Table of Contents

Graph Transformations
  Definitions
  Requirements
  First Examples

Pointer-Push & Pull
  Unlabeled Digraphs
  Edge Labeled Digraphs

Simple Graphs

Peer-to-Peer Networks
  Implementation of Pointer-Push & Pull
  Advantage of Pointer-Push & Pull
  Example: 3nuts
Graph Transformations

Definitions

Definition (Simple Digraph)
A simple digraph $G = (V, E)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E \subseteq \{ (u, v) | u, v \in V, u \neq v \}$.

Definition (Multi-Digraph)
A multi-digraph $G = (V, E, #)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E = \{ (u, v) | u, v \in V \}$ with multiplicities given by $# : E \rightarrow \mathbb{N}_0$. 
Graph Transformations

Definitions

Definition (Simple Digraph)

A simple digraph $G = (V, E)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$. 
Graph Transformations

Definitions

Definition (Simple Digraph)
A simple digraph $G = (V, E)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$.

Definition (Multi-Digraph)
A multi-digraph $G = (V, E, \#)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E = \{(u, v) \mid u, v \in V\}$ with multiplicities given by $\#: E \to \mathbb{N}_0$.

\[\text{Diagram:}\]

\[\text{Nodes: 0, 1, 2, 1, 0}\]

\[\text{Edges: 0 \rightarrow 2 \rightarrow 1, 1 \rightarrow 0 \rightarrow 0}\]
Graph Transformations

Definitions

Definition (Simple Digraph)
A simple digraph $G = (V, E)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$.

Definition (Multi-Digraph)
A multi-digraph $G = (V, E, \#)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E = \{(u, v) \mid u, v \in V\}$ with multiplicities given by $\# : E \to \mathbb{N}_0$. 

\[
\begin{array}{c}
0 \xrightarrow{2} 1 \xleftarrow{2} 0
\end{array}
\sim
\begin{array}{c}
\circlearrowleft
\end{array}
\begin{array}{c}
\circlearrowleft
\end{array}
\]
Graph Transformations
Definitions

Definition (Graph Transformation)
Let $\mathcal{G} \subseteq \{G \mid G$ is a multi-digraph with $n$ nodes$\}$. A graph transformation is a random transition $\tau : \mathcal{G} \rightsquigarrow \mathcal{G}$ such that

$$\forall G \in \mathcal{G} : \sum_{G' \in \mathcal{G}} \Pr \left( \tau(G) = G' \right) = 1.$$
Graph Transformations

Definitions

Definition (Graph Transformation)

Let \( \mathcal{G} \subseteq \{ G \mid G \text{ is a multi-digraph with } n \text{ nodes} \} \). A graph transformation is a random transition \( \tau : \mathcal{G} \rightsquigarrow \mathcal{G} \) such that

\[
\forall G \in \mathcal{G} : \sum_{G' \in \mathcal{G}} \Pr(\tau(G) = G') = 1.
\]

If \( |\mathcal{G}| < \infty \), \( \tau \) defines a Markov chain, where the set of states is \( \mathcal{G} \) and the transition matrix is \( T \in \mathbb{R}^{|\mathcal{G}| \times |\mathcal{G}|} \) with \( t_{G,G'} = \Pr(\tau(G) = G') \).
Graph Transformations

Requirements

Requirements for graph transformations used in peer-to-peer networks:

- **Soundness:** \( \forall G \in G: \tau(G) \in G \)

- **Generality:** \( \forall G, G' \in G: \lim_{k \to \infty} \Pr(\tau^k(G) = G') > 0 \)

- **Uniform generality:** \( \forall G, G' \in G: \lim_{k \to \infty} \Pr(\tau^k(G) = G') = \frac{1}{|G|} \)

- **Feasibility:** \( \tau \) can be described by a simple routine with a straightforward implementation in a distributed maintained network.

