SIAM J. SCI. COMPUT.
Vol. 31, No. 1, pp. 45–73

AGGREGATION ALGORITHMS FOR PERTURBED MARKOV CHAINS WITH APPLICATIONS TO NETWORKS MODELING*

ANNA GAMBIN†, PIOTR KRZYŻANOWSKI†, AND PIOTR POKAROWSKI†

Abstract. A powerly perturbed Markov chain (MC) is a family of finite MCs given by transition probability matrices $P(\varepsilon) = (p_{ij}(\varepsilon))$ (or generators for continuous-time MCs) such that $p_{ij}(\varepsilon) = \delta_{ij}\varepsilon^{d_{ij}} + o(\varepsilon^{d_{ij}})$, $\varepsilon \to 0$. In the simplest case we have popular nearly completely decomposable (NCD) chains. It has been shown by directed forest expansions that solutions of linear systems related to $P(\varepsilon)$ have a form $x_i(\varepsilon) = \alpha_i\varepsilon^{a_i} + o(\varepsilon^{a_i})$ for some vectors $\boldsymbol{\alpha} = (\alpha_i)$ and $\mathbf{a} = (a_i)$. Many characteristics of MCs such as stationary distributions, mean passage times, or limiting matrices can be defined in this way. We present efficient combinatorial aggregation algorithms for computing $\mathbf{a}$ and $\boldsymbol{\alpha}$. The algorithms rely on grouping the states of MCs in such a way that the probability of changing the state inside the group is of a greater order of magnitude than the interactions between groups. In contrast to existing methods, our approach is based on a combinatorial framework and can be seen as an algorithmization of the famous matrix tree theorem. We establish some preliminary results on the complexity of our algorithms. Numerical experiments on several network models, such as queueing systems or the Google problem, show the potential applicability of the method in real-life problems.

Key words. Markov chain, decomposability, aggregation

AMS subject classifications. 05C50, 05C85, 15A51, 60J10, 60J22, 65C40, 65F10, 68P20, 68R10, 68W25

DOI. 10.1137/050624716

1. Introduction.

Markov chains (MCs) are used in various areas of computer science and in other disciplines. Markov modeling, in particular reliability modeling, is used to estimate the mean time to failure of components in systems as diverse as software systems and aerospace systems. The applicability of the MC approach is not restricted to computer science and engineering; other disciplines which benefit from it are biology, genetics, agriculture, economics, demographics, and education [9]. Since the long-run behavior of a Markovian system starting from any state is revealed through the stationary distribution, its computation, which leads to a system of linear equations, is crucial in estimating performance measures for modeled systems. For queueing systems, these measures may be the average number of customers, the mean waiting time, or the blocking probability for a specific queue. In communication systems, these measures may be the total packet loss rate, the probability of an empty system, or the communication throughput for packet switching networks.

Besides stationary distribution, some other characteristics of an MC are also considered, such as *first-passage time* between states or the *number of visits in a fixed state before absorption*. To compute them, one has also to solve a system of linear equations.

These systems may, in general, be difficult to solve numerically, mainly because of the large number of states which these systems may occupy. It is not uncommon for thousands of states to be generated even for simple applications. On the other hand these MCs are often sparse and possess specific structure.

*Received by the editors February 18, 2005; accepted for publication (in revised form) April 17, 2008; published electronically October 16, 2008. The research described in this paper was partially supported by Polish Ministry of Education and Science grant N206 004 32/0806.
http://www.siam.org/journals/sisc/31-1/62471.html
†Faculty of Mathematics, Informatics, and Mechanics, Warsaw University, ul. Banacha 2, 02-097 Warsaw, Poland (aniag@mimuw.edu.pl, przykry@mimuw.edu.pl, pokar@mimuw.edu.pl).

Therefore, a lot of research has been done concerning the numerical solution of some linear equations that occur when one studies MCs (see for example [8, 16, 38]). Almost all kinds of methods for solving a system of linear equations are adapted into this context: iterative and direct methods, projection techniques, and the concept of preconditioning (see [39]). The applicability of a method depends strongly on the structure of an MC considered. For example, there exist very efficient methods tailored to problems with a lot of structure, such as Toeplitz or near-Toeplitz matrices [1, 29].

For solving the chain of medium size, direct methods could be applied. Among others, the GTH algorithm (Grassmann-Taksar-Heyman [14]) is adopted in section 3 for dealing with small subsystems arising from the decomposition of a large MC.

When the state space of the chain is large, and the system matrix has sparse structure, most direct solving methods suffer from fill-in, which in turn increases their computational complexity and memory demands, usually beyond acceptable levels.

To avoid immense fill-in, iterative methods can be used, as in that case, the only operation in which the matrices are involved are multiplications by one or more vectors. These operations are cheap and do not alter the form of the matrix. For these reasons, iterative methods (such as Gauss–Seidel iteration, successive over-relaxation, or the power method) have traditionally been preferred to direct methods. On the other hand, a major disadvantage of iterative methods is a very long time often required for convergence to the desired solution, if there is no efficient preconditioner available. The construction of such a preconditioner is, in general, not an easy task; see e.g. [3, 27], and usually requires some additional properties of the problem.

In this paper we consider *nearly uncoupled* or *nearly completely decomposable* (NCD; see [6, 38, 9]) MCs. Such chains often arise in queueing network analysis, large-scale economic modeling, and computer systems performance evaluation. The state space of these chains can be naturally divided into groups of states such that transitions between states belonging to different groups are significantly less likely than transitions between states within the same group.

