On the Efficiency-Optimal Markov Chains for Distributed Networking Applications

Chul-Ho Lee Do Young Eun

***Abstract*—The Metropolis-Hastings (MH) algorithm, in addition to its application for Markov Chain Monte Carlo sampling or simulation, has been popularly used for constructing a random walk that achieves a given, desired stationary distribution over a graph. Applications include crawling-based sampling of large graphs or online social networks, statistical estimation or inference from massive scale of networked data, efficient searching algorithms in unstructured peer-to-peer networks, randomized routing and movement strategies in wireless sensor networks, to list a few. Despite its versatility, the MH algorithm often causes self-transitions of its resulting random walk at some nodes, which is not efficient in the sense of the Peskun ordering – a partial order between off-diagonal elements of transition matrices of two different Markov chains, and in turn results in deficient performance in terms of asymptotic variance of time averages and expected hitting times with slower speed of convergence. To alleviate this problem, we present simple yet effective distributed algorithms that are guaranteed to improve the MH algorithm over time when running on a graph, and eventually reach 'efficiency-optimality', while ensuring the same desired stationary distribution throughout.**

I. Introduction

Motivation: The Metropolis-Hastings (MH) algorithm is the most celebrated Markov Chain Monte Carlo (MCMC) method for sampling from a given probability distribution and has been the *de facto* technique for many networking applications whenever a need arises to construct a Markovian random walk *on a graph* achieving a desired stationary distribution. For example, the MH algorithm has been used as a common building block in the unbiased graph sampling [1]–[3] where the target distribution is uniform over the graph, in the information dissemination for the distributed implementation of fountain codes (or network storage application) [4], [5], in decentralized algorithms for solving a class of optimization problems over sensor nodes [6], in peer-to-peer networking applications such as randomized searching, non-uniform membership management, network topology construction [7], [8], etc. While the widespread popularity of the MH algorithm comes mainly from its lightweight, distributed operation, it typically incurs self-transitions at some nodes, which in turn prevents the 'Metropolized' random walk from exploring the graph efficiently and quickly. It leads to poor graph sampling accuracy and also gives rise to longer delay of the walk until to reach destination(s) for generic networking applications.

Chul-Ho Lee is with DMC R&D Center, Samsung Electronics, Korea, and Do Young Eun is with Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC, USA. Email: {clee4, dyeun}@ncsu.edu. This work was done while the first author was with North Carolina State University. This work was supported in part by National Science Foundation under Grant No. CNS-1217341.

Related Work: There is a body of research works about speeding up a random walk, or a Markov chain, on a graph in terms of its mixing time, hitting time, and/or cover time. First, speeding up a *simple* random walk has been done using local neighborhood exploration [9] or using previous local history to avoid backtracking [10] up to the self-avoiding walk [11] as an extreme case, but its stationary distribution π becomes unknown or out of control. There have also been a few heuristic algorithms on how to construct a Markov chain for graph sampling applications [12], [13], but with no theoretical framework. More importantly, they are valid only when the target stationary distribution π is uniform, thus not applicable to other applications where non-uniform π is desired for better performance [7], [8], [14], [15].

For an arbitrary target stationary distribution π, it is possible to find the fastest mixing (reversible) Markov chain (FMMC) on a graph [16], but only by 'globally' solving a semi-definite programming (SDP) with a complete knowledge of the entire graph, making it virtually impossible or computationally prohibitive even for moderately-sized graphs. While a distributed sub-gradient algorithm for such a SDP has been proposed under a randomized gossip setting [17], it still requires to compute eigenvalues and eigenvectors of the chains, which is a highly non-trivial task. [18], [19] have shown that certain non-reversible Markov chains or lifted Markov chains mix substantially faster than their reversible counterparts, and this concept has been applied to design average consensus algorithms [20], [21]. However, it is still unknown how to construct such a non-reversible chain or lifted chain in a distributed or decentralized fashion for a general graph. The only exception, to the best of our knowledge, is the work in [3] that shows how to convert a given reversible chain with π into a non-reversible one with the same π on a general graph for sampling applications. Nonetheless, there is still a room for further improvement *within* a class of reversible chains, which will be orthogonal and complementary to the algorithm in [3].

Contributions: Thus motivated, this paper seeks to address how to improve the Metropolized random walk, or construct more 'efficient' random walk achieving the same π, in a localized and distributed manner, while maintaining its reversibility. For this purpose, we adopt a notion of the Peskun ordering [22] – a partial order between off-diagonal elements of two different transition matrices, which enables to compare the Metropolized random walk and other random walks, or to rank the *efficiency* of such 'competing' (reversible) Markov chains, in terms of asymptotic variance of time averages and

expected hitting times. Then, we propose simple yet effective distributed algorithms that are guaranteed to keep improving every step of the way in the sense of Peskun ordering, eventually reaching the 'Peskun-optimality' or 'efficiency-optimality', from which no further improvement can be made within a class of reversible chains with the same $\boldsymbol{\pi}$. We also show such an improvement over the plain MH algorithm remains intact even when the desired stationary distribution changes from $\boldsymbol{\pi}$ to $\boldsymbol{\pi}'$ over time, typically is the case in any dynamic environment. We provide numerical results using various graph settings to demonstrate that our distributed algorithms are indeed quick, leading to the optimal class within a few steps. Our distributed algorithms outperform the FMMC for a number of key performance metrics such as the mean hitting time for delay and the asymptotic variance for sampling accuracy, thus offering computationally viable tune-up of a given reversible Markov chain on larger graphs for which existing optimization-based approaches (e.g., FMMC via SDP) often fail or become inapplicable.

II. Preliminaries

We review the MH algorithm and its application in constructing a random walk on a graph to achieve a given, desired stationary distribution. Consider an irreducible (discrete-time) Markov chain $\{X_t \in \mathcal{S}\}_{t\geq 0}$ with a finite state space $\mathcal{S}$ and its transition matrix $\mathbf{P}=\{P_{ij}\}_{i,j\in\mathcal{S}}$ where $P_{ij}=\mathbb{P}\{X_{t+1}=j|X_t=i\}$. Let $\boldsymbol{\pi}=\{\pi_i\}_{i\in\mathcal{S}}$ be a probability distribution with $\pi_i>0$ for all i. The MH algorithm [23], [24] was proposed to obtain a transition matrix $\mathbf{P}$ with $\boldsymbol{\pi}$ as its unique stationary distribution while satisfying the reversibility condition, i.e., $\pi_i P_{ij}=\pi_j P_{ji}$ for all $i,j\in\mathcal{S}$.

The MH algorithm is defined as follows. At the current state i of X_t, the next state X_{t+1} is proposed with *proposal* probability q_{ij} – the state transition probability of an arbitrary irreducible Markov chain on the same state space, where $q_{ij}>0$ if and only if $q_{ji}>0$. We call $\mathbf{Q}=\{q_{ij}\}_{i,j\in\mathcal{S}}$ a proposal matrix. The proposed state transition to $X_{t+1}=j$ is accepted with probability $\alpha_{ij}=\min\left\{1,\frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\}$, and rejected with probability $1-\alpha_{ij}$ in which case $X_{t+1}=i$. Thus, the transition probability P_{ij} is given by

$$P_{ij}=q_{ij}\alpha_{ij}=\min\{q_{ij},q_{ji}\pi_j/\pi_i\} \quad (i\neq j), \tag{1}$$

with $P_{ii}=1-\sum_{j\neq i}P_{ij}$, which ensures that $\mathbf{P}$ is reversible with respect to $\boldsymbol{\pi}$, and the uniqueness of $\boldsymbol{\pi}$ is granted due to the finite state space and irreducibility.

Consider a connected, undirected, non-bipartite graph $G=(\mathcal{N},\mathcal{E})$ with a set of nodes $\mathcal{N}=\{1,2,\ldots,n\}$ and an edge set $\mathcal{E}$. The graph G is assumed to have no multiple edges and no self-loops. Let $N(i)=\{j\in\mathcal{N}:(i,j)\in\mathcal{E}\}$ be a set of neighbors of node i with degree $d_i=|N(i)|$. The maximum degree of the graph is $d_{\max}=\max_{i\in\mathcal{N}}d_i$. In constructing a random walk (or a finite irreducible Markov chain) on G, each edge $(i,j)\in\mathcal{E}$ is associated with some positive transition probability $P_{ij}>0$ with which the random walk makes a transition from node i to node j, and we allow the random walk to make a self-transition on each node (although G has no self-loops). Clearly, $P_{ij}=0$ if $(i,j)\notin\mathcal{E}$ and $i\neq j$.