- **Convergence rate:** After a small number of transitions a good approximation of the ultimate distribution on \( G \) is achieved.
Graph Transformations
Requirements

Requirements for graph transformations used in peer-to-peer networks:

- **Soundness**: \( \forall G \in \mathcal{G} : \tau(G) \in \mathcal{G} \)
Graph Transformations
Requirements

Requirements for graph transformations used in peer-to-peer networks:

- **Soundness**: \( \forall G \in \mathcal{G} : \tau(G) \in \mathcal{G} \)
- **Generality**: \( \forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') > 0 \)
Graph Transformations
Requirements

Requirements for graph transformations used in peer-to-peer networks:

- **Soundness:** $\forall G \in \mathcal{G} : \tau(G) \in \mathcal{G}$
- **Generality:** $\forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') > 0$
  - Uniform generality: $\forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') = \frac{1}{|\mathcal{G}|}$
- **Feasibility:** $\tau$ can be described by a simple routine with a straightforward implementation in a distributed maintained network.
- **Convergence rate:** After a small number of transitions a good approximation of the ultimate distribution on $G$ is achieved.
Graph Transformations

Requirements

Requirements for graph transformations used in peer-to-peer networks:

- **Soundness**: \( \forall G \in \mathcal{G} : \tau(G) \in \mathcal{G} \)
- **Generality**: \( \forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') > 0 \)
  - Uniform generality: \( \forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') = \frac{1}{|\mathcal{G}|} \)
- **Feasibility**: \( \tau \) can be described by a simple routine with a straightforward implementation in a distributed maintained network.
Graph Transformations

Requirements

Requirements for graph transformations used in peer-to-peer networks:

- **Soundness:** $\forall G \in \mathcal{G} : \tau(G) \in \mathcal{G}$
- **Generality:** $\forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') > 0$
  - Uniform generality: $\forall G, G' \in \mathcal{G} : \lim_{k \to \infty} \Pr(\tau^k(G) = G') = \frac{1}{|\mathcal{G}|}$
- **Feasibility:** $\tau$ can be described by a simple routine with a straightforward implementation in a distributed maintained network.
- **Convergence rate:** After a small number of transitions a good approximation of the ultimate distribution on $\mathcal{G}$ is achieved.
Graph Transformations

First Examples
Graph Transformations
First Examples

Let \( G_u := \left\{ G \mid \begin{array}{l} G \text{ is a weakly-connected } d\text{-out-regular} \\ \text{multi-digraph with } n \text{ nodes} \end{array} \right\} \).

\( G = (V, E, \#) \) is \( d\)-out-regular \( \iff \forall u \in V : \sum_{v \in V} \#((u, v)) = d \)
Graph Transformations

First Examples

Let \( G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}. \)

\( G = (V, E, \#) \text{ is } d\text{-out-regular } \iff \forall u \in V : \sum_{v \in V} \#((u, v)) = d \)

Pointer-Push:
Graph Transformations

First Examples

Let $G_u := \left\{ G \left| \begin{array}{l} \text{G is a weakly-connected d-out-regular} \\ \text{multi-digraph with n nodes} \end{array} \right. \right\}$. 

$G = (V, E, \#)$ is $d$-out-regular $\iff \forall u \in V : \sum_{v \in V} \#((u, v)) = d$

**Pointer-Push:**

- is sound
- is feasible
- is not general
Graph Transformations
First Examples

Let $G_u := \left\{ G \mid \begin{array}{l} G \text{ is a weakly-connected } d\text{-out-regular} \\ \text{multi-digraph with } n \text{ nodes} \end{array} \right\}$. 

$G = (V, E, \#)$ is $d$-out-regular $\iff \forall u \in V : \sum_{v \in V} \#((u, v)) = d$

**Pointer-Pull:**

![Diagram of Graph Transformations]
Graph Transformations

First Examples

Let $G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$. $G = (V, E, #)$ is $d$-out-regular $\iff \forall u \in V : \sum_{v \in V} #((u, v)) = d$

Pointer-Pull:

\[\text{is not sound}\]
Pointer-Push&Pull

Unlabeled Digraphs

Let $G_u := \left\{ G \mid \begin{array}{l} G \text{ is a weakly-connected } d\text{-out-regular} \\ \text{multi-digraph with } n \text{ nodes} \end{array} \right\}$. 
**Pointer-Push&Pull**