For solving nearly uncoupled MCs, a family of new methods has been proposed recently. They are jointly classified as *iterative aggregation/disaggregation* [19, 35] methods and are based on a very attractive decompositional approach. The idea follows the well-known *divide and conquer* principle—if the model is too large or too complex to analyze, it is divided into separate subproblems.

The aggregation algorithms are designed to exploit the nearly decomposable structure in one of two ways. In the first approach, some states are excluded from state space, and the chain based on remaining state space is solved; the other way is to lump the groups of closely related states together and solving the lumped MC, treating each lump as a single state. This latter approach, which is discussed in this paper, also forms a background of a new class of aggregation algorithms proposed in [33].

This novel approach, joining successful research from different areas such as combinatorics, linear algebra, and numerical analysis leads to algorithms which have several advantages over previous aggregation methods. Firstly, the applicability of traditional aggregation methods is restricted to MCs with a regular NCD structure; in particular, the existence of asymptotically transient states (i.e., those states with the outgoing probability of a bigger order of magnitude than the ingoing probability; cf. 2.1) are problematic (such states are very common in large MCs resulting from practical examples). Algorithms derived from the method presented here work correctly when such states are present. Also, most of the decomposition methods considered in the literature are designed only to solve the problem of stationary distribution, and other characteristics of the MC are neglected. In contrast, one of the

algorithms presented in section 3 allows us to compute also the *mean hitting time*; designing the procedures for other characteristics is discussed latter. To our knowledge, this is the first time an aggregation scheme was used to calculate MC characteristics other than stationary distribution.

Finally, the algorithm produces not only an approximation to the stationary distribution; it also returns, under reasonable assumptions, an asymptotic formula for the characteristics of small perturbations of the MC under consideration. This may be valuable information when dealing with a *family* of perturbed MCs, as discussed in the introduction to section 3.

The paper is organized as follows. In section 2 a mathematical theory behind the algorithms is sketched. The next section contains algorithms themselves. The complexity analysis and several case studies can be found in section 3.5. We report the results of experiments we have performed—they are very promising and justify the applicability of the algorithms in practical problems.

2. Directed forests and MCs. This section is devoted to mathematical preliminaries crucial for aggregation algorithms considered in section 3. We discuss several known results, which allow us to express the solution of systems of linear equations by means of weighted directed forests (in the underlying graph). Having such *forests' expansion* of the characteristics under consideration (such as stationary distribution), we are looking for an effective procedure to compute it. To this end the concept of a *powerly perturbed MC* is used (see [33]), as in this case there exist effective recursive formulas enabling us to evaluate the forest expansions. This procedure yields an approximation of required characteristics.

2.1. Preliminaries. Let us define a set of *states*, $S = \{1, 2, \dots s\}$ and $E \subseteq S \times S$. Consider a *directed graph* $G = (S, E)$. State j is *reachable* from state i ($i \to j$) if there exists a sequence of edges of the graph leading from i to j. A *strong component* of G is any maximal subgraph C of G such that $i \to j$ for any two states $i, j \in C$. A strong component is *absorbing* if it has no outgoing edges. The strong absorbing components of G will be called *closed classes* in what follows.

An acyclic subgraph $f = (S, E_f)$ of G containing all of its vertices, in which any state has out-degree at most 1 is called a *directed spanning forest*. A set of states $R \subseteq S$, with no outgoing edges in E_f forms a *root* of a forest. When the root is singleton we talk about a *directed spanning tree*. For simplicity, we write shortly forest (tree) instead of directed spanning forest (tree).

Let $\mathcal{F}_G(R)$ denote the set of all forests in G having the root R (the forest could be identified with the set of its edges). We will omit the subscript G and write $\mathcal{F}(i_1, \dots i_m)$ instead of $\mathcal{F}(\{i_1, i_2, \dots i_m\})$. For fixed $i \notin R$ and $j \in R$, $\mathcal{F}_{ij}(R) \subseteq \mathcal{F}(R)$ denotes a set of forests, with the root R containing a path from i to j.

Let $A = (a_{ij})_{i,j=1}^{s}$ be a square matrix of size s with real elements. The *weighted graph* induced by A is the matrix A together with the graph $G(A) = (S, E)$, where $(i, j) \in E$ if and only if $a_{ij} \neq 0$. In $G(A)$ we define the (*multiplicative*) *weight* of a forest $f = (S, E_f)$ as

$$w(f) = \prod_{(i,j) \in E_f} (-a_{ij}),$$

and the weight of a set $\mathcal{F}$ of forests is defined by

$$w(\mathcal{F}) = \sum_{f \in \mathcal{F}} w(f)$$

(we set $w(\emptyset) = 0$).

A matrix $\mathbf{L} = (l_{ij})_{i,j=1}^{s}$, $l_{ij} \in \mathbb{C}$ satisfying $l_{ii} = -\sum_{j:\, j\neq i} l_{ij}$ for $i = 1, \ldots, s$ will be called a *Laplacian matrix*. Such matrices appear as a discretization of the Laplace operator.