One popular version of the MH algorithm (e.g., [4], [5], [16]) for constructing a random walk on G with a desired $\boldsymbol{\pi}$ is to use the following proposal probabilities: $q_{ij}^{\mathsf{max}}=1/d_{\max}$ if $(i,j)\in\mathcal{E}$ and $q_{ij}^{\mathsf{max}}=0$ if $(i,j)\notin\mathcal{E}$ and $i\neq j$, with $q_{ii}^{\mathsf{max}}=1-d_i/d_{\max}$. Then, the transition probabilities of a random walk on G by the MH algorithm are given by

$$P_{ij}^{\mathsf{mh\text{-}max}}=\begin{cases}\frac{1}{d_{\max}}\min\left\{1,\frac{\pi_j}{\pi_i}\right\} & \text{if } (i,j)\in\mathcal{E},\\ 0 & \text{if } (i,j)\notin\mathcal{E}, i\neq j,\end{cases} \tag{2}$$

with $P_{ii}^{\mathsf{mh\text{-}max}}=1-\sum_{j\neq i}P_{ij}^{\mathsf{mh\text{-}max}}$. Another popular version of the MH algorithm (e.g., [1]–[3], [5]) is to use the transition probabilities of a simple random walk as the proposal probabilities, i.e., $q_{ij}^{\mathsf{srw}}=1/d_i$ if $(i,j)\in\mathcal{E}$ and $q_{ij}^{\mathsf{srw}}=0$, otherwise. The resulting transition probabilities on G become

$$P_{ij}^{\mathsf{mh\text{-}srw}}=\begin{cases}\min\left\{\frac{1}{d_i},\frac{1}{d_j}\frac{\pi_j}{\pi_i}\right\} & \text{if } (i,j)\in\mathcal{E},\\ 0 & \text{if } (i,j)\notin\mathcal{E}, i\neq j,\end{cases} \tag{3}$$

with $P_{ii}^{\mathsf{mh\text{-}srw}}=1-\sum_{j\neq i}P_{ij}^{\mathsf{mh\text{-}srw}}$. From (2)–(3), we know that $\mathbf{P}^{\mathsf{mh\text{-}max}}=\{P_{ij}^{\mathsf{mh\text{-}max}}\}$ and $\mathbf{P}^{\mathsf{mh\text{-}srw}}=\{P_{ij}^{\mathsf{mh\text{-}srw}}\}$ are both reversible with respect to $\boldsymbol{\pi}$, while $P_{ij}^{\mathsf{mh\text{-}max}}\leq P_{ij}^{\mathsf{mh\text{-}srw}}$ for all i,j $(i\neq j)$ and thus $P_{ii}^{\mathsf{mh\text{-}max}}\geq P_{ii}^{\mathsf{mh\text{-}srw}}$ for all i.

The Metropolized random walk with $\mathbf{P}^{\mathsf{mh\text{-}max}}$ is, intuitively, less efficient than the one with $\mathbf{P}^{\mathsf{mh\text{-}srw}}$ due to larger self-transitions, as shall be made more precise shortly. One natural question is then whether it is possible to find (or construct) another transition matrix that is 'even better' than $\mathbf{P}^{\mathsf{mh\text{-}srw}}$ while achieving the same $\boldsymbol{\pi}$. More generally, is it possible to construct more efficient random walks on graph than the one induced by the MH algorithm with *any* arbitrary proposal matrix (including $\{q_{ij}^{\mathsf{max}}\}$ and $\{q_{ij}^{\mathsf{srw}}\}$)? We address this question throughout the rest of the paper, and focus on the improvement within a class of reversible Markov chains to the maximum extent possible.

III. Comparing Reversible Chains: An Efficiency Perspective

For a given graph G, there are potentially many different irreducible, reversible Markov chains preserving the same stationary distribution $\boldsymbol{\pi}$, e.g., those with $\mathbf{P}^{\mathsf{mh\text{-}srw}}$ and $\mathbf{P}^{\mathsf{mh\text{-}max}}$. It is thus necessary to identify how to compare these 'competing' Markov chains, or rather, which one is 'better' or more efficient than the others. The rate of convergence of the chain to stationarity, or mixing time, has been a common criterion for measuring such speed of convergence. Let $P^t(i,j)$ be its t-step transition probability from state i to state j $(i,j\in\mathcal{S})$, and let $1=\lambda_1>\lambda_2\geq\cdots\geq\lambda_n\geq -1$ be the eigenvalues of $\mathbf{P}$. If we define the total variation distance $||P^t(i,\cdot)-\boldsymbol{\pi}||_{TV}=\max_{A\subseteq\mathcal{S}}|P^t(i,A)-\pi_A|$, and the mixing time as

$$t_{mix}(\varepsilon)=\min\{t\geq 1:\max_{i\in\mathcal{S}}||P^t(i,\cdot)-\boldsymbol{\pi}||_{TV}\leq\varepsilon\},$$

then $t_{mix}(\varepsilon)\leq\log(1/(\varepsilon\pi_{\min}))/(1-\lambda(\mathbf{P}))$ [25], where

$$\lambda(\mathbf{P})=\max\{\lambda_2,|\lambda_n|\} \tag{4}$$

is the second largest eigenvalue modulus (SLEM) of $\mathbf{P}$ [16], [25]. That is, the mixing time is mainly determined by the SLEM, and the smaller SLEM leads to the faster (smaller) mixing time.

In this paper, however, we look at the comparison of reversible Markov chains from a different, but important perspective. This is done based on a partial order called the Peskun ordering [22] between off-diagonal elements of transition matrices of two different Markov chains and its relationship to ordering reversible chains with the same $\boldsymbol{\pi}$ in terms of asymptotic variance and expected hitting times.

Definition 1 (Peskun ordering): [22] For two finite irreducible Markov chains on $\mathcal{S}$ with transition matrices $\mathbf{P}$ and $\tilde{\mathbf{P}}$, it is said that $\tilde{\mathbf{P}}$ dominates $\mathbf{P}$ off the diagonal, written as $\mathbf{P} \preceq \tilde{\mathbf{P}}$, if $P_{ij} \le \tilde{P}_{ij}$ for all $i, j \in \mathcal{S}$ $(i \neq j)$. □

We also define $\mathbf{P} \prec \tilde{\mathbf{P}}$, if $P_{vw} < \tilde{P}_{vw}$ for at least one $(v, w) \in \mathcal{E}$ while $P_{ij} \le \tilde{P}_{ij}$ for all other i, j $(i \neq j)$. Note that the Peskun ordering $\mathbf{P} \preceq \tilde{\mathbf{P}}$ is a partial order (it is reflexive, anti-symmetric, and transitive) while $\mathbf{P} \prec \tilde{\mathbf{P}}$ is a *strict* partial order (it is irreflexive and transitive).

Let $\{X_t\}_{t\ge 0}$ and $\{\tilde{X}_t\}_{t\ge 0}$ be irreducible, *reversible* Markov chains on a finite state space $\mathcal{S}$ with transition matrices $\mathbf{P}$ and $\tilde{\mathbf{P}}$ having the same $\boldsymbol{\pi}$, respectively. For a function $f : \mathcal{S} \to \mathbb{R}$, define an estimator $\hat{\mu}_t = \sum_{s=1}^{t} f(X_s)/t$ for $\mathbb{E}_{\boldsymbol{\pi}}(f) = \sum_{i\in\mathcal{S}} f(i)\pi_i$. It is well known that $\lim_{t\to\infty} \hat{\mu}_t = \mathbb{E}_{\boldsymbol{\pi}}(f)$ for any function f with $\mathbb{E}_{\boldsymbol{\pi}}(|f|) < \infty$ (see, e.g., [25], [26]). The asymptotic variance of the estimate $\hat{\mu}_t$ is defined as

$$\begin{aligned}\sigma^2(\mathbf{P}, f) &= \lim_{t\to\infty} t \cdot \mathrm{Var}\,(\hat{\mu}_t) \\ &= \lim_{t\to\infty} \frac{1}{t}\mathbb{E}\Big\{ \big[\sum_{s=1}^{t} (f(X_s) - \mathbb{E}_{\boldsymbol{\pi}}(f))\big]^2 \Big\}\end{aligned}$$

which is *independent* of the distribution of the initial state X_0 [22]. We similarly define $\sigma^2(\tilde{\mathbf{P}}, f)$ for the estimate $\hat{\mu}_t$ obtained based on $\{\tilde{X}_t\}$ with $\tilde{\mathbf{P}}$. The asymptotic variance decides approximately how many samples are required to achieve a certain accuracy of the estimate $\hat{\mu}_t$,* and the overall correlation structure of the sequence $f(X_s)$. For this reason, it has played an important role in ordering the *efficiency* among competing Markov chains with the same $\boldsymbol{\pi}$ for MCMC samplers [22], [27], and also can serve as a performance metric to evaluate the sampling accuracy of Markov chain samplers (or random walk-based estimators) for unbiased graph sampling [1]–[3], and the queueing performance under distributed wireless scheduling applications [28]. It is often said that $\{\tilde{X}_t\}$ is *at least as efficient as* $\{X_t\}$ if $\sigma^2(\mathbf{P}, f) \ge \sigma^2(\tilde{\mathbf{P}}, f)$ for any f with $\mathrm{Var}_{\boldsymbol{\pi}}(f) < \infty$ [27]. Then, the Peskun ordering between $\mathbf{P}$ and $\tilde{\mathbf{P}}$ implies the efficiency ordering† as follows:

Lemma 1: [22] If $\mathbf{P} \preceq \tilde{\mathbf{P}}$, then $\sigma^2(\mathbf{P}, f) \ge \sigma^2(\tilde{\mathbf{P}}, f)$ for any f with $\mathrm{Var}_{\boldsymbol{\pi}}(f) < \infty$. □

*This is formally captured by the Central Limit Theorem for a finite, irreducible Markov chain under $\mathrm{Var}_{\boldsymbol{\pi}}(f) < \infty$, saying that $\sqrt{t}(\hat{\mu}_t - \mathbb{E}_{\boldsymbol{\pi}}(f))$ converges to a Gaussian random variable with zero mean and variance $\sigma^2(\mathbf{P}, f)$ as $t \to \infty$ [26].