Unlabeled Digraphs

Let \( \mathcal{G}_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}. \)

**Pointer-Push&Pull:**

![Diagram of a weakly-connected d-out-regular multi-digraph]
Unlabeled Digraphs
Let $\mathcal{G}_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

**Pointer-Push&Pull:**

![Diagram of Pointer-Push&Pull]
Pointer-Push&Pull

Unlabeled Digraphs
Let $G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

Pointer-Push&Pull:
Let $G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

**Pointer-Push&Pull:**

- Is sound
- Is feasible
- Is general
- Is uniform general
Pointer-Push&Pull
Unlabeled Digraphs

Let $\mathcal{G}_u := \{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

Pointer-Push&Pull:

- is sound
- is feasible
- is general
- is uniform general
Pointer-Push&Pull

Unlabeled Digraphs

Let $G = (V, E, \#)$, $u \in V$. $N^+(u) := \{ v \in V \mid \#((u, v)) > 0 \}$. 
Pointer-Push&Pull

Unlabeled Digraphs

Let $G = (V, E, \#)$, $u \in V$. $N^+(u) := \{v \in V \mid \#((u, v)) > 0\}$.

---

**Algorithm 2** Unlabeled Pointer-Push&Pull: $\tau_u : G_u \rightsquigarrow G_u$

1. $v_1 \leftarrow^{R} V$
2. **if** random event with probability $\frac{|N^+(v_1)|}{d}$ occurs **then**
3. $v_2 \leftarrow^{R} N^+(v_1)$
4. **if** random event with probability $\frac{|N^+(v_2)|}{d}$ occurs **then**
5. $v_3 \leftarrow^{R} N^+(v_2)$
6. $\#((v_1, v_2)) := \#((v_1, v_2)) - 1$
7. $\#((v_2, v_3)) := \#((v_2, v_3)) - 1$
8. $\#((v_2, v_1)) := \#((v_2, v_1)) + 1$
9. $\#((v_1, v_3)) := \#((v_1, v_3)) + 1$
Pointer-Push&Pull
Unlabeled Digraphs

Let \( G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\} \).
Let $G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

Lemma

$\forall G, G' \in G_u : \Pr (\tau_u(G) = G') = \Pr (\tau_u(G') = G)$. 
Pointer-Push&Pull
Unlabeled Digraphs

Let $G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

Lemma
$\forall G, G' \in G_u : \Pr(\tau_u(G) = G') = \Pr(\tau_u(G') = G)$.

Proof.
Suppose $G'$ is reached from $G$ using the path $(v_i, v_j, v_k)$.
Pointer-Push&Pull
Unlabeled Digraphs

Let \( \mathcal{G}_u := \left\{ G \middle| G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\} \).

Lemma
\( \forall G, G' \in \mathcal{G}_u : \Pr(\tau_u(G) = G') = \Pr(\tau_u(G') = G) \).

Proof.
Suppose \( G' \) is reached from \( G \) using the path \((v_i, v_j, v_k)\).
\( \Rightarrow \) \( G \) can be reached from \( G' \) exactly with \((v_j, v_i, v_k)\).
Let $G_u := \{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

**Lemma**

$\forall G, G' \in G_u : \Pr(\tau_u(G) = G') = \Pr(\tau_u(G') = G)$.

**Proof.**

Suppose $G'$ is reached from $G$ using the path $(v_i, v_j, v_k)$.

$\Rightarrow G$ can be reached from $G'$ exactly with $(v_j, v_i, v_k)$.

$\Rightarrow \Pr(\tau_u(G) = G') = \frac{1}{n} \cdot \frac{1}{d} \cdot \frac{1}{d} = \Pr(\tau_u(G') = G)$
Pointer-Push&Pull

Unlabeled Digraphs

Let \( \mathcal{G}_u \) := \( \{ G \mid G \) is a weakly-connected \( d \)-out-regular multi-digraph with \( n \) nodes \}.

Lemma

Let \( G, G' \in \mathcal{G}_u \). \( G' \) can be reached from \( G \) with at most \( 10nd \) Pointer-Push&Pull operations.
Pointer-Push&Pull

Unlabeled Digraphs
Let \( \mathcal{G}_u := \left\{ G \mid \text{G is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\} \).