Let $X = (X_t)_{t\geq 0}$ be an MC, with state space S. MCs are usually defined by a transition probability matrix $\mathbf{P} = (p_{ij})_{i,j=1}^{s}$ (when time is discrete) or by a generator $\mathbf{Q} = (q_{ij})_{i,j=1}^{s}$ (when time is continuous). Denote by $\mathbf{I}$ the identity matrix of size s. It is easily seen that matrices $\mathbf{L}(\mathbf{P}) = \mathbf{I} - \mathbf{P}$ and $\mathbf{L}(\mathbf{Q}) = -\mathbf{Q}$ are Laplacian matrices. Many facts we consider here are identical for discrete and continuous time (for details, we refer to [12, 18, 23]), and for simplicity we will identify finite MCs with *MC Laplacian matrices*, i.e., Laplacian matrices whose off–diagonal elements are nonnegative.

For $U, W \subseteq S$, and a square matrix $\mathbf{A}$ of size s, let us denote by $\mathbf{A}(U\big|W)$ the submatrix of $\mathbf{A}$ obtained after removing the rows and columns indexed by U and W, respectively. For the simplicity of notation we write $\mathbf{A}(U)$ instead of $\mathbf{A}(U\big|U)$ and $\mathbf{A}_{ij}$ instead of $\mathbf{A}(\{i\}\big|\{j\})$.

Let $\mathbf{e}_s$ and $\mathbf{0}_s$ denote column vectors, with each component equal to 1 and 0, respectively.

Many characteristics of MCs are described by the systems of linear equations in one of the following forms:

$$\mathbf{L}^T(R)\mathbf{x} = \mathbf{b}, \tag{2.1}$$

$$\mathbf{L}(R)\mathbf{x} = \mathbf{b}, \tag{2.2}$$

where $\mathbf{b}$ is a nonnegative vector of size $s - |R|$. As an example of (2.1), consider computing the *stationary distribution* defined as a nonnegative, normalized vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_s)^T$, which is solution of the following system:

$$\boldsymbol{\pi}^T\mathbf{L} = \mathbf{0}_s^T. \tag{2.3}$$

Another interesting characteristics of the MC, the *mean hitting time*, is an expected number of steps before reaching the set R if we start from state i. More formally, for $R \subseteq S$, $i \notin R$, we have

$$m_i(R) = \mathbf{E}(\min\{t \geq 0 : X_t \in R\}|X_0 = i).$$

It is easy to verify (cf. [18]) that

$$m_i(R) = 1 + \sum_{j\notin R} p_{ij} m_j(R),$$

and the matrix form we have an example of (2.2)

$$\mathbf{L}(R)\mathbf{m} = \mathbf{e},$$

where $(\mathbf{m} = (m_i(R))_{i\in S\setminus R})$.

2.2. Matrix tree theorem. We express the solution of a system of linear equations as a rational function of directed forest weights (called the *forest expansion*). For stationary distribution this is formulated in the MC tree theorem (Theorem 2.1). We also give analogous expansions for other characteristics. Unfortunately, expressions obtained could contain an exponential number of terms. In section 3.1 we present

recursive formulas yielding the effective procedures for evaluating forest expansions in the special case of powerly perturbed MCs.

Without loss of generality assume that the states are numbered in such a way that $R = \{s-|R|+1, \dots, s\}$; we assume that $R \neq \emptyset$. For a Laplacian matrix $\mathbf{L}$ of size $s \times s$, a set $R \subseteq S$ and $i, j \notin R$ consider forests in graph $G(\mathbf{L})$. The following facts are simple consequences of the famous matrix tree theorem proven in [11] and [4], respectively (see also [33]):

$$\det \mathbf{L}(R) = w(\mathcal{F}(R));$$
$$\text{if } w(\mathcal{F}(R)) \neq 0, \text{ then } \quad \mathbf{L}(R)^{-1} = \left[\frac{w(\mathcal{F}_{ij}(R \cup \{j\}))}{w(\mathcal{F}(R))}\right]_{i,j \in S \setminus R} \tag{2.4}$$

(we omit subscripts and write $\mathcal{F}(R)$ instead of $\mathcal{F}_{G(\mathbf{L})}(R)$, etc.). Now, consider the system of linear equations of the form (2.1):

$$\mathbf{A}^T \mathbf{x} = \mathbf{b} \tag{2.5}$$

for $\mathbf{A} = (a_{ij})_{i,j=1}^{s-1}$, $\mathbf{x} = (x_1, \dots, x_{s-1})^T$, $\mathbf{b} = (b_1, \dots, b_{s-1})^T$, and $\mathbf{x}, \mathbf{b} \in \mathbb{C}^{s-1}$. Set

$$\mathbf{L} = comp(\mathbf{A}, \mathbf{b}) = \begin{pmatrix} \mathbf{A} & \mathbf{l} \\ -\mathbf{b}^T & b \end{pmatrix}, \quad where \tag{2.6}$$

$$\mathbf{l} = \left(-\sum_{j=1}^{s-1} a_{1j}, \dots, -\sum_{j=1}^{s-1} a_{s-1,j} \right)^T \quad and \quad b = \sum_{j=1}^{s-1} b_j. \tag{2.7}$$

Recall that $\mathcal{F}(i)$, $i = 1, \dots, s$ denotes the set of spanning trees rooted in the state i in the graph $G(\mathbf{L})$. The following facts can be concluded from (2.4) and Cramer formulas:

(i) *The system $\mathbf{A}^T \mathbf{x} = \mathbf{b}$ has exactly one solution if and only if $w(\mathcal{F}(s)) \neq 0$;*

