointer-Push&Pull operation will generate a particular simple digraph with higher probability than a particular multi-digraph. Therefore, and because there is no additional overhead compared to the unlabeled-Pointer-Push&Pull operation, the labeled-Pointer-Push&Pull operation will be preferable in the majority of cases.

By definition the Pointer-Push&Pull operations guarantee out-regularity only. In many applications it is desirable to have regular digraphs. Therefore we now analyze the indegree distribution of the evolving graphs. On the one hand a node v increases its indegree by one if chosen as starting node.[1] This occurs with probability $p = 1/n$. On the other hand v's indegree is decreased only if v is chosen as second node of a Pointer-Push&Pull operation. This happens with probability proportional to v's indegree, i.e. $|E^-(v)|p/d$. Knowing this, the change of v's indegree can be modeled by a Markov chain with states $0, \ldots, nd - d$ representing the current indegree and transition probabilities as shown in Figure 9. Analyzing the stationary probability distribution of this Markov chain leads to an almost Poisson distribution with expectation d.

Lemma 6 *Starting with an arbitrary d-out-regular weakly connected edge labeled multi-digraph with n nodes and applying labeled-Pointer-Push&Pull operations repeatedly the indegrees of the nodes will be almost Poisson distributed, i.e. a node will have indegree k with probability*

$$p_k = \frac{d^k}{k!} \left(\sum_{i=0}^{nd-d} \frac{d^i}{i!} \right)^{-1} .$$

2.3 Simple Digraphs

Although the labeled-Pointer-Push&Pull operation generates simple digraphs with higher probability than the unlabeled-Pointer-Push&Pull operation, the following theorem shows that simple digraphs are still outnumbered.

Theorem 4 *The fraction p^* of d-out-regular edge labeled simple digraphs with n nodes in all d-out-regular edge labeled multi-digraphs with n nodes is bounded by*

$$e^{-\frac{d^2}{1-\frac{d}{n}}} \leq p^* \leq e^{-d} .$$

[1]Actually v's indegree will not increase if the first edge chosen is a self-loop. However the expected overall number of self-loops in a d-out-regular multi-digraph is constant, or to be more specific d. So, we neglect the constant number of affected nodes in our analysis and assume that the node has no self-loops during the graph transformations.

PROOF. In a d-out-regular edge labeled simple digraph neither self-loops nor multi edges occur. This implies that there are $(n-1)!/(n-d-1)!$ possibilities for a nodes neighborhood while there are n^d possibilities if self-loops and multi-edges are allowed. This gives an upper bound of

$$\left(\frac{(n-1)!}{(n-d-1)!n^d}\right)^n \leq \frac{(n-1)^{dn}}{n^{dn}} = \left(1-\frac{1}{n}\right)^{dn} \leq e^{-d} .$$

For the lower bound of d-out-regular edge-labeled simple multi-digraphs, observe that $\frac{(n-1)!}{(n-d-1)!} \geq (n-d)^d$. This implies

$$\left(\frac{(n-1)!}{(n-d-1)!n^d}\right)^n \geq \frac{(n-d)^{dn}}{n^{dn}} = \left(1-\frac{d}{n}\right)^{dn} = \left(1-\frac{d}{n}\right)^{(\frac{n-d}{d})\frac{d^2n}{n-d}} \geq e^{-\frac{d^2n}{n-d}} ,$$

and thus proves the theorem. □

Given a node the probability of being the source of only simple edges (not multiple edges or self-loops) is quite high, i.e. at least $1-\frac{d}{n-d-1}$ (which suffices practical needs). Whereas, if we consider the complete graph, the fraction of simple digraphs created by the labeled-Pointer-Push&Pull operation is rather small, i.e. decreasing exponentially with the degree. Since multiple edges are an unnecessary waste of resources one might want to generate simple digraphs only. A straight forward solution is to modify the labeled- or unlabeled-Pointer-Push&Pull operation such that it is only applied if the resulting digraph is simple. We call this modified graph transformation *simple*-Pointer-Push&Pull. Unfortunately, the simple-Pointer-Push&Pull operation is not general for the domain of simple digraphs. To see this consider any symmetric digraph, i.e. a digraph where for each edge (u,v) also the edge (v,u) is in the edge set. In such digraphs no simple-Pointer-Push&Pull operation can be applied, since each operation would either create a multi-edge or a self-loop and therefore leave the domain of simple digraphs. The only way to reach all simple digraphs using Pointer-Push&Pull operations is to allow the multiple occurrence of edges, i.e. the use of multi-digraphs.