Lemma
Let \( G, G' \in \mathcal{G}_u \). \( G' \) can be reached from \( G \) with at most 10nd Pointer-Push&Pull operations.

Proof.
Let \( G = (V, E, \#) \) with \( V = \{v_1, \ldots, v_n\} \).
Let $G_u := \{ G \mid G$ is a weakly-connected $d$-out-regular multi-digraph with $n$ nodes $\}$.

**Lemma**

Let $G, G' \in G_u$. $G'$ can be reached from $G$ with at most 10nd Pointer-Push&Pull operations.

**Proof.**

Let $G = (V, E, \#)$ with $V = \{v_1, \ldots, v_n\}$.

Define $G_c := (V, E, \#_c)$ with:

$$\forall u \in V : \#_c ((u, v_1)) = d,$$

$$\forall u \in V, v \in V \setminus \{v_1\} : \#_c ((u, v)) = 0.$$
Pointer-Push&Pull

Unlabeled Digraphs
Let \( \mathcal{G}_u := \left\{ G \left| G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right. \right\}. \)

**Lemma**
Let \( G, G' \in \mathcal{G}_u. \) \( G' \) can be reached from \( G \) with at most 10nd Pointer-Push&Pull operations.

**Proof.**
Let \( G = (V, E, \#) \) with \( V = \{v_1, \ldots, v_n\}. \)
Define \( G_c := (V, E, \#_c) \) with:
\[
\forall u \in V : \#_c ((u, v_1)) = d, \\
\forall u \in V, v \in V \setminus \{v_1\} : \#_c ((u, v)) = 0.
\]

\( \Rightarrow \) To show: \( G_c \) can be reached from \( G \) with at most 5nd Pointer-Push&Pull operations.
Pointer-Push&Pull
Unlabeled Digraphs

Case 1: \( \exists j \in \{2, \ldots, n\} : \#((v_1, v_j)) > 0 \)
Pointer-Push&Pull
Unlabeled Digraphs

Case 1: \( \exists j \in \{2, \ldots, n\} : \#((v_1, v_j)) > 0 \)

Case 1.1: \( j \neq k \neq 1 \)
Pointer-Push&Pull
Unlabeled Digraphs

**Case 1:** \( \exists j \in \{2, \ldots, n\} : \#((v_1, v_j)) > 0 \)

**Case 1.1:** \( j \neq k \neq 1 \)

**Case 1.2:**
Pointer-Push&Pull
Unlabeled Digraphs

Case 1: \( \exists j \in \{2, \ldots, n\} : \#((v_1, v_j)) > 0 \)

Case 1.1: \( j \neq k \neq 1 \)

Case 1.2:

Case 1.3:
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: 
\[ \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \]

Case 2.1:
\[ 1 \neq j \neq k \neq 1 \]
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)

Case 2.1:
\( 1 \neq j \neq k \neq 1 \)
Case 2: $\#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0$

Case 2.1: $1 \neq j \neq k \neq 1$

Case 2.2: $1 \neq j \neq k \neq 1 \neq l$

Case 2.3: $1 \neq j \neq k \neq 1$
Pointer-Push&Pull

Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)

Case 2.1:
1 \( \neq j \neq k \neq 1 \)
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)

Case 2.1:
\(1 \neq j \neq k \neq 1\)
Pointer-Push&Pull

Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)

Case 2.1: 
\( 1 \neq j \neq k \neq 1 \)
Pointer-Push&Pull

Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)

Case 2.1: 
1 \( \neq j \neq k \neq 1 \)

Case 2.2: 
1 \( \neq j \neq k \neq 1 \neq l \neq j \)
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\}: \#((v_1, v_j)) = 0 \)

Case 2.1: \( 1 \neq j \neq k \neq 1 \)

Case 2.2: \( 1 \neq j \neq k \neq 1 \neq l \neq j \)
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: $\#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0$

Case 2.1: $1 \neq j \neq k \neq 1$

Case 2.2: $1 \neq j \neq k \neq 1 \neq l \neq j$
Pointer-Push&Pull
Unlabeled Digraphs