(ii) *if $w(\mathcal{F}(s)) \neq 0$, then the only solution of (2.5) is given by $x_i = w(\mathcal{F}(i))/w(\mathcal{F}(s))$.*

Consider again the problem of computing the stationary distribution $\boldsymbol{\pi} = (\pi_1, \dots, \pi_s)^T$ of an MC defined by the Laplacian matrix $\mathbf{L} = (l_{ij})_{i,j=1}^{s}$. Assume that an underlying graph of the MC has exactly one absorbing strong component, and the states are numbered in such a way that $\pi_s > 0$. Now, putting $\mathbf{A} = \mathbf{L}_{ss}$, $\mathbf{x} = (\frac{\pi_1}{\pi_s}, \dots, \frac{\pi_{s-1}}{\pi_s})^T$, and $\mathbf{b} = (-l_{s,1}, \dots, -l_{s,s-1})^T$ in (2.5) leads to the following theorem.

THEOREM 2.1 (MC tree theorem). *If the underlying graph of an MC has exactly one absorbing strong component, then the stationary distribution is given by*

$$\pi_i = \frac{w(\mathcal{F}(i))}{\sum_{j \in S} w(\mathcal{F}(j))} \qquad i = 1, \dots, s.$$

This was proven independently by many authors; among them [13, 34]. Analogously [33], one obtains a forest expansion for the mean hitting time for a set $R \subseteq S$:

$$m_i(R) = \frac{\sum_{j \notin R} w(\mathcal{F}_{ij}(R \cup \{j\}))}{w(\mathcal{F}(R))}.$$

Other characteristics widely studied in the literature such as: $\mu_{ij}(R) = \mathbf{E}_i[\sum_{0 \leq t < \tau_R} \mathbf{1}(X_t = j)]$—the mean number of visits before absorption, and $p_{ik}(R) = \mathbf{Pr}_i\{X_{\tau_R} = k\}$—the probability distribution in the hitting time of R can be com-

puted by solving systems of the form (2.2). For an MC determined by Laplacian matrix $\mathbf{L}$ such that graph $G(\mathbf{L})$ contains a spanning forest rooted in a fixed subset R of states, there exist (analogously to Theorem 2.1) the following forest expansions for $i, j \in S \setminus R$, and $k \in R$:

$$\mu_{ij}(R) = \frac{w(\mathcal{F}_{ij}(R \cup \{j\}))}{w(\mathcal{F}(R))},$$
$$p_{ik}(R) = \frac{w(\mathcal{F}_{ik}(R))}{w(\mathcal{F}(R))}.$$

In section 3 we present the combinatorial aggregation algorithms approximating vectors $\boldsymbol{\pi} = \pi_{ij}$ and $\mathbf{m} = m_{ij}(R)$. Analogous constructions for $\mathbf{p} = p_{ik}(R)$ and $\boldsymbol{\mu} = \mu_{ij}(R)$ are discussed.

The next section introduces the concept of powerly perturbed MCs—a wide class of MCs, for which there exists an effective way to approximate the solution of (2.1) or (2.2).

2.3. Powerly perturbed MCs. An NCD MC [6, 9, 38] is defined as a family of MCs $\{\mathbf{L}(\varepsilon),\ \varepsilon \in (0, \varepsilon_1)\}$ indexed by a small parameter $\varepsilon < \varepsilon_1$, for some constant $\varepsilon_1 > 0$:

$$\mathbf{L}(\varepsilon) = \begin{pmatrix} \mathbf{L}_1 & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_2 & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{L}_m \end{pmatrix} + \varepsilon \mathbf{L}',$$

where $\mathbf{L}_1, \mathbf{L}_2, \dots, \mathbf{L}_m$ are irreducible Laplacians of size $s_1, s_2, \dots, s_m$, respectively, and $\mathbf{L}'$ is a Laplacian of size $s = s_1 + s_2 + \cdots + s_m$.

In the literature one can find some generalizations of NCD MCs, aiming in expressing several different orders of the magnitude of interaction strength (see for example [15]):

1. The *linearly perturbed* MCs

$$\mathbf{L}(\varepsilon) = \mathbf{L}_0 + \varepsilon \mathbf{L}_1,$$

   where $\mathbf{L}_0$ and $\mathbf{L}_1$ are some (not necessarily block diagonal) Laplacian matrices;
2. The *polynomially or analytically perturbed* MCs

$$\mathbf{L}(\varepsilon) = \sum_{n=0}^{N} \varepsilon^n \mathbf{L}_n,$$

   where every $\mathbf{L}_n$ is a Laplacian matrix, $N \leq \infty$.

In [33] a wider class of perturbed MCs have been defined, subsuming both classes above. For given functions $A, B : \mathbb{R} \to \mathbb{R}$, the notation $A(\varepsilon) \sim B(\varepsilon)$ means that

$$\lim_{\varepsilon \to 0} \frac{A(\varepsilon)}{B(\varepsilon)} = 1.$$

We also set $A(\varepsilon) \sim 0$, if there exists $\varepsilon_1 \neq 0$ such that for any $\varepsilon \in (-\varepsilon_1, \varepsilon_1)$, $A(\varepsilon) = 0$.