†The efficiency ordering does not imply the mixing time ordering in general, although they are related with each other [27].

While the Peskun ordering was originally introduced along with its implication for the asymptotic variance (efficiency ordering), we below show that the Peskun ordering also enables to compare reversible chains with the same $\boldsymbol{\pi}$ for the expected hitting time. We define the hitting time of a subset $A \subseteq \mathcal{S}$ for the Markov chain $\{X_t\}$ as $T_A = \min\{t \ge 0 : X_t \in A\}$, and then define two different *expected* hitting times as follows. For any $i \in \mathcal{S}$ and $A \subseteq \mathcal{S}$,

$$\mathbb{E}_i\{T_{\boldsymbol{\pi}}\} = \sum_{j\in\mathcal{S}} \mathbb{E}\{T_j | X_0 = i\}\pi_j, \text{ and} \tag{5}$$

$$\mathbb{E}_{\boldsymbol{\pi}}\{T_A\} = \sum_{i\in\mathcal{S}} \mathbb{E}\{T_A | X_0 = i\}\pi_i. \tag{6}$$

It is known that the quantity $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$, which is also called the Kemeny's constant, does not depend on i [29], [30]. It thus follows that $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\} = \sum_i \pi_i \mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$. Since the Kemeny's constant is the expected time to reach a random destination, it is closely related to the random search time via random walk search methods, including the one based on the MH algorithm in peer-to-peer networks [7]. In addition, it is often necessary to measure the delay taken for the walk (with a desired stationary distribution) until to reach a subset of nodes in various network applications [8], [31], [32]. Thus, the above expected hitting times can serve as important performance metrics. As in (5)–(6), we similarly define the hitting time of $A \subseteq \mathcal{S}$ for $\{\tilde{X}_t\}$ as $\tilde{T}_A = \min\{t \ge 0 : \tilde{X}_t \in A\}$, and subsequently define $\mathbb{E}_i\{\tilde{T}_{\boldsymbol{\pi}}\} = \sum_{j\in\mathcal{S}} \mathbb{E}\{\tilde{T}_j | \tilde{X}_0 = i\}\pi_j$ and $\mathbb{E}_{\boldsymbol{\pi}}\{\tilde{T}_A\} = \sum_{i\in\mathcal{S}} \mathbb{E}\{\tilde{T}_A | \tilde{X}_0 = i\}\pi_i$ for $i \in \mathcal{S}$ and $A \subseteq \mathcal{S}$, respectively. We then have the following.

Lemma 2: If $\mathbf{P} \preceq \tilde{\mathbf{P}}$, then $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\} \ge \mathbb{E}_i\{\tilde{T}_{\boldsymbol{\pi}}\}$ for all $i \in \mathcal{S}$, and $\mathbb{E}_{\boldsymbol{\pi}}\{T_A\} \ge \mathbb{E}_{\boldsymbol{\pi}}\{\tilde{T}_A\}$ for any non-empty subset $A \subseteq \mathcal{S}$. □

Proof: Let $1 = \lambda_1 > \lambda_2 \ge \cdots \ge \lambda_n \ge -1$ be the eigenvalues of $\mathbf{P}$, and $1 = \tilde{\lambda}_1 > \tilde{\lambda}_2 \ge \cdots \ge \tilde{\lambda}_n \ge -1$ be the eigenvalues of $\tilde{\mathbf{P}}$. It is known that if $\mathbf{P}$ and $\tilde{\mathbf{P}}$ are reversible with respect to $\boldsymbol{\pi}$, and $\mathbf{P} \preceq \tilde{\mathbf{P}}$, then $\lambda_i \ge \tilde{\lambda}_i$ for $i = 2, 3, \ldots, n$ [27]. It thus follows that for $i \in \mathcal{S}$,

$$\mathbb{E}_i\{T_{\boldsymbol{\pi}}\} = \sum_{k=2}^{n} \frac{1}{1-\lambda_k} \ge \sum_{k=2}^{n} \frac{1}{1-\tilde{\lambda}_k} = \mathbb{E}_i\{\tilde{T}_{\boldsymbol{\pi}}\}. \tag{7}$$

The representation of $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$ (resp. $\mathbb{E}_i\{\tilde{T}_{\boldsymbol{\pi}}\}$) in terms of eigenvalues of $\mathbf{P}$ (resp. $\tilde{\mathbf{P}}$) can be found in [29, Ch.3, Proposition 13] or [30, Eq.(10)].

Next, the extremal characterization of the mean hitting time of a subset for a reversible Markov chain in [29, Ch.3, Proposition 41] says that for any non-empty subset $A \subseteq \mathcal{S}$,

$$\mathbb{E}_{\boldsymbol{\pi}}\{T_A\} = \sup_g \left\{ \frac{1}{\mathcal{E}_{\mathbf{P},\boldsymbol{\pi}}(g,g)} : -\infty < g < \infty,\ g(\cdot) = 1 \text{ on } A, \right. \\ \left. \text{and } \sum_{i\in\mathcal{S}} \pi_i g(i) = 0 \right\}, \tag{8}$$

where the Dirichlet form $\mathcal{E}_{\mathbf{P},\boldsymbol{\pi}}(g,g)$ is defined as $\mathcal{E}_{\mathbf{P},\boldsymbol{\pi}}(g,g) = \frac{1}{2}\sum_{i,j\in\mathcal{S}} \pi_i P_{ij} \left(g(i) - g(j)\right)^2$ for functions $g : \mathcal{S} \to \mathbb{R}$ exclud-

ing $g \equiv 0$. Then, from $\mathbf{P} \preceq \tilde{\mathbf{P}}$, we have

$$\mathcal{E}_{\mathbf{P},\boldsymbol{\pi}}(g,g) = \frac{1}{2}\sum_{i,j\in\mathcal{S}} \pi_i P_{ij}\left(g(i)-g(j)\right)^2 \leq \frac{1}{2}\sum_{i,j\in\mathcal{S}} \pi_i \tilde{P}_{ij}\left(g(i)-g(j)\right)^2 = \mathcal{E}_{\tilde{\mathbf{P}},\boldsymbol{\pi}}(g,g)$$

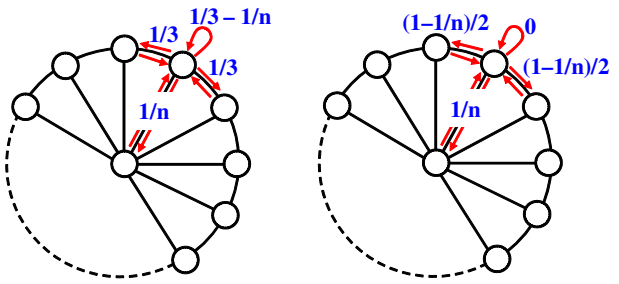
for any given function g. Together with (8), this implies that $\mathbb{E}_{\boldsymbol{\pi}}\{T_A\} \geq \mathbb{E}_{\boldsymbol{\pi}}\{\tilde{T}_A\}$. ■

From (2)–(3), one can see that $\mathbf{P}^{\text{mh-srw}} \succeq \mathbf{P}^{\text{mh-max}}$, i.e., $\mathbf{P}^{\text{mh-srw}}$ is more efficient than $\mathbf{P}^{\text{mh-max}}$. Clearly, in view of Lemmas 1–2, $\mathbf{P}^{\text{mh-srw}}$ is a better choice than $\mathbf{P}^{\text{mh-max}}$ when constructing a random walk on a graph with a given stationary distribution $\boldsymbol{\pi}$. Here, we note that there exists a tradeoff between cost and efficiency; $\mathbf{P}^{\text{mh-max}}$ with proposal distribution $q_{ij}^{\text{max}} = 1/d$ for any $d \geq d_{\max}$ requires only local information such as the target stationary distribution of neighbors up to a constant multiple (i.e., π_j/π_i for $j \in N(i)$). On the other hand, $\mathbf{P}^{\text{mh-srw}}$ with proposal distribution $q_{ij}^{\text{srw}} = 1/d_i$ achieves higher efficiency at the cost of additional information – the degree of neighbors, i.e., d_j for $j \in N(i)$.