Case 2: $\#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\}: \#((v_1, v_j)) = 0$

Case 2.1: $1 \neq j \neq k \neq 1$

Case 2.2: $1 \neq j \neq k \neq 1 \neq l \neq j$
Pointer-Push&Pull

Unlabeled Digraphs

Case 2: \( \#( (v_1, v_1) ) = d, \forall j \in \{2, \ldots, n\} : \#( (v_1, v_j) ) = 0 \)

Case 2.1: \( 1 \neq j \neq k \neq 1 \)

Case 2.2: \( 1 \neq j \neq k \neq 1 \neq l \neq j \)

Case 2.3: \( 1 \neq j \)
Pointer-Push&Pull

Unlabeled Digraphs

**Case 2:** \( \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \)

**Case 2.1:**
1 \( \not\equiv j \not\equiv k \not\equiv 1 \)

**Case 2.2:**
1 \( \not\equiv j \not\equiv k \not\equiv 1 \not\equiv l \not\equiv j \)

**Case 2.3:**
1 \( \not\equiv j \)


**Pointer-Push&Pull**

*Unlabeled Digraphs*

**Case 2:**

$$\#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0$$

**Case 2.1:**

$$1 \neq j \neq k \neq 1$$

**Case 2.2:**

$$1 \neq j \neq k \neq 1 \neq l \neq j$$

**Case 2.3:**

$$1 \neq j$$
**Pointer-Push&Pull**

Unlabeled Digraphs

**Case 2:** $\#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0$

**Case 2.1:** $1 \neq j \neq k \neq 1$

**Case 2.2:** $1 \neq j \neq k \neq 1 \neq l \neq j$

**Case 2.3:** $1 \neq j$
Pointer-Push&Pull

Unlabeled Digraphs

**Case 2:** 
\[ \#((v_1, v_1)) = d, \forall j \in \{2, \ldots, n\} : \#((v_1, v_j)) = 0 \]

**Case 2.1:** 
1 \( \neq j \neq k \neq 1 \)

**Case 2.2:** 
1 \( \neq j \neq k \neq 1 \neq l \neq j \)

**Case 2.3:** 
1 \( \neq j \)
Pointer-Push&Pull

Unlabeled Digraphs
Let $G_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

Theorem
$\forall G, G' \in G_u : \lim_{k \to \infty} \Pr(\tau^k_u(G) = G') = \frac{1}{|G_u|}$
Pointer-Push&Pull

Unlabeled Digraphs
Let $\mathcal{G}_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

Theorem
$\forall G, G' \in \mathcal{G}_u : \lim_{k \to \infty} \Pr(\tau^k_u(G) = G') = \frac{1}{|\mathcal{G}_u|}$

Proof.
Follows from properties of corresponding Markov chain with transition matrix $T$: 

1. $T$ is symmetric $\Rightarrow (\frac{1}{|\mathcal{G}_u|}, ..., \frac{1}{|\mathcal{G}_u|})^T = (\frac{1}{|\mathcal{G}_u|}, ..., \frac{1}{|\mathcal{G}_u|})$ is stationary distribution
2. Markov chain is irreducible
3. $T$ has some non-zero diagonal entries $\Rightarrow$ Markov chain is aperiodic
Pointer-Push&Pull

Unlabeled Digraphs

Let \( \mathcal{G}_u := \left\{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\} \).

**Theorem**

\[ \forall G, G' \in \mathcal{G}_u : \lim_{k \to \infty} \Pr(\tau^k_u(G) = G') = \frac{1}{|\mathcal{G}_u|} \]

**Proof.**

Follows from properties of corresponding Markov chain with transition matrix \( T \):

- \( T \) is symmetric
  \[ \Rightarrow \left( \frac{1}{|\mathcal{G}_u|}, \ldots, \frac{1}{|\mathcal{G}_u|} \right) T = \left( \frac{1}{|\mathcal{G}_u|}, \ldots, \frac{1}{|\mathcal{G}_u|} \right) \text{ is stationary distribution} \]
Pointer-Push&Pull

Unlabeled Digraphs

Let $\mathcal{G}_u := \left\{ G \left| G\text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n\text{ nodes} \right. \right\}$.