DEFINITION 2.2. *A family* $\{\mathbf{L}(\varepsilon) = (l_{ij}(\varepsilon))_{i,j=1}^{s},\ \varepsilon \in (0, \varepsilon_1)\}$ *of Laplacian matrices of size* $s \times s$ *is a* powerly perturbed *MC, if there exist matrices* $\boldsymbol{\Delta} = (\delta_{ij})_{i,j \in S}$,

and $\mathbf{D} = (d_{ij})_{i,j\in S}$, $\delta_{ij} \geq 0$ *and* $d_{ij} \in \mathbb{R} \cup \{\infty\}$ *for* $i, j \in S$ *such that the asymptotic behavior of Laplacians* $\mathbf{L}(\varepsilon)$ *is determined by* $\mathbf{\Delta}$ *and* $\mathbf{D}$ *as follows:*

$$-l_{ij}(\varepsilon) \sim \delta_{ij}\varepsilon^{d_{ij}}. \tag{2.8}$$

We assume that $d_{ij} = \infty$ *if and only if* $\delta_{ij} = 0$.

DEFINITION 2.3. *A family* $\{\mathbf{b}(\varepsilon),\ \varepsilon \in (0, \varepsilon_1)\}$ *of nonnegative vectors of size* u *such that for some vectors* $\zeta = (\zeta_i)_{i=1}^u$ *and* $\mathbf{z} = (z_i)_{i=1}^u$, *with* $\zeta_i \geq 0$, $z_i \in \mathbb{R} \cup \{\infty\}$, *for* $i = 1, \ldots, u$, *the following holds:*

$$b_i(\varepsilon) \sim \zeta_i \varepsilon^{z_i} \tag{2.9}$$

is called a powerly perturbed *nonnegative vector.*

Consider the following graph induced by matrix $\mathbf{D}$ (we take into account asymptotically nonzero entries):

$$G^*(\mathbf{D}) = (S, \{(i,j) \in S \times S :\ d_{ij} < \infty\}).$$

For an arbitrary forest f and a set $\mathcal{F}$ of forests in $G^*(\mathbf{D})$ we study the following parameters:

(i) $\left\{ \begin{array}{l} d(f) = \sum\limits_{(i,j)\in f} d_{ij} \\ \delta(f) = \prod\limits_{(i,j)\in f} \delta_{ij} \end{array} \right\}$ an asymptotic weight of the forest f.

(ii) $\left\{ \begin{array}{l} d(\mathcal{F}) = \min\limits_{f\in\mathcal{F}} d(f) \\ \delta(\mathcal{F}) = \sum\limits_{f\in\mathcal{F}:\, d(f)=d(\mathcal{F})} \delta(f) \end{array} \right\}$ an asymptotic weight of the set of forests F.

Now, let $w(f)(\varepsilon)$ and $w(\mathcal{F})(\varepsilon)$ denote the weight of a forest f and a set $\mathcal{F}$ of forests in the graph $G(\mathbf{L}(\varepsilon))$ induced by $\mathbf{L}(\varepsilon)$, respectively. Observe that, for sufficiently small ε, $G(\mathbf{L}(\varepsilon)) = G^*(\mathbf{D})$. The following propositions are easy to prove.

PROPOSITION 2.4. *Consider a powerly perturbed MC defined by* $\mathbf{L}$, *with matrices* $\mathbf{\Delta}$ *and* $\mathbf{D}$ *such that* (2.8) *above holds; furthermore let* f *and* $\mathcal{F}$ *be a forest and a set of forests in* $G^*(\mathbf{D})$. *Then*

(i) $w(f)(\varepsilon) \sim \delta(f)\varepsilon^{d(f)}$;

(ii) $w(\mathcal{F})(\varepsilon) \sim \delta(\mathcal{F})\varepsilon^{d(\mathcal{F})}$.

We describe the asymptotics of the solutions of systems $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{A}^T\mathbf{x} = \mathbf{b}$, related to powerly perturbed MCs, in terms of directed forests expansions. It turns out that a solution of a system of linear equations, for a perturbed chain, can be treated as a perturbed vector.

THEOREM 2.5 (see [33]). *Let matrices* $\mathbf{\Delta}$ *and* $\mathbf{D}$ *be such that* (2.8) *above holds, for a powerly perturbed MC* $\{\mathbf{L}(\varepsilon), \varepsilon < \varepsilon_1\}$; *let* $R \subseteq S$, *where* S *is a set of states. Moreover, let vectors* ζ *and* $\mathbf{z}$ *of size* $u = s - |R|$ *be such that* (2.9) *holds, for a powerly perturbed vector* $\mathbf{b}$. *Suppose that there exist a forest with the root* R *in* $G^*(\mathbf{D})$. *Then the following hold:*

(1) *The solution* $\mathbf{x}(\varepsilon) = (x_i(\varepsilon))_{i\in S\setminus R}$ *of the system*

$$\mathbf{L}^T(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon)$$

satisfies for $i \in S \setminus R$ the relation

$$x_i(\varepsilon) \sim \eta_i \varepsilon^{h_i}.$$

(2) *The solution $\mathbf{x}(\varepsilon) = (x_i(\varepsilon))_{i \in S \setminus R}$ of the system*

$$\mathbf{L}(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon)$$

satisfies for $i \in S \setminus R$ the relation

$$x_i(\varepsilon) \sim \eta'_i \varepsilon^{h'_i},$$

where the coefficients η_i, h_i, η'_i, and h'_i are some constants, $i = 1, \ldots, u$.

This theorem can be seen as a generalization of the MC tree theorem in three respects. First, powerly perturbed MCs are considered. Second, a general class of problems is taken into account. And third, part (2) deals with nontransposed matrices.