3. POINTER-PUSH&PULL IN PEER-TO-PEER NETWORKS

The labeled- and unlabeled-Pointer-Push&Pull operations are in particular useful for the maintenance of distributed random networks, as they are used in peer-to-peer networks for example. Applying Pointer-Push&Pull operations ensures that the network stays truly random, even if peers join, leave, or fail in non-random fashion.

The Pointer-Push&Pull operation directly competes with the 1-Flipper operation we introduced in [12]. Both graph transformations are able to maintain truly random networks, yet in different domains. The 1-Flipper (see Figure 1) operation maintains (undirected) connected regular graphs, whereas the Pointer-Push&Pull operation maintains out-regular weakly connected multi-digraphs. While the domain of the 1-Flipper may be the traditional one for random networks, there is a strong argument in favor of the Pointer-Push&Pull operation and the use multi-digraphs: a Pointer-Push&Pull operation needs only two nodes to communicate per operation, while the 1-Flipper operation needs four nodes to communicate with each other. This implies lower network overhead and improved locality for each operation.

We will now discuss how to implement Pointer-Push&Pull operations in a distributed network without central coordination. A Pointer-Push&Pull operation on the path $P = (v_1, v_2, v_3)$ consists of the following sequential steps:

Step 1 v_1 requests a random neighbor from v_2

Step 2 v_2 replaces v_3 by v_1 in its neighborhood list and sends the ID of v_3 to v_1

Step 3 v_1 receives the ID of v_3 from v_2 and replaces v_2 by v_3 in its neighborhood list

These three steps involve only two network operations between v_1 and v_2. This shows, that a Pointer-Push&Pull operation does not introduce additional overhead compared to periodical verification of the neighborhood, which is mandatory in dynamic networks.

In the previous section we have shown that a single Pointer-Push&Pull operation never disconnects a network. However, things are different when there are concurrent Pointer-Push&Pull operations with intersecting paths. They bear the risk of disconnecting a network. To see this, consider a directed path (s,t,u,v) in the network and two concurrent Pointer-Push&Pull operations $\mathcal{PP}_P$ and $\mathcal{PP}_{P'}$ with $P = (s,t,u)$ and $P' = (t,u,v)$. Now consider the following situation. $\mathcal{PP}_{P'}$ has removed edge (u,v) and created edge (u,t). At this point of time $\mathcal{PP}_P$ starts and finishes before $\mathcal{PP}_{P'}$ continues, i.e. $\mathcal{PP}_P$ creates edges (s,u), (t,s) and removes edges (s,t), (t,u). Then, $\mathcal{PP}_{P'}$ can not be finished since $\mathcal{PP}_{P'}$ will try to remove edge (t,u) which no longer exists. Even worse, the resulting network is possibly disconnected since there is no path between u and v.

Fortunately, there is a simple solution to this problem. To prevent the interference of Pointer-Push&Pull operations the first edge of each operation is locked for the use by further operations. This implies that this edge can not be used by multiple Pointer-Push&Pull operations at the same time. The second edge of a Pointer-Push&Pull operation does not need to be locked, since it will be replaced immediately without any delay due to network communication.

If a Pointer-Push&Pull operation choses an edge, which is currently locked, then there are two possible solutions. First, the Pointer-Push&Pull operation can of course be canceled. This is unproblematic since no changes have been made to the network graph, yet. Another option is to wait for the lock to disappear. This is reasonable since an operation will usually need few milliseconds. Furthermore, the number of intersecting operations will be low if the interval in which a node starts Pointer-Push&Pull operations is chosen reasonably.

4. CONCLUSIONS AND OPEN PROBLEMS

In this paper, we have presented a random graph transformation approach for maintaining a stable backbone for peer-to-peer networks: The Pointer-Push&Pull-operation. This simple operation involving only two peers at a time is able to establish and maintain a stable network: Every d-out-regular weakly connected multi-digraph occurs with same probability. Such a random graph is an expander graph a.a.s. for a generalized notion of expansion for multi-digraphs. Therefore small diameter and high connectivity is guaranteed. An interesting feature is that this scheme recovers from any worst case situation. The operation itself exchanges only two messages between the peers. It can be implemented with minimal effort and does not need any global knowledge of the network so that the Pointer-Push&Pull operation can be used in parallel at all peers.

All findings in this paper are proved using Markov theory. In our view this is the only reasonable way as graph transformations like the Pointer-Pull operation perform very well in simulations of large networks, yet eventually disconnect the network as we have proven here. This work improves our results presented in [12], where we presented the 1-Flipper operation for undirected connected regular graphs, in two ways. The Pointer-Push&Pull operation is far simpler than the 1-Flipper and multi-digraphs are easier to maintain than undirected graphs.