Theorem

$\forall G, G' \in \mathcal{G}_u : \lim_{k \to \infty} \Pr(\tau^k_u(G) = G') = \frac{1}{|\mathcal{G}_u|}$

Proof.

Follows from properties of corresponding Markov chain with transition matrix $T$:

- $T$ is symmetric
  $\Rightarrow \left( \frac{1}{|\mathcal{G}_u|}, \ldots, \frac{1}{|\mathcal{G}_u|} \right) T = \left( \frac{1}{|\mathcal{G}_u|}, \ldots, \frac{1}{|\mathcal{G}_u|} \right)$ is stationary distribution

- Markov chain is irreducible
Pointer-Push&Pull
Unlabeled Digraphs
Let $G_u := \{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

Theorem
$\forall G, G' \in G_u : \lim_{k \to \infty} \Pr(\tau_u^k(G) = G') = \frac{1}{|G_u|}$

Proof.
Follows from properties of corresponding Markov chain with transition matrix $T$:

- $T$ is symmetric
  \[
  \Rightarrow \left( \frac{1}{|G_u|}, \ldots, \frac{1}{|G_u|} \right) T = \left( \frac{1}{|G_u|}, \ldots, \frac{1}{|G_u|} \right) \text{ is stationary distribution}
  \]
- Markov chain is irreducible
- $T$ has some non-zero diagonal entries
  \Rightarrow \text{Markov chain is aperiodic}
Definition (Edge Labeled Multi-Digraph)

An edge labeled $d$-out-regular multi-digraph $G = (V, E)$ is defined by a node set $V = \{v_1, ..., v_n\}$ and a set of directed edges $E \subseteq \{(u, v, i) | u, v \in V, i \in \{1, ..., d\}\}$ with:

$$\forall u \in V \forall i \in \{1, ..., d\} \exists! N^+(u, i) \in V : (u, N^+(u, i), i) \in E.$$
Pointer-Push & Pull

Edge Labeled Digraphs

**Definition (Edge Labeled Multi-Digraph)**

An edge labeled $d$-out-regular multi-digraph $G = (V, E)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E \subseteq \{(u, v, i) \mid u, v \in V, i \in \{1, \ldots, d\}\}$ with:

\[
\forall u \in V \forall i \in \{1, \ldots, d\} \exists ! N^+(u, i) \in V : (u, N^+(u, i), i) \in E.
\]
Pointer-Push&Pull

Edge Labeled Digraphs

Definition (Edge Labeled Multi-Digraph)

An edge labeled $d$-out-regular multi-digraph $G = (V, E)$ is defined by a node set $V = \{v_1, \ldots, v_n\}$ and a set of directed edges $E \subseteq \{(u, v, i) \mid u, v \in V, i \in \{1, \ldots, d\}\}$ with:

$$\forall u \in V \forall i \in \{1, \ldots, d\} \exists! N^+(u, i) \in V : (u, N^+(u, i), i) \in E.$$
Pointer-Push&Pull
Edge Labeled Digraphs

Definition (Equivalence Class)
Let $G$ be an unlabeled $d$-out-regular multi-digraph. $[G]$ denotes the set of all edge labeled $d$-out-regular multi-digraphs describing $G$ when omitting the edge labels.
Pointer-Push&Pull

Edge Labeled Digraphs

Definition (Equivalence Class)

Let $G$ be an unlabeled $d$-out-regular multi-digraph. $[G]$ denotes the set of all edge labeled $d$-out-regular multi-digraphs describing $G$ when omitting the edge labels.

$G \Rightarrow |G|$. The lower the number of multi-edges in $G$, the larger is $|G|$. 

![Diagram of $G$ and $[G]$](image)
Definition (Equivalence Class)

Let $G$ be an unlabeled $d$-out-regular multi-digraph. $[G]$ denotes the set of all edge labeled $d$-out-regular multi-digraphs describing $G$ when omitting the edge labels.