From the proof of this theorem we can deduce the asymptotic forest expansions for MC characteristics in terms of parameters $d(\mathcal{F})$ and $\delta(\mathcal{F})$, for suitably defined sets $\mathcal{F}$ of forests in $G^*(comp(\mathbf{D}, \mathbf{z}))$. Both cases (1) and (2), although described similarly, differ substantially in difficulty—this will be visible later in section 3 in different structures of algorithms.

In the simpler case (1) of a transposed system for stationary probability distribution, we have

$$x_i(\varepsilon) \sim \eta_i \varepsilon^{h_i},$$

where

$$\begin{aligned} h_i &= d\left(\mathcal{F}(\{i\})\right) - \min_{j \in S} d\left(\mathcal{F}(\{j\})\right), \\ \eta_i &= \delta\left(\mathcal{F}(\{i\})\right) \Bigg/ \sum_{j:\, h_j = 0} \delta\left(\mathcal{F}(\{j\})\right). \end{aligned} \tag{2.10}$$

For the other case (2), more complicated formulas express the asymptotic coefficients η'_i and h'_i:

$$\begin{aligned} a_i &= \min_{j \in S \setminus R} [d(\mathcal{F}_{ij}(R \cup \{j\})) + z_j], \\ h'_i &= a_i - d(\mathcal{F}(R)), \\ \eta'_i &= \left(\sum_{j \in S \setminus R : d(\mathcal{F}_{ij}(R \cup \{j\})) + z_j = a_i} \delta(\mathcal{F}_{ij}(R \cup \{j\})) \cdot \zeta_j \right) \Big/ \delta(\mathcal{F}(R)). \end{aligned} \tag{2.11}$$

While in (2.10) it is sufficient to consider spanning trees rooted in some state i, in (2.11) it is necessary to take into account spanning forests rooted in $R \cup \{j\}$.

Unfortunately, all obtained expressions for asymptotic coefficients are computationally nontractable, at least directly, because of their exponential length. We discuss the aggregation approach, yielding effective and accurate procedures for computing the asymptotic coefficients and approximate values of the interesting characteristics of a powerly perturbed MC.

3. Combinatorial aggregation algorithms. We discuss in this section a combinatorial approach to compute the asymptotic coefficients (vectors $\mathbf{h}$, $\boldsymbol{\eta}$ and $\mathbf{h}'$, $\boldsymbol{\eta}'$ from Theorem 2.5) approximating the solution of systems $\mathbf{L}(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon)$ and $\mathbf{L}^T(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon)$. The algorithms reduce the size of the state space of a Markov chain by lumping together closely related states. This process is repeated in the consecutive phases of aggregation during which some graphs induced by matrices $\mathbf{D}$ and $\boldsymbol{\Delta}$ are considered. The algorithms group states of the MC belonging to the same closed class of the graph and solve the system of linear equations restricted to this class. A smaller size, hence tractable, system of equations can be solved by a direct method. The solution of this system is used to upgrade the values of an interested characteristic computed for each state of the original MC. Such an iterative upgrade scheme, enabling us to calculate effectively the asymptotic coefficients in the case of stationary distribution and mean hitting time, is discussed.

During the computation process one needs to solve small MCs corresponding to one aggregate (i.e., to solve the system of linear equations of the form $\mathbf{L}(R)\mathbf{x} = \mathbf{b}$ or $\mathbf{L}^T(R)\mathbf{x} = \mathbf{b}$).

If one has to solve the system with specific structure, which often is the case with MC problems, the standard LU decomposition is often not the best choice. The GTH method [14, 24, 33] seems to be the most suitable if the size of the system is moderate. Numerically, this algorithm, which is a specific variant of the Gaussian elimination, behaves very well: its numerical features are discussed in many papers; see, e.g., [5, 8, 16]. The GTH variant of Gaussian elimination can be also used to solve the system of the form (2.2). The only difference here is dealing with a nontransposed matrix; now the pivots are calculated as upper-diagonal row sums.

On the other hand, when the matrix $\mathbf{L}(R)$ is very large *and* sparse, in practice it usually is solved by an iterative method, such as block successive over-relaxation (BSOR) (see Algorithm 4) or the power method.

We consider two usage scenarios of powerly perturbed MCs. First, powerly perturbed MCs could be used to approximate the characteristics of MCs, which are given by a particular numerical Laplacian matrix. The approximations derived in this way could be further improved by, e.g., iterative methods. Such kind of problems are presented in sections 4.2 and 4.3.

Assume that our task is to compute the stationary distribution vector $\boldsymbol{\pi}$. The algorithm takes as an input a Laplacian $\mathbf{L} = (l_{ij})$ (which defines the MC). The computational process consists of following stages:

1. Analyze the structure of $\mathbf{L}$ to determine the parameter $\varepsilon > 0$ (for more details see section 4.4).
2. Construct a matrix $\mathbf{D} = (d_{ij})$ such that
$$d_{ij} = \min\{n : -l_{ij} > \varepsilon^n, n = 0, 1, \dots\}.$$
3. Construct a matrix $\boldsymbol{\Delta} = (\delta_{ij})$ such that
$$-l_{ij} = \delta_{ij}\varepsilon^{d_{ij}}.$$
Notice that d_{ij} and δ_{ij} are the analogs of exponent and significand (mantissa) in the floating-point representation, while ε corresponds to the base (radix) of the numeral system.
4. Run Algorithm 1 to compute vectors η and $\mathbf{h}$.
5. Set $\pi_i(\varepsilon) = \eta_i\varepsilon^{h_i}$.