The convergence rate of the Pointer-Push&Pull operation is still unknown. In case of the 1-Flipper, there is a recent paper [3] claiming that the convergence rate can be bounded by a high degree polynomial, i.e. roughly $d^{56}n^{53}$. Yet, we strongly believe that this bound is far from the true behavior of the 1-Flipper operation.

There is also another graph transformation which is related to the 1-Flipper operation. Cooper, Dyer, and Greenhill [2] showed polynomial convergence speed for the *Switch* operation on regular graphs. A switch operation chooses two edges $\{i,j\},\{k,l\}$ uniformly at random, then chooses a perfect matching of $\{i,j,k,l\}$ and replaces $\{i,j\},\{k,l\}$ with the edges describing the matching if the resulting graph stays simple. Whereas the 1-Flipper chooses a random path of three edges only and flips the two outer edges if the resulting graph stays simple. Cooper et al. give an upper bound of $d^{17}n^7\log(dn\epsilon^{-1})$ for the convergence rate of the switch operation. However, they guess that $\mathcal{O}(n\log n)$ operations suffice, for constant degree, but this seems beyond the reach of known proof techniques. $\mathcal{O}(n\log n)$ is also what we conjecture for the Pointer-Push&Pull and the 1-Flipper operations based on experiments. Simulations and a real-world software implementation indicate quick convergence to the steady state probability distribution for the Pointer-Push&Pull operation.

In this paper we have only discussed multi-digraphs since it turns out that the Pointer-Push&Pull oepration is unable to maintain simple digraphs. Yet, it is not clear if there exists a similar operation for simple digraphs suitable for peer-to-peer networks.

5. REFERENCES

[1] S. M. Botros and S. R. Waterhouse. Search in jxta and other distributed networks. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing (P2P 2001)*, pages 30–35, 2001.

[2] C. Cooper, M. Dyer, and C. Greenhill. Sampling regular graphs and a peer-to-peer network. In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 980–988, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[3] T. Feder, A. Guetz, M. Mihail, and A. Saberi. The flip markov chain and peer-to-peer networks. http://www.stanford.edu/∼saberi/switch.pdf, 2006.

[4] C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Mar. 2004.

[5] Gnutella. The gnutella protocol specification v0.4.

[6] K. Hildrum, J. D. Kubiatowicz, S. Rao, and B. Y. Zhao. Distribted object location in a dynamic network. In *Proceedings of the fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA-02)*, pages 41–52, New York, Aug. 10–13 2002. ACM Press.

[7] M. A. Jovanovic, F. S. Annexstein, and K. A. Berman. Scalability issues in large peer-to-peer networks — a case study of Gnutella. Technical report, University of Cincinnati, 2001.

[8] M. F. Kaashoek and D. R. Karger. Koorde: A simple degree-optimal distributed hash table. In *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, pages 98–107, 2003.

[9] M. Kim and M. Medard. Robustness in large-scale random networks. In *IEEE INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, Mar. 2004.

[10] M. Kim, M. Medard, and E. A. M. Torres. On reliability of large-scale random networks. http://web.mit.edu/minkyu/www/doc/ToNsubmitted.pdf, 2005.

[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th international conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM Press.

[12] P. Mahlmann and C. Schindelhauer. Peer-to-peer networks based on random transformations of connected regular undirected graphs. In *SPAA'05: Proceedings of the seventeenth annual ACM Symposium on Parallelism in Algorithms and Architectures*, pages 155–164, New York, NY, USA, 2005. ACM Press.

[13] M. Naor and U. Wieder. Novel architectures for p2p applications: the continuous-discrete approach. In *SPAA '03: Proceedings of the fifteenth annual ACM Symposium on Parallel Algorithms and Architectures*, pages 50–59, New York, NY, USA, 2003. ACM Press.

[14] G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter p2p networks. In *FOCS '01: Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, page 492, Washington, DC, USA, 2001. IEEE Computer Society.

[15] M. Ripeanu. Peer-to-peer architecture case study: Gnutella network. In *Proceedings of the 1st International Conference on Peer-to-Peer Computing*

(P2P 2001), pages 99–100, 2001.

[16] J. Ritter. Why gnutella can't scale. http://www.darkridge.com/~jpr5/doc/gnutella.html, 2001.

[17] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In R. Guerin, editor, *Proceedings of the ACM SIGCOMM 2001 Conference (SIGCOMM-01)*, volume 31, 4 of *Computer Communication Review*, pages 149–160, New York, Aug. 27–31 2001. ACM Press.

[18] N. C. Wormald. Models of random regular graphs. In *Surveys in Combinatorics, 1993, Walker (Ed.), London Mathematical Society Lecture Note Series 187, Cambridge University Press*. 1999.