$\Rightarrow$ The lower the number of multi-edges in $G$, the larger is $| [G] |$. 
Pointer-Push&Pull
Edge Labeled Digraphs

Let $\mathcal{G}_l := \left\{ G \mid G \text{ is an edge labeled weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$. 
Let $G_l := \left\{ G \mid G \text{ is an edge labeled weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \right\}$.

**Algorithm 4** Labeled Pointer-Push&Pull: $\tau_l : G_l \leadsto G_l$

1: $v_1 \leftarrow^R V$
2: $i \leftarrow^R \{1, \ldots, d\}$
3: $v_2 := N^+(v_1, i)$
4: $j \leftarrow^R \{1, \ldots, d\}$
5: $v_3 := N^+(v_2, j)$
6: $E := (E \setminus \{(v_1, v_2, i), (v_2, v_3, j)\}) \cup \{(v_2, v_1, j), (v_1, v_3, i)\}$
Pointer-Push&Pull

Edge Labeled Digraphs

Let \( G_l := \{ G | G \) is an edge labeled weakly-connected \( d \)-out-regular multi-digraph with \( n \) nodes \}.

As before: \( \tau_l \) is

- sound
- feasible
- general
- uniform general
Pointer-Push&Pull

Edge Labeled Digraphs

Let $G_l := \{ G \mid G \text{ is an edge labeled weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

As before: $\tau_l$ is

- sound
- feasible
- general
- uniform general

Let $G_u := \{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

Theorem

$\forall G, G' \in G_u : \lim_{k \to \infty} \Pr(\tau_l^k(G) = G') = \frac{|G'|}{|G_l|}$.

A particular simple digraph is more probable than a particular multi-digraph.
Pointer-Push&Pull

Edge Labeled Digraphs

Let $\mathcal{G}_l := \{ G \mid G \text{ is an edge labeled weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

As before: $\tau_l$ is

- sound
- feasible
- general
- uniform general

Let $\mathcal{G}_u := \{ G \mid G \text{ is a weakly-connected } d\text{-out-regular multi-digraph with } n \text{ nodes} \}$.

**Theorem**

$\forall G, G' \in \mathcal{G}_u : \lim_{k \to \infty} \Pr(\tau_l^k(G) = G') = \frac{|[G']|}{|\mathcal{G}_l|}$.

$\Rightarrow$ A particular simple digraph is more probable than a particular multi-digraph.
Simple Graphs
Simple Graphs

Pointer-Push\&Pull cannot be restricted to simple graphs:
Simple Graphs

Let $G_{s} \triangleq \{ G \mid G \text{ is an undirected connected } d\text{-regular simple graph with } n \text{ nodes} \}$. 

1-Flipper:
Simple Graphs

Let $G_s := \left\{ G \mid G \text{ is an undirected connected } d\text{-regular simple graph with } n \text{ nodes} \right\}$.

1-Flipper:
Simple Graphs

Let $\mathcal{G}_s := \left\{ G \mid G \text{ is an undirected connected } d\text{-regular simple graph with } n \text{ nodes} \right\}$.

1-Flipper:

- is sound
- is feasible
- is general
- is uniform general
Simple Graphs

Let \( G_s := \left\{ G \mid G \text{ is an undirected connected } d\text{-regular simple graph with } n \text{ nodes} \right\} \).

1-Flipper:

- is sound
- is feasible
- is general
- is uniform general

- Four peers have to participate actively
- Digraphs are sufficient in practice
Peer-to-Peer Networks
Implementation of Pointer-Push&Pull

1. $v_1$ requests a random neighbor from $v_2$
2. $v_2$ replaces $v_3$ by $v_1$ in neighborhood list and sends ID of $v_3$ to $v_1$
3. $v_1$ receives ID of $v_3$ from $v_2$ and replaces $v_2$ by $v_3$ in neighborhood list

$\Rightarrow$ only two network operations
$\Rightarrow$ no additional overhead to periodical neighborhood verification
Peer-to-Peer Networks
Implementation of Pointer-Push&Pull