ALGORITHM 1. Calculate asymptotic coefficient η and h, i.e., approximate the stationary distribution $\pi_i = \eta_i \varepsilon^{h_i}$.

construct $G^0 = (S^0, E^0)$
$k := 1$
repeat
  find partition of G^{k-1} into closed classes
  construct S^k
  for each closed class in S^k, say I^k **do**
    construct Laplacian L_k
    compute stationary distribution, i.e., solve the system $\mathbf{L}_k^T \mathbf{x} = \mathbf{b}$
    compute $h^*(I^k)$ (cf. (3.1) in section 3.1)
    for each aggregated state I^{k-1} in I^k **do**
      compute $\eta^k(I^{k-1}|I^k)$ and $h^k(I^{k-1}|I^k)$
      for each state i aggregated into state I^{k-1} **do**
        upgrade $\eta_i = \eta(i|I^k)$ and $h_i = h(i|I^k)$ according to (3.4)
      end for
    end for
    for all neighbors of class I^k **do**
      determine shortest edges
    end for
  end for
  construct new set of aggregated edges E^k
  $G^k := (S^k, E^k)$
  $k := k + 1$
until G^k has only one closed class
set $\pi_i := 0$, for all transient states i in G

We note that, if $\varepsilon < \min_{ij}\{-l_{ij}\}$, we have $d_{ij} = 0$ (for all i, j); hence $\mathbf{L} = \mathbf{\Delta}$, and Algorithm 1 gives the exact solution (in particular, $h_i = 0$ for all i). In that case, no aggregation can be done and the algorithm runs a direct method. On the other hand, larger ε makes it possible to exploit a specific block structure of the Laplacian matrix, which improves the efficiency. Hence, there exists a tradeoff between the time and space requirements of the algorithm and the precision of the approximation.

In the second scenario powerly perturbed MCs could be used to compute combined analytical and numerical approximations of a Markov model defined with a free, small parameter ε. In such a case we consider a family $\mathbf{L}(\varepsilon)$, given by matrices $\mathbf{D}$ and $\mathbf{\Delta}$, such that $-l_{ij} \sim \delta_{ij}\varepsilon^{d_{ij}}$. Like in the previous scenario, we calculate vectors $\boldsymbol{h}$ and $\boldsymbol{\eta}$ using a combinatorial aggregation approach and put $\pi_{ij} \sim \eta_{ij}\varepsilon^{h_{ij}}$. This approximation is analytical in flavor, because it depends *explicitly* on the parameter ε. However, coefficients η_i and h_i are computed numerically. A problem of this kind is discussed in section 4.5.

3.1. Fast computation of forest expansions. The task of computing exponents h_i is of quite a different nature than the task of computing the coefficients η_i. While the former can be performed using purely combinatorial methods, the latter uses a procedure of solving a system of linear equations, exposed to numerical errors. In what follows we concentrate on h_i and state some facts on *shortest forests*, useful in computing exponents. We explain how to calculate η_i in section 3.2.

Consider the graph $G = G^*(\mathbf{D})$ and its subgraph $G_{\min}$, consisting of the *shortest* edges outgoing from each vertex, i.e., for each vertex i, of those d_{ij} which are equal to

$$h^*(i) = \min_j d_{ij}. \tag{3.1}$$

Shortest edges correspond to the largest probability of moving from state i to j. Recall that $\mathbf{D} = \{d_{ij}\}$. In a single step of the aggregation process, the graph G is replaced by another graph $G' = aggr(G)$. Vertices of G' are closed classes I of $G_{\min}$ together with transient states in $G_{\min}$. Edges (I, J) in G' are weighted by d_{IJ} defined by the following formula:

$$\begin{aligned} d_{IJ} &= \min_{i\in I, j\in J}(d_{ij} + h(i|I)), \quad \text{where} \\ h(i|I) &= \max_{k\in I} h^*(k) - h^*(i). \end{aligned} \tag{3.2}$$

Values $h(i|I)$ are computed in Algorithm 1—they correspond to coefficients h_i in a graph induced by a closed class I. Proposition 3.1 below justifies such an aggregation scheme in order to calculate h_i—recall from (2.10) that to this aim we need $d(\mathcal{F}(i))$. From this fact (and from an accompanying fact for η_i) one derives the correctness of (3.3) in section 3.2. Let i denote any state of G such that there exists some tree rooted in i ($\mathcal{F}(i) \neq \emptyset$). By a *shortest tree* rooted in i we mean any tree rooted in i such that $d(f)$ is minimal, i.e.,

$$d(f) = \min_{f'\in\mathcal{F}(i)} d(f') = d(\mathcal{F}(i)).$$

PROPOSITION 3.1 (see [32]). *Let f be a shortest tree in G, rooted in i, and let I be a closed class in $G_{\min}$ containing i, i.e., $i \in I$. Let f_I be a shortest tree in the subgraph of $G_{\min}$ induced by I, rooted in i. Moreover, let f' be a shortest tree in G', rooted in I. The following holds (for simplicity, we apply here notation $d(\_)$ to graph G' as well):*

$$d(f) = d(f_I) + d(f').$$

For a nontransposed case, we need a similar fact for forests, related to (2.11).