APPENDIX

A. THE POINTER-PUSH AND THE POINTER-PULL OPERATION

The Pointer-Push operation is a simple graph transformation defined on a path of two edges applicable to multi-digraphs.

Definition 8 (Pointer-Push Operation) *Consider a multi-digraph $G = (V, E, \#_E)$ and nodes v_1, v_2, $v_3 \in V$ forming a directed path $P = (v_1, v_2, v_3)$ in G. Then, the Pointer-Push operation $PUSH_P$ transforms graph G to a graph $PUSH_P(G) = (V, E', \#_{E'})$ with*

$$E' = (E \setminus (v_1, v_2)) \cup (v_1, v_3).$$

Figure 3 illustrates the Pointer-Push operation. A randomized version of the Pointer-Push operation is given by Algorithm 3.

Algorithm 3 Random Pointer-Push

Choose random node $v_1 \in V$
$v_2 \leftarrow$ random node $\in N^+(v_1)$
$v_3 \leftarrow$ random node $\in N^+(v_2)$
$E \leftarrow (E \setminus (v_1, v_2)) \cup (v_1, v_3)$

The following Lemma shows that the Pointer-Push operation is sound.

Lemma 7 *Applying random Pointer-Push operations to a connected muldi-digraph G will preserve connectivity of G. Furthermore the outdegree of each node in G will stay the same.*

PROOF. Concerning the outdegree note, that one of the outgoing edges of v_1 is deleted and one new edge starting at v_1 is created. Connectivity is preserved because all participating nodes of a Pointer-Push operation stay connected. However, the resulting graph may be weakly connected. □

For the asymptotic behavior of the random Pointer-Push operation the following theorem holds.

Theorem 5 *A series of random Pointer-Push operations will transform any connected multi-digraph into a connected set of stars in the limit with probability* 1.

PROOF. When applying random Pointer-Push operations there is a non-zero probability that a node creates self edges. Eventually, all outgoing edges of a node will point to itself and therefore this node will be a sink. Every time the second node of a Pointer-Push operation is such a sink $s \in V$, the third node will also be s. Therefore, the random Pointer-Push operation will create edges pointing to the sink what in turn further increases the probability of random Pointer-Push operations to end in a sink. Note, that there is no way to remove edges pointing to a sink. So, in the long run the graph will consist of at least one sink and all other nodes pointing directly to sinks. □

Intuitively, nodes with higher indegree are prone to receive even higher indegree. The above theorem shows, that this causes the Pointer-Push operation to not provide generality.

We can overcome the problem of further increasing the indegree of nodes which already have high indegree with the following graph transformation, called Pointer-Pull operation.

Definition 9 (Pointer-Pull Operation) *Consider a multi-digraph $G = (V, E, \#_E)$ and nodes v_1, v_2, v_3, $v_4 \in V$ forming a directed path $P = (v_1, v_2, v_3, v_4)$ in G. Then, the Pointer-Pull operation $PULL_P$ transforms graph G to a graph $PULL_P(G) = (V, E', \#'_E)$ with*

$$E' = (E \setminus (v_3, v_4)) \cup (v_3, v_1).$$

The Pointer-Pull operation is illustrated in Figure 4 and a randomized version is given by Algorithm 4.

Algorithm 4 Random Pointer-Pull

Choose random node $v_1 \in V$
$v_2 \leftarrow$ random node $\in N^+(v_1)$
$v_3 \leftarrow$ random node $\in N^+(v_2)$
$v_4 \leftarrow$ random node $\in N^+(v_3)$
$E \leftarrow (E \setminus (v_3, v_4)) \cup (v_3, v_1)$

As in case of the Pointer-Push operation, the Pointer-Pull operation does not change the outdegree of any node. The intuition, in contrast to the Pointer-Push operation, is that applying random Pointer-Pull operations may balance the indegree of the nodes. This is because the starting node v_1 of each operation will increase its indegree by one and v_1 is chosen uniformly at random. Furthermore, nodes with high indegree have a higher probability to be endpoint of a Pointer-Pull operation and therefore, higher probability to get their indegree decreased. Even if this is the case, the following theorem shows a major drawback of the Pointer-Pull operation.

Theorem 6 *Starting with an arbitrary multi-digraph G with n nodes, random Pointer-Pull operations disconnect G into n components of single nodes with slopes in the limit.*

PROOF. From every digraph, this terminal graph is reachable by a series of random Pointer-Pull operations. Furthermore, applying random Pointer-Pull operations to the terminal graph will transform the terminal graph to itself. So, in the limit the Markov chain described by the random Pointer-Pull operation converges to this terminal graph. □