IV. Main Results

A. Revisiting the MH Algorithm: New Observation

We first point out that there is a room for further improvement on the MH algorithm when running on a graph G. Consider $\mathbf{P}^{\text{mh-srw}}$ obtained from the MH algorithm in (3) to achieve a uniform stationary distribution of interest. As an example, consider a 'wheel' graph with $n+1$ nodes. Figure 1(a) is the resulting MH chain in (3) with uniform target distribution $\pi_i = 1/(n+1)$. The walk has self-transition probability of $\frac{1}{3}-\frac{1}{n}$ at each of n nodes on the rim, while it has zero self-transition probability at the center node. Figure 1(b) shows another reversible chain with transition matrix $\tilde{\mathbf{P}}$, where the self-transition probabilities are symmetrically redirected into more 'useful' transitions among nodes on the rim. Clearly, $\tilde{\mathbf{P}}$ is reversible with respect to the same uniform stationary distribution, but now $\tilde{\mathbf{P}} \succ \mathbf{P}^{\text{mh-srw}}$. This example clearly demonstrates possible improvement over $\mathbf{P}^{\text{mh-srw}}$ when the stationary distribution is uniform. In what follows, we explain why such an improvement is possible in a much more general setting with arbitrary stationary distribution, and formally state how to achieve the improvement.



(a) $\mathbf{P}^{\text{mh-srw}}$ (b) $\tilde{\mathbf{P}}$

Fig. 1. A wheel graph with $n+1$ nodes and uniform target distribution

Consider a transition matrix $\mathbf{P} = \{P_{ij}\}_{i,j\in\mathcal{N}}$ obtained from the MH algorithm with any proposal matrix $\mathbf{Q}$ in order to achieve a given, desired stationary distribution $\boldsymbol{\pi}$. Suppose that there are two nodes v and w that are neighbors of each other, i.e., $(v,w) \in \mathcal{E}$, and that self-transition probabilities at nodes v and w are both positive, i.e., $P_{vv} > 0$ and $P_{ww} > 0$. This situation arises, in fact, quite frequently. For instance, consider $\mathbf{P}^{\text{mh-srw}}$ in (3) with a uniform stationary distribution. For an edge (v,w), if there exists a neighbor x of node v and another neighbor y of node w such that $d_x > d_v$ and $d_y > d_w$, then $P_{vv}^{\text{mh-srw}} > 0$ and $P_{ww}^{\text{mh-srw}} > 0$. This is because $P_{vx}^{\text{mh-srw}} = P_{xv}^{\text{mh-srw}} = \min\{1/d_v, 1/d_x\} < 1/d_v$ and $P_{wy}^{\text{mh-srw}} = P_{yw}^{\text{mh-srw}} = \min\{1/d_w, 1/d_y\} < 1/d_w$. See Figure 2 for an illustrative example. Similarly, even for a non-uniform stationary distribution, it is very likely that such 'fat' edges whose both ends have non-zero self-transition probabilities exist whenever there is a mismatch between steady-state probabilities π_i and node degrees d_i.

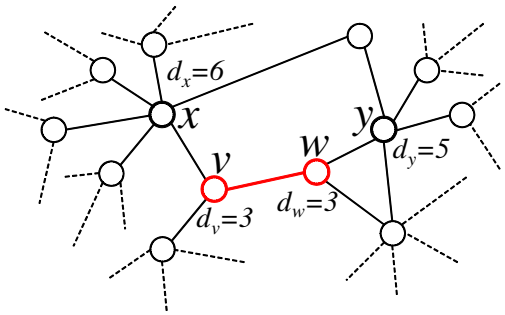


Fig. 2. An example graph with 'fat edge' (v,w)

We then note that there are infinitely many ways to improve the chain $\mathbf{P}$ given by the MH algorithm (with any proposal matrix) in the sense of Peskun ordering, by simply modifying transition probabilities between nodes v and w as follows:

> Construct a new transition matrix $\tilde{\mathbf{P}} = \{\tilde{P}_{ij}\}$ in which, for any constant $\beta_v, \beta_w \in (0,1]$,
>
> $$\tilde{P}_{vw} = P_{vw} + \min\left\{\beta_v P_{vv}, \beta_w P_{ww}\frac{\pi_w}{\pi_v}\right\}, \text{ and}$$
> $$\tilde{P}_{wv} = P_{wv} + \min\left\{\beta_w P_{ww}, \beta_v P_{vv}\frac{\pi_v}{\pi_w}\right\}, \quad (9)$$
>
> while $\tilde{P}_{ij} = P_{ij}$ for all the other i,j $(i \neq j)$, and $\tilde{P}_{ii} = 1 - \sum_{j\neq i}\tilde{P}_{ij}$ for all $i \in \mathcal{N}$.

The new transition matrix $\tilde{\mathbf{P}}$ is also reversible with respect to $\boldsymbol{\pi}$ by noting that

$$\pi_v\tilde{P}_{vw} = \pi_v P_{vw} + \min\{\pi_v\beta_v P_{vv}, \pi_w\beta_w P_{ww}\} = \pi_w P_{wv} + \min\{\pi_w\beta_w P_{ww}, \pi_v\beta_v P_{vv}\} = \pi_w\tilde{P}_{wv},$$

while $\tilde{\mathbf{P}}$ is (strictly) better than $\mathbf{P}$ in the sense of Peskun ordering, i.e., $\tilde{\mathbf{P}} \succ \mathbf{P}$. Note that the above operation of constructing $\tilde{\mathbf{P}}$ from $\mathbf{P}$ is only done between neighboring nodes v and w. This is the main ingredient for our proposed distributed algorithms improving the MH algorithm to *the maximum*

extent possible. To be precise, we define the following notion of Peskun-optimality.

Definition 2: For a given graph G, consider an irreducible, reversible Markov chain $\mathbf{P}$ on G achieving stationary distribution $\boldsymbol{\pi}$. $\mathbf{P}$ is said to be *Peskun-optimal*, if there does not exist another irreducible, reversible Markov chain on G with transition matrix $\tilde{\mathbf{P}} \succ \mathbf{P}$ while having the same $\boldsymbol{\pi}$. □

Peskun-optimal transition matrix does not need to be unique for a given $\boldsymbol{\pi}$. Let $\Omega_{\boldsymbol{\pi}}$ be the set of all transition matrices of reversible chains defined over the same graph G with the same $\boldsymbol{\pi}$. Since $\tilde{\mathbf{P}} \succ \mathbf{P}$ is a strict partial order, based on the ordering between *every* off-diagonal element of $\tilde{\mathbf{P}}$ and $\mathbf{P}$, there may exist $\mathbf{P}, \mathbf{P}' \in \Omega_{\boldsymbol{\pi}}$ for which neither $\mathbf{P} \succ \mathbf{P}'$ nor $\mathbf{P} \prec \mathbf{P}'$ holds, and thus there can be multiple Peskun-optimal transition matrices. It is also possible that the standard MH algorithm readily provides the Peskun-optimal transition matrix. For instance, if $\pi_i \propto d_i$ for each i, then $\mathbf{P}^{\text{mh-srw}}$ is already Peksun-optimal. Nonetheless, as mentioned earlier, further improvement is possible for very general cases whenever there is a mismatch between the desired stationary distribution $\boldsymbol{\pi}$ and the stationary distribution of the proposal matrix $\mathbf{Q}$. Our next result presents a convenient way to test for Peskun-optimality (or efficiency-optimality).

Lemma 3: For an irreducible, reversible Markov chain on a graph G with transition matrix $\mathbf{P} = \{P_{ij}\}_{i,j \in \mathcal{N}}$ achieving a stationary distribution $\boldsymbol{\pi}$, $\mathbf{P}$ is Peskun-optimal if and only if $P_{ii} \cdot P_{jj} = 0$ for every $(i,j) \in \mathcal{E}$. □

Proof: ($\Longrightarrow$) Suppose that $\mathbf{P}$ is Peskun-optimal and there exists an edge $(v,w) \in \mathcal{E}$ such that $P_{vv} \cdot P_{ww} > 0$ ($P_{vv} > 0$ and $P_{ww} > 0$). As shown above, we can construct a new transition matrix $\tilde{\mathbf{P}} \succ \mathbf{P}$ which is reversible with respect to the same $\boldsymbol{\pi}$. Since the Markov chain with $\mathbf{P}$ is irreducible, and $\tilde{P}_{ij} \geq P_{ij}$ for all i,j ($i \neq j$), the Markov chain with $\tilde{\mathbf{P}}$ remains irreducible. Thus, $\mathbf{P}$ is not Peskun-optimal, which leads to a contradiction.