1. $v_1$ requests a random neighbor from $v_2$
2. $v_2$ replaces $v_3$ by $v_1$ in neighborhood list and sends ID of $v_3$ to $v_1$
3. $v_1$ receives ID of $v_3$ from $v_2$ and replaces $v_2$ by $v_3$ in neighborhood list
Peer-to-Peer Networks
Implementation of Pointer-Push&Pull

1. $v_1$ requests a random neighbor from $v_2$
2. $v_2$ replaces $v_3$ by $v_1$ in neighborhood list and sends ID of $v_3$ to $v_1$
3. $v_1$ receives ID of $v_3$ from $v_2$ and replaces $v_2$ by $v_3$ in neighborhood list

⇒ only two network operations
⇒ no additional overhead to periodical neighborhood verification
Peer-to-Peer Networks

Advantage of Pointer-Push & Pull

- Constant and small out-degree
- Logarithmic diameter
- High connectivity

Open Problems:
- Convergence rate: $O(n \log n)$ supposed
- Simulations indicate quick convergence
- Similar operation for simple digraphs
Peer-to-Peer Networks
Advantage of Pointer-Push&Pull

Pointer-Push&Pull leads (with high probability) to random graphs with

- constant and small out-degree
- logarithmic diameter
- high connectivity
Peer-to-Peer Networks

Advantage of Pointer-Push&Pull

Pointer-Push&Pull leads (with high probability) to random graphs with

- constant and small out-degree
- logarithmic diameter
- high connectivity

Open Problems:

- Convergence rate
  - $O(n \log n)$ supposed
  - Simulations indicate quick convergence
- Similar operation for simple digraphs
Peer-to-Peer Networks

Example: 3nuts
Peer-to-Peer Networks

Example: 3nuts

Data tree:
Prefix tree of data identities
Peer-to-Peer Networks

Example: 3nuts

Network tree:
Peer-to-Peer Networks

Example: 3nuts

Network tree:
Peer-to-Peer Networks

Example: 3nuts

Network tree:
Peer-to-Peer Networks

Example: 3nuts

Network tree:
Peer-to-Peer Networks

Example: 3nuts

Network tree:
Peer-to-Peer Networks

Example: 3nuts

Network tree:
Peer-to-Peer Networks

Example: 3nuts

Network tree:

For each random network a peer has to save:
- random neighbors
Peer-to-Peer Networks

Example: 3nuts

Network tree:

For each random network a peer has to save:

- random neighbors
- branch links to each child
  - random branch links
  - local branch links
Peer-to-Peer Networks

Example: 3nuts

Network tree:

For each random network a peer has to save:

- random neighbors
- branch links to each child
  - random branch links
  - local branch links
- responsible peers
Peer-to-Peer Networks
Example: 3nuts

Role of Pointer-Push&Pull:
Peer-to-Peer Networks

Example: 3nuts

Role of Pointer-Push&Pull:
- maintain truly random networks
  ⇒ robustness
Peer-to-Peer Networks
Example: 3nuts

Role of Pointer-Push&Pull:

- maintain truly random networks
  ⇒ robustness
- spread information among peers, e.g. tree structure, weights
Peer-to-Peer Networks
Example: 3nuts

Role of Pointer-Push&Pull:
- maintain truly random networks
  ⇒ robustness
- spread information among peers, e.g. tree structure, weights
- update random branch links and guarantee them to be truly random
Peer-to-Peer Networks
Example: 3nuts

Role of Pointer-Push&Pull:
■ maintain truly random networks
  ⇒ robustness
■ spread information among peers, e.g. tree structure, weights
■ update random branch links and guarantee them to be truly random
■ measure round trip times and find good local branch links
Peer-to-Peer Networks

Example: 3nuts

Routing:
Peer-to-Peer Networks
Example: 3nuts

Routing:
Peer-to-Peer Networks

Example: 3nuts

Routing:
Peer-to-Peer Networks

Example: 3nuts

Routing:
Peer-to-Peer Networks

Example: 3nuts

Routing:

- Use random branch links
  \[ \Rightarrow \text{Number of hops in 3nuts with } n \text{ peers is in } O(\log n) \text{ with high probability} \]
- Use local branch links
  \[ \Rightarrow \text{Experimental evaluation shows that this can benefit routing} \]