($\Longleftarrow$) Suppose that $\mathbf{P}$ is not Peskun-optimal, or there exists another transition matrix $\tilde{\mathbf{P}} \succ \mathbf{P}$ of an irreducible, reversible Markov chain on G with the same $\boldsymbol{\pi}$. So, there should be at least one $(v,w) \in \mathcal{E}$ such that $\tilde{P}_{vw} > P_{vw}$ while $\tilde{P}_{ij} \geq P_{ij}$ for all other i,j ($i \neq j$). It then follows that $P_{vv} = 1 - \sum_{u \neq v} P_{vu} > 1 - \sum_{u \neq v} \tilde{P}_{vu} = \tilde{P}_{vv} \geq 0$. In addition, from the reversibility condition with respect to $\boldsymbol{\pi}$, we have $\tilde{P}_{wv} = \frac{\pi_v}{\pi_w} \tilde{P}_{vw} > \frac{\pi_v}{\pi_w} P_{vw} = P_{wv}$. Following the same line, we also have $P_{ww} > 0$. Therefore, there exists an edge (v,w) such that $P_{vv} > 0$ and $P_{ww} > 0$. ■

Remark 1: The chain $\tilde{\mathbf{P}}$ in Figure 1(b) can be obtained by setting uniform $\boldsymbol{\pi}$ and applying the operation in (9) with $\beta_v = \beta_w = 1/2$ for neighboring nodes $v \neq w$ on the rim. Lemma 3 guarantees that this chain $\tilde{\mathbf{P}}$ is Peskun-optimal. □

B. Distributed Algorithms for Achieving Efficiency-Optimality

We present simple yet effective distributed algorithms to obtain the Peskun-optimality when starting from the plain MH algorithm running on a network or graph where each node can communicate with its neighbors to exchange local information. In this setting, the Metropolized random walk corresponds to a packet, message, or agent of interest moving over the nodes in the graph, and its transition from one node to another is controlled and governed by each node on the graph.

Recall that the operation in (9) is done between neighboring nodes v and w to obtain a 'Peskun-better' transition matrix $\tilde{\mathbf{P}}$ from the plain $\mathbf{P}$, and so it is implementable via local information exchange between v and w in a distributed manner. That is, an improvement over the plain MH algorithm up to the Peskun-optimality should be possible by leveraging local message passing to conduct the operation in (9) with properly chosen β_v and β_w for every edge (v,w) with non-zero self-transition probabilities at both ends. This observation allows us to develop distributed and iterative algorithms using local message passing. Before proceed, we explain the basic principle behind our proposed algorithms.

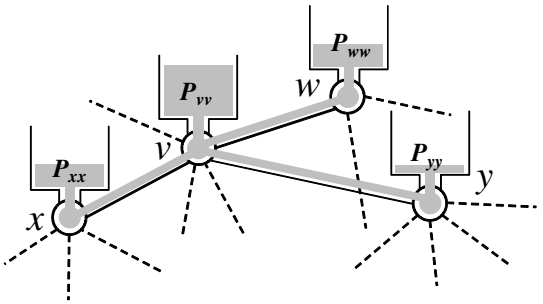


Fig. 3. Progressive emptying process to drain out as much water as possible from every tank.

Imagine that each node i is equipped with a water tank in which the amount of water is given by the self-transition probability P_{ii} arising from the plain MH algorithm, and there exists a water pipe over each edge $(i,j) \in \mathcal{E}$. Figuratively speaking, our objective here is to perform *progressive emptying* in order to drain out as much water as possible in every tank (ideally at the same rate while ensuring the reversibility) under the condition that the water drainage is only available for each non-empty water tank, connected to other non-empty tank(s) via the water pipe(s) as depicted in Figure 3. The emptying process at each node i, if started, continues until its water tank becomes empty or all its neighbors' water tanks become empty. This is conceptually similar to the well-known *progressive filling* algorithm (but opposite to our objective) to achieve max-min fair allocation [33], which is why we use the term 'progressive-emptying' as a mnemonic.

We first present a distributed iterative algorithm under a synchronized network setting, whose operations are described in Algorithm 1. It aims to iteratively keep improving the MH algorithm in the sense of Peskun ordering and finally attain the Peskun-optimality. Let $\mathbf{P}_k = \{P_k(i,j)\}_{i,j \in \mathcal{N}}$ be the transition matrix by our proposed algorithm at k^{th} iteration, with $\mathbf{P}_0$ set to be the one by the MH algorithm with any arbitrarily given proposal matrix. $\mathbf{P}_\infty$ denotes the transition matrix when the operation of our algorithm stops at every node, or no further update of the transition matrix is available. We also define by $m_k(i) = |\{j \in N(i) : P_k(j,j) > 0\}|$ the number of neighbors

of node i, each of which has *non-zero* self-transition transition probability, at k^{th} iteration.

Algorithm 1 Synchronous Algorithm ($k = 1, 2, \ldots$)

1: For each node $i \in \mathcal{N}$:
2: **if** $P_{k-1}(i,i) > 0$ and $m_{k-1}(i) > 0$ **then**
3: Update the transition probabilities from node i as

$$P_k(i,j) = \begin{cases} P_{k-1}(i,j) + \min\left\{\frac{P_{k-1}(i,i)}{m_{k-1}(i)}, \frac{P_{k-1}(j,j)}{m_{k-1}(j)}\frac{\pi_j}{\pi_i}\right\}, \\ \qquad\qquad \text{if } (i,j) \in \mathcal{E} \text{ and } p_{k-1}(j,j) > 0, \\ P_{k-1}(i,j), \text{ if } (i,j) \in \mathcal{E} \text{ and } P_{k-1}(j,j) = 0, \\ 0, \qquad\quad \text{if } (i,j) \notin \mathcal{E} \text{ and } i \neq j, \end{cases} \tag{10}$$

with $P_k(i,i) = 1 - \sum_{j \neq i} P_k(i,j)$.
4: Update $m_k(i)$.
5: **else**
6: Stop the algorithm operation (no update of the transition probabilities starting from node i).
7: **end if**

Algorithm 1 is, in fact, nothing but to repeat the operation in (9) with $\beta_i = 1/m_{k-1}(i)$ and $\beta_j = 1/m_{k-1}(j)$ for every edge $(i,j) \in \mathcal{E}$ for which $P_{k-1}(i,i)P_{k-1}(j,j) > 0$ at every k^{th} iteration ($k \geq 1$). It can also be thought of as a realization of performing the aforementioned progressive emptying. We then have the following.

Theorem 1 (Properties of Algorithm 1): $\mathbf{P}_k$ is reversible with respect to $\boldsymbol{\pi}$, and $\mathbf{P}_{k-1} \preceq \mathbf{P}_k$ for all $k \geq 1$. Also, $\mathbf{P}_\infty$ is Peskun-optimal. □

Proof: The reversibility proof is by induction. By definition of the MH algorithm, $\mathbf{P}_0$ is reversible with respect to $\boldsymbol{\pi}$. As the induction hypothesis, assume that $\mathbf{P}_{k-1} = \{P_{k-1}(i,j)\}_{i,j \in \mathcal{N}}$ is reversible with respect to $\boldsymbol{\pi}$ for any $k = 1, 2, \ldots$. Fix node $v \in \mathcal{N}$. Suppose that $P_{k-1}(v,v) > 0$ and $m_{k-1}(v) > 0$. (If otherwise, the algorithm would stop at node v, implying that $P_k(v,u) = P_{k-1}(v,u)$ and $P_k(u,v) = P_{k-1}(u,v)$ for all $u \in \mathcal{N} \backslash \{v\}$, and the reversibility condition between nodes v and u would hold by the induction hypothesis.) Then, observe that $m_{k-1}(v) > 0$ means there exists a neighbor w of node v such that $P_{k-1}(w,w) > 0$. At the same time, for such node w, $P_{k-1}(v,v) > 0$ implies that $m_{k-1}(w) > 0$. Thus, by the operation in (10), we have

$$\begin{aligned} &\pi_v P_k(v,w) \\ &= \pi_v P_{k-1}(v,w) + \min\left\{\pi_v \frac{P_{k-1}(v,v)}{m_{k-1}(v)}, \pi_w \frac{P_{k-1}(w,w)}{m_{k-1}(w)}\right\} \\ &= \pi_w P_{k-1}(w,v) + \min\left\{\pi_w \frac{P_{k-1}(w,w)}{m_{k-1}(w)}, \pi_v \frac{P_{k-1}(v,v)}{m_{k-1}(v)}\right\} \\ &= \pi_w P_k(w,v), \end{aligned}$$

where the second equality is from the induction hypothesis. In addition, if $P_{k-1}(x,x) = 0$ for any neighbor x of node v, then the algorithm operation in (10) says that $P_k(v,x) = P_{k-1}(v,x)$. In this case, the algorithm also stops at node x, which implies $P_k(x,v) = P_{k-1}(x,v)$. The induction hypothesis ensures the reversibility condition between nodes v and x. The reversibility trivially holds for the other cases in (10). Therefore, $\mathbf{P}_k$ is reversible with respect to $\boldsymbol{\pi}$ for all k.

Moreover, $\mathbf{P}_{k-1} \preceq \mathbf{P}_k$ for $k = 1, 2, \ldots$ follows from the algorithm operation in (10) implying that $P_k(i,j) \geq P_{k-1}(i,j)$ for all $i, j \in \mathcal{N}$ ($i \neq j$) and thus $P_k(i,i) \leq P_{k-1}(i,i)$ for all i. Finally, the stopping rule in Algorithm 1 ensures that $\mathbf{P}_\infty$ satisfies the condition in Lemma 3 for Peskun-optimality. ■

Remark 2: Algorithm 1 can be modified in many different ways while satisfying Theorem 1. For instance, $m_{k-1}(i)$ and $m_{k-1}(j)$ in (10) can be replaced by any larger integer values, e.g., d_i and d_j, respectively. □

Remark 3: Since the transition matrix $\mathbf{P}_t$ is time-varying, the resulting Markov chain is *time-inhomogeneous*. It is not difficult to see that for any t, $\mathbf{P}_t$ is aperiodic, as the undirected, connected graph G is non-bipartite (i.e., it contains an odd cycle), and has the same unique stationary distribution $\boldsymbol{\pi}$. It then follows from Theorem 8.5 [34, pp.247] that the convergence to the stationary distribution $\boldsymbol{\pi}$ is ensured. □

We also point out that any transition matrix $\mathbf{P}_t \succeq \mathbf{P}_0$ ($t > 0$) by Algorithm 1 is still beneficial even when the desired stationary distribution changes from $\boldsymbol{\pi}$ to $\boldsymbol{\pi'}$ over time, as typically would be the case in dynamic environments. To see this, we first show an order-preserving property on the Peskun ordering. Suppose that there are two different transition matrices $\mathbf{Q} = \{q_{ij}\}$ and $\tilde{\mathbf{Q}} = \{\tilde{q}_{ij}\}$ of irreducible, reversible Markov chains on the same graph G (that may have different stationary distributions). For a given graph G, we obtain new transition matrices $\mathbf{P}$ and $\tilde{\mathbf{P}}$, both reversible with respect to $\boldsymbol{\pi'}$, from the MH algorithm using *proposal* matrices $\mathbf{Q}$ and $\tilde{\mathbf{Q}}$, respectively. We then have the following.

Lemma 4: For any stationary distribution $\boldsymbol{\pi'}$, if $\tilde{\mathbf{Q}} \succeq \mathbf{Q}$, then $\tilde{\mathbf{P}} \succeq \mathbf{P}$. □

Proof: For every edge $(i,j) \in \mathcal{E}$, from (1), observe that $\tilde{P}_{ij} = \min\{\tilde{q}_{ij}, \tilde{q}_{ji}\pi'_j/\pi'_i\} \geq \min\{q_{ij}, q_{ji}\pi'_j/\pi'_i\} = P_{ij}$ since $\tilde{\mathbf{Q}} \succeq \mathbf{Q}$. Also, $\tilde{P}_{ij} = P_{ij} = 0$ for every $(i,j) \notin \mathcal{E}$ ($i \neq j$). Therefore, $\tilde{\mathbf{P}} \succeq \mathbf{P}$. ■

Lemma 4 asserts that any transition matrix $\mathbf{P}_t$ by Algorithm 1 is *better reusable* as a proposal matrix for the MH algorithm than any other $\mathbf{P}_s$ ($s < t$), in order to achieve a different stationary distribution $\boldsymbol{\pi'}$. Since the new transition matrix (say, $\mathbf{P'}_t$), which is reversible with respect to $\boldsymbol{\pi'}$, may not be Peskun-optimal, we can also subsequently run Algorithm 1 starting from $\mathbf{P'}_t$ to attain a Peskun-optimal transition matrix with respect to $\boldsymbol{\pi'}$.

We next explain how to modify Algorithm 1 to be working asynchronously, as it may not be always the case that every node in a graph can update its transition probabilities simultaneously at each iteration. For example, when a multi-hop network composed of wireless terminals (or nodes) forms a graph according to the communication availability between geographically neighboring nodes, a node cannot communicate with its multiple neighbors at the same time due to interference. It is thus desirable to develop an *asynchronous* algorithm improving the MH algorithm on a graph, which is

done iteratively, but asynchronously in a distributed fashion. In view of the operation in (9), this should be always possible whenever there exists an edge (i,j) with non-zero self-transition probabilities at both ends. Algorithm 2 shows one such instance where $\beta_i = \beta_j = 1$ for a randomly chosen edge $(i,j) \in \mathcal{E}$, if $P_{k-1}(i,i)P_{k-1}(j,j) > 0$, at every k^{th} iteration ($k \geq 1$). $\mathbf{P}_0$ is again set to be the one by the MH algorithm.

Algorithm 2 Asynchronous Algorithm ($k = 1, 2, \ldots$)

1: Choose an edge $(i,j) \in \mathcal{E}$ uniformly at random.
2: **if** $P_{k-1}(i,i) > 0$ and $P_{k-1}(j,j) > 0$ **then**
3: Update the transition probabilities starting from each of nodes i and j as

$$P_k(i,j) = P_{k-1}(i,j) + \min\left\{P_{k-1}(i,i), P_{k-1}(j,j)\frac{\pi_j}{\pi_i}\right\},$$

while $P_k(i,u) = P_{k-1}(i,u)$ for $u \in \mathcal{N} \setminus \{i,j\}$, and

$$P_k(j,i) = P_{k-1}(j,i) + \min\left\{P_{k-1}(j,j), P_{k-1}(i,i)\frac{\pi_i}{\pi_j}\right\},$$

while $P_k(j,v) = P_{k-1}(j,v)$ for $v \in \mathcal{N} \setminus \{i,j\}$,
with $P_k(i,i) = 1 - \sum_{u \neq i} P_k(i,u)$ and $P_k(j,j) = 1 - \sum_{v \neq j} P_k(j,v)$.
4: **end if**

As can be seen from Algorithm 2 (uniform edge selection), only two neighboring nodes i and j on a graph can communicate with each other at each iteration in order to update the transition probability from i to j (and vise versa) as well as their self-transition probabilities ensuring the sums of rows equal to one. Note that the uniform edge selection can be achieved by running Poisson clocks in a real implementation. Specially, each edge $(i,j) \in \mathcal{E}$ can be associated with a Poisson clock with the same rate γ, or any one of two neighboring nodes can be associated with a Poisson clock with γ initiating their communication.

In Algorithm 2, we do not explicitly state its stopping rule. As shall be shown below, once every edge $(i,j) \in \mathcal{E}$ has been chosen by the uniform edge selection, the update of the transition matrix is no longer available, and we will then have reached $\mathbf{P}_\infty$. Thus, the *stopping rule* can be made from a node point of view as follows: for each node i, as long as each of edges incident to node i has been selected at least once, node i stops the algorithm operation (no more update of transition probabilities starting from node i). It is also worth noting that due to the randomized nature of Algorithm 2, the convergent transition matrix $\mathbf{P}_\infty$ is not unique but can be different, depending on the realizations of the random sequence of selected edges. Nonetheless, all those $\mathbf{P}_\infty$ are still Peskun-optimal, as shall be shown below. In addition, we also obtain an upper bound on the 'expected time to converge' to $\mathbf{P}_\infty$, formally defined as $\mathbb{E}\{Z\} = \mathbb{E}\{\min\{k \geq 0 : \mathbf{P}_k = \mathbf{P}_\infty\}\}$.

Theorem 2 (Properties of Algorithm 2): For all $k \geq 1$, $\mathbf{P}_k$ is reversible w.r.t. $\boldsymbol{\pi}$, and $\mathbf{P}_{k-1} \preceq \mathbf{P}_k$. Also, $\mathbf{P}_\infty$ is Peskun-optimal, and $\mathbb{E}\{Z\} \leq |\mathcal{E}| \sum_{l=1}^{|\mathcal{E}|} \frac{1}{l} = \Theta(|\mathcal{E}| \log |\mathcal{E}|)$. □

Proof: For a uniformly chosen edge $(i,j) \in \mathcal{E}$ such that $P_{k-1}(i,i) > 0$ and $P_{k-1}(j,j) > 0$, if $\pi_i P_{k-1}(i,j) = \pi_j P_{k-1}(j,i)$, then by the algorithm operations, we have

$$\begin{aligned}\pi_i P_k(i,j) &= \pi_i P_{k-1}(i,j) + \min\{\pi_i P_{k-1}(i,i), \pi_j P_{k-1}(j,j)\} \\ &= \pi_j P_{k-1}(j,i) + \min\{\pi_j P_{k-1}(j,j), \pi_i P_{k-1}(i,i)\} \\ &= \pi_j P_k(j,i).\end{aligned}$$

Thus, after repeating the similar induction argument as in the proof of Theorem 1, one can show that $\mathbf{P}_k$ is reversible with respect to $\boldsymbol{\pi}$ for all k. It also easily follows from the algorithm operations that $\mathbf{P}_{k-1} \preceq \mathbf{P}_k$ for all k.

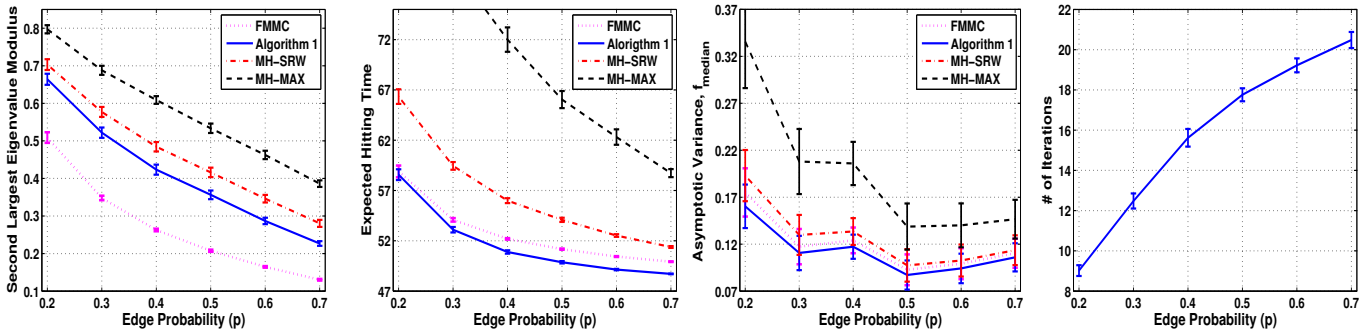
Observe now that for node i and its neighbor j with $P_{k-1}(i,i) > 0$ and $P_{k-1}(j,j) > 0$, the algorithm operations give that if $P_{k-1}(i,i) < P_{k-1}(j,j)\pi_j/\pi_i$, then $P_k(i,j) = P_{k-1}(i,j) + P_{k-1}(i,i)$, implying $P_k(i,i) = 0$. Similarly, if $P_{k-1}(i,i) \geq P_{k-1}(j,j)\pi_j/\pi_i$, then $P_k(j,j) = 0$. Thus, after every edge $(i,j) \in \mathcal{E}$ has been chosen at least once, there does not exist node i and its neighbor j with $P_k(i,i) > 0$ and $P_k(j,j) > 0$. By Lemma 3, the convergent transition matrix $\mathbf{P}_\infty$ is Peskun-optimal, irrespective of any realization sequence of the randomly selected edges. This also implies that $\mathbb{E}\{Z\}$ is upper-bounded by the cover time of a complete graph composed of $|\mathcal{E}|$ nodes with self-loops, i.e., the average number of time steps taken by a simple random walk until to visit all $|\mathcal{E}|$ nodes. By the usual coupon collector arguments, we have $\mathbb{E}\{Z\} \leq |\mathcal{E}| \sum_{l=1}^{|\mathcal{E}|} 1/l = \Theta(|\mathcal{E}| \log |\mathcal{E}|)$. ■

V. Numerical Evaluation

In this section, we present numerical results to demonstrate how our algorithms leading to the class of efficiency-optimal chains perform and quantitatively capture their improvements over existing algorithms in the literature for a number of key metrics such as the asymptotic variance, expected hitting time, as well as the SLEM $\lambda(\mathbf{P})$ (or the mixing time) over various graph settings. We first consider a family of random graphs [16] with 50 nodes. To generate such a graph G, we assign random numbers R_{ij} (independently and uniformly distributed over the interval $[0,1]$) for nodes i and j. Then, for each threshold value $p \in (0,1)$, we place an edge between nodes i and j into the graph G if $R_{ij} < p$. By increasing p from 0.2 to 0.7, we obtain an increasing, monotone family of random graphs with 50 nodes, i.e., it is equivalent to adding edges into the graph G when increasing p from 0.2 to 0.7. Each data point in all our simulation figures is obtained by taking an average over 50 different realizations of the graphs, for each of which the network connectivity is ensured. In all figures, we also provide 95% confidence intervals.

For each realization of the graph with 50 nodes (while the number of edges varies depending on the value of p), we run the following four chains on this graph: (i) the MH chain $\mathbf{P}^{\text{mh-max}}$ in (2), (ii) the MH chain $\mathbf{P}^{\text{mh-srw}}$ in (3), (iii) the efficiency-optimal chain $\mathbf{P}_\infty$ obtained by running our Algorithm 1 starting $\mathbf{P}_0 := \mathbf{P}^{\text{mh-srw}}$, and (iv) the fastest-mixing Markov chain (FMMC) $\mathbf{P}^*$ [16].[‡] For each of these

[‡]To obtain the FMMC for each graph with a given $\boldsymbol{\pi}$, we use CVX [35], a package for specifying and solving convex programs, to solve the SDP problem [16], enabling us to obtain the resulting transition matrix $\mathbf{P}^*$.



(a) SLEM $\lambda(\mathbf{P})$ (b) Expected Hitting Time $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$ (c) Asymptotic variance $\sigma^2(\mathbf{P}, f)$ (d) The number of iterations to $\mathbf{P}_\infty$

Fig. 4. Performance comparison of various chains with the same stationary distribution

chains considered, we then compute all its 50 eigenvalues and eigenvectors, giving us the SLEM of each chain as defined in (4) and the expected hitting time $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$ by using the formula in (7). For the asymptotic variance $\sigma^2(\mathbf{P}, f)$ of a reversible chain $\mathbf{P}$ with a given f, we utilize the following representation [34]:

$$\sigma^2(\mathbf{P}, f) = \sum_{i=2}^{n} \frac{1+\lambda_i}{1-\lambda_i} \left|\langle f, \boldsymbol{v}_i \rangle_{\boldsymbol{\pi}}\right|^2,$$

where $\lambda_i, \boldsymbol{v}_i$, $i = 1, 2, \ldots, n$ are the eigenvalue and eigenvector pair of the matrix $\mathbf{P}$ such that $\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle_{\boldsymbol{\pi}} = \delta_{ij}$ for $i, j = 1, 2, \ldots, n$, where $\delta_{ij} = 1$ if $i = j$ and zero otherwise. Here, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{\pi}} = \sum_{i=1}^{n} x(i) y(i) \pi_i$ is the inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ with respect to $\boldsymbol{\pi}$.

Figures 4(a)–(c) show the resulting SLEM, the expected hitting time, and the asymptotic variance, respectively, under four different chains. In all cases, we assume the target distribution is uniform. For the asymptotic variance $\sigma^2(\mathbf{P}, f)$ in Figure 4(c), we set $f(i) = \mathbf{1}_{\{d(i)=d_{\text{median}}\}}$ for $i \in \mathcal{N}$, where d_{median} is the median degree over each instance of the graph G. Figure 4(d) shows the number of iterations under Algorithm 1 starting $\mathbf{P}_0 := \mathbf{P}^{\text{mh-srw}}$ until the time-varying chain $\mathbf{P}_t$ reaches the class of Peskun-optimal chains $\mathbf{P}_\infty$. First, we observe that in all cases, our Algorithm 1 leading to $\mathbf{P}_\infty$ outperforms $\mathbf{P}^{\text{mh-srw}}$ (MH-SRW), which is still better than $\mathbf{P}^{\text{mh-max}}$ (MH-MAX), as predicted by the Peskun ordering relationship $\mathbf{P}^{\text{mh-max}} \preceq \mathbf{P}^{\text{mh-srw}} \preceq \mathbf{P}_\infty$.[§] There is, however, no such outright comparison between $\mathbf{P}_\infty$ and FMMC. As shown in Figure 4(a), FMMC gives smaller SLEM (and thus faster mixing time) than our efficiency-optimal chain $\mathbf{P}_\infty$. This is well expected since the FMMC is constructed specifically for this purpose by solving the optimization problem of minimizing $\lambda(\mathbf{P})$ out of all reversible chains defined on the same graph with the same $\boldsymbol{\pi}$ [16].

For the expected hitting time and the asymptotic variance, however, we note that our efficiency-optimal chain $\mathbf{P}_\infty$ outperforms the FMMC, as shown in Figures 4(b)–(c), which translate into better delay performance and higher sampling efficiency for various networking applications. More importantly, we here maintain that obtaining $\mathbf{P}_\infty$ is simple and can be made in a distributed fashion, in contrast to the FMMC with mostly centralized and computationally expensive algorithms to solve SDP [16], [17]. As Figure 4(d) shows, the convergence to the efficiency-optimal class is quick, monotonically improving itself over each step (as also shown in Theorem 1), offering far more versatility for various performance metrics and more amenable for dynamic environments with changing $\boldsymbol{\pi}$, when coupled with the 'better reusable' property in Lemma 4.

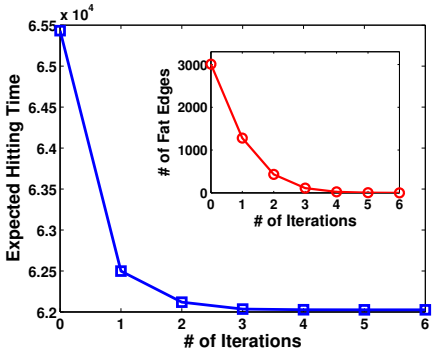


Fig. 5. Expected hitting time on power-grid graph. Inset figure shows the number of 'fat-edges' for chains over iteration.

Figure 5 shows the expected hitting time $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$ as in (7) for the time-varying chain $\mathbf{P}_t$ when running our Algorithm 1, over a power-grid graph. This graph is an undirected, unweighted network with 4941 nodes and 6594 edges, representing the topology of the western states power grid of the United States, and also exhibits the small-world property [36]. We set uniform stationary distribution, with the initial chain $\mathbf{P}_0 = \mathbf{P}^{\text{mh-srw}}$. As expected, $\mathbb{E}_i\{T_{\boldsymbol{\pi}}\}$ monotonically decreases, as the chain becomes more efficient over steps.[¶] The inset figure shows how the number of 'fat-edges' (i.e., edge (i, j) with $P_{ii}P_{jj} > 0$) decreases over steps. It turns out that our Algorithm 1 finishes after only 6 steps, leaving no fat-edge behind, implying $\mathbf{P}_6$ is Peskun-optimal by Lemma 3.

[§]The Peskun ordering guarantees the ordering on the eigenvalues λ_i as also stated in the proof of Lemma 2, not on the SLEM of the chain. Our results on the ordered performance on SLEM thus suggest that $\lambda(\mathbf{P})$ is mainly determined by λ_2, not on $|\lambda_n|$.

[¶]Note that for this size of graph, it was not possible to run the SDP solver to obtain FMMC.

Although we have used Algorithm 1 to obtain $\mathbf{P}_\infty$ in our numerical evaluation for comparison, we point out that the asynchronous version in Algorithm 2 may well run equally for all the scenarios so far. The only difference would be the number of required steps till converging to the class of efficiency-optimal chains, which scales well with the size of the graph, as proven in Theorem 2. In addition, even when no interaction among neighboring node is possible, we note that our algorithms can further be modified so as to obtain efficiency-optimal chains, as long as some local neighborhood information (such as degrees of a neighbor) can be made available to the 'agent' walking on the graph [37], [38]. In this case, whenever the chain (or the agent) is on node i, it can adjust its own transition probabilities from i on-the-fly by utilizing second-hop neighborhood information (i.e., P_{jj} for neighboring node j). Similar to the arguments in the proof of Theorem 2, such an algorithm would converge to the efficiency-optimal one once the chain has visited every node of the graph at least once, i.e., after the 'cover time' of the underlying time-varying chain.

VI. Conclusion

We have developed a formal framework toward efficiency-optimal chains on an arbitrarily given graph, from which no further improvement is possible in the sense of Peskun ordering, for various networking applications. More efficient chains on a graph directly translate into better performance such as smaller average delay and higher sampling efficiency of a graph, and are also closely related with fast mixing Markov chains. Our algorithms scale well with the size of the graph, take advantage of the order-preserving property for dynamic environments, and can be made distributed under various networking constraints with or without node interactions. Our numerical results confirm all the theoretical findings and demonstrate that our algorithms are comparable to (and often better than) the well known fastest mixing Markov chain, but without all those high computational costs. The outcome of our framework (efficiency-optimal reversible chains) can be combined with any other existing module that converts a given reversible chain into a non-reversible one for potentially better performance, while keeping its distributed nature intact.

References

[1] D. Stutzbach, R. Rejaie, N. Duffield, S. Sen, and W. Willinger, "On unbiased sampling for unstructured peer-to-peer networks," *IEEE/ACM Trans. on Networking*, vol. 17, no. 2, pp. 377–390, 2009.

[2] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou, "Practical recommendations on crawling online social networks," *IEEE JSAC*, vol. 29, no. 9, pp. 1872–1892, 2011.

[3] C.-H. Lee, X. Xu, and D. Y. Eun, "Beyond random walk and metropolis-hastings samplers: Why you should not backtrack for unbiased graph sampling," in *ACM SIGMETRICS*, June 2012.

[4] Y. Lin, B. Liang, and B. Li, "Data persistence in large-scale sensor networks with decentralized fountain codes," in *IEEE INFOCOM*, 2007.

[5] D. Vukobratović, Č. Stefanović, V. Crnojević, F. Chiti, and R. Fantacci, "Rateless packet approach for data gathering in wireless sensor networks," *IEEE JSAC*, vol. 28, no. 7, pp. 1169–1179, 2010.

[6] B. Johansson, C. M. Carretti, and M. Johansson, "On distributed optimization using peer-to-peer communications in wireless sensor networks," in *IEEE SECON*, June 2008.

[7] M. Zhong and K. Shen, "Popularity-biased random walks for peer-to-peer search under the square-root principle," in *IPTPS*, Feb. 2006.

[8] M. Zhong, K. Shen, and J. Seiferas, "The convergence-guaranteed random walk and its applications in peer-to-peer networks," *IEEE Trans. on Computers*, vol. 57, pp. 619–633, 2008.

[9] P. Berenbrink, C. Cooper, T. R. R. Elsässer, and T. Sauerwald, "Speeding up random walks with neighborhood exploration," in *ACM-SIAM SODA*, Jan. 2010.

[10] N. Alon, I. Benjamini, E. Lubetzky, and S. Sodin, "Non-backtracking random walks mix faster," *Communications in Contemporary Mathematics*, vol. 9, no. 4, pp. 585–603, 2007.

[11] B. D. Hughes, *Random walks and random environments I*. Oxford University Press, 1995.

[12] A. Awan, R. A. Ferreira, S. Jagannathan, and A. Grama, "Distributed uniform sampling in unstructured peer-to-peer networks," in *HICSS'06*.

[13] D. Wang, Z. Li, and G. Xie, "Towards unbiased sampling of online social networks," in *IEEE ICC*, June 2011.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proceedings of the 16th International Conference on Supercomputing*, June 2002.

[15] C.-H. Lee and D. Y. Eun, "Toward distributed optimal movement strategy for data harvesting in wireless sensor networks," in *IEEE SECON*, June 2012.

[16] S. Boyd, P. Diaconis, and L. Xiao, "Fastest mixing markov chain on a graph," *SIAM Review*, vol. 46, no. 4, pp. 667–689, 2004.

[17] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized Gossip Algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[18] F. Chen, L. Lovász, and I. Pak, "Lifiting markov chains to speed up mixing," in *ACM STOC*, May 1999.

[19] P. Diaconis, S. Holmes, and R. M. Neal, "Analysis of a nonreversible markov chain sampler," *Annals of Applied Probability*, vol. 10, no. 3, pp. 726–752, 2000.

[20] K. Jung and D. Shah, "Fast gossip via nonreversible random walk," in *IEEE Information Theory Workshop (ITW)*, Mar. 2006.

[21] W. Li and H. Dai, "Accelerating distributed consensus via lifting markov chains," in *IEEE ISIT*, June 2007.

[22] P. H. Peskun, "Optimum monte-carlo sampling using markov chains," *Biometrika*, vol. 60, pp. 607–612, 1973.

[23] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[24] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[25] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov chains and mixing times*. American Mathematical Society, 2009.

[26] G. O. Roberts and J. S. Rosenthal, "General state space Markov chains and MCMC algorithms," *Probability Surveys*, vol. 1, pp. 20–71, 2004.

[27] A. Mira, "Ordering and improving the performance of monte carlo markov chains," *Statistical Science*, vol. 16, no. 4, pp. 340–350, 2001.

[28] C.-H. Lee, D. Y. Eun, S.-Y. Yun, and Y. Yi, "From Glauber dynamics To Metropolis algorithm: Smaller delay in optimal CSMA," in *IEEE ISIT*, July 2012.

[29] D. Aldous and J. Fill, *Reversible Markov chains and random walks on graphs*. monograph in preparation.

[30] M. Levene and G. Loizou, "Kemeny's constant and the random surfer," *American Mathematical Monthly*, vol. 109, no. 8, pp. 741–745, 2002.

[31] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-to-peer networks," in *IEEE INFOCOM*, March 2004.

[32] C.-K. Chau and P. Basu, "Exact analysis of latency of stateless opportunistic forwarding," in *IEEE INFOCOM*, April 2009.

[33] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Prentice-Hall, 1992.

[34] P. Brémaud, *Markov chains: gibbs fields, monte carlo simulation, and queues*. Springer-Verlag, 1999.

[35] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," http://cvxr.com/cvx.

[36] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, June 1998.

[37] A. Dasgupta, R. Kumar, and D. Sivakumar, "Social sampling," in *ACM SIGKDD*, Aug. 2012.

[38] P. Wang, B. Ribeiro, J. Zhao, J. C.-S. Lui, D. Towsley, and X. Guan, "Practical characterization of large networks using neighborhood information," http://arxiv.org/abs/1311.3037, Nov. 2013.