PROPOSITION 3.2 (see [32]). *Let R be a subset of the states of G such that (s.t.) $\mathcal{F}(R)$ is nonempty. Let $I \neq J$ be closed classes or transient states in $G_{\min}$ s.t. R and $I \cup J$ are disjoint. Fix a state $i \in I$. Let f_j be a shortest tree in the subgraph of $G_{\min}$ induced by J, rooted in j. The following holds (similarly as before, we extend here notation $d(\_)$ and $\mathcal{F}\_(\_)$ to graph G'):*

$$\min_{j\in J} d(\mathcal{F}_{ij}(R \cup \{j\})) = \min_{j\in J} d(f_j) + d(\mathcal{F}_{IJ}(R \cup \{J\})).$$

We finish here the discussion about the mathematical background of algorithms—these are presented themselves in sections 3.2 and 3.3. Due to a lack of space, a lot of mathematical facts cited here are left without proof and for some others the argumentation is only sketched. For more details we refer the reader to [32]; see also [33].

Figure 3.1 illustrates the concept of shortest forests factorization from Propositions 3.1 and 3.2.

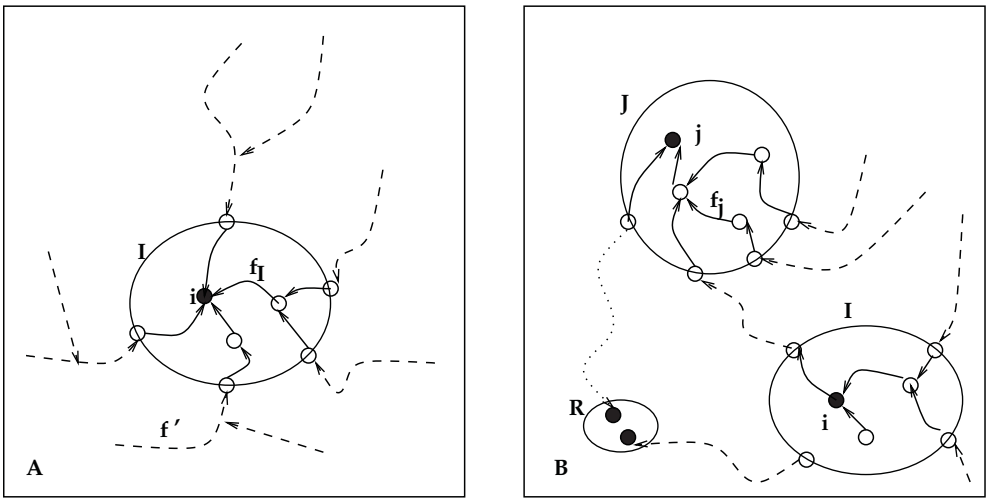

FIG. 3.1. A: *shortest tree rooted in* i *from Proposition* 3.1, B: *shortest forest rooted in* $R \cup j$ *from Proposition* 3.2.

3.2. Stationary distribution. In this section, we restrict our attention to the class of problems defined by a system of linear equations of the form

$$\mathbf{L}^T(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon).$$

As an illustrative example, we consider here the stationary distribution—analogous results for MC characteristics defined by the system with nontransposed $\mathbf{L}$ are discussed in section 3.3.

Recall that, for the powerly perturbed MC $\mathbf{L}(\varepsilon)$ with the state space S, the stationary probability vector $\boldsymbol{\pi}(\varepsilon)$ satisfies the relation

$$\pi_i(\varepsilon) \sim \eta_i \varepsilon^{h_i}, \text{ where}$$

$$h_i := d\left(\mathcal{F}(\{i\})\right) - \min_{j \in S} d\left(\mathcal{F}(\{j\})\right),$$
$$\eta_i := \delta\left(\mathcal{F}(\{i\})\right) \Big/ \sum_{j:\, h_j = 0} \delta\left(\mathcal{F}(\{j\})\right).$$

Aiming at computing the coefficients η_i and h_i effectively, consider the following aggregation process, which gives rise to the sequence of graphs $G^i = (S^i, E^i)$ for $i = 0, 1, \ldots$; starting from $i = 1$, the superscript i enumerates consecutive phases of the algorithm.

Initially, define the graph G^0 as $G_{\min}$, where $G = G^*(\mathbf{D})$. So, in the first step we start with the subgraph of $G^*(\mathbf{D})$ consisting of all shortest edges outgoing from every vertex. For $k = 1, 2, \ldots$, we define inductively $G'_k = aggr(G_{k-1})$. Recall that the states of G'_k are all closed classes and all transient states in G^{k-1}; the set of edges linking the aggregated states is constructed as in (3.2). Now, as a new graph G_k we take $(G'_k)_{\min}$, whose states are the same as in G'_k and whose edges are the shortest edges in G'_k.

Notice that the main loop ends precisely when the partitioning of G_{k-1} results in the only one closed class together with possibly some transient states.

From now on, we identify the aggregated state $I \in S^k$, with the set of states from S it contains. For a fixed state i, consider the family of closed classes (aggregated states):

$$\{i\} \subseteq I^1 \subseteq I^2 \subseteq \cdots \subseteq I^n = S \setminus T$$

(T denotes the subset of transient states) containing i during the consecutive phases of aggregation. We assume that the graph G^n, resulting from the nth phase, is irreducible. Denote by $\eta(i|I^k)$ and $h(i|I^k)$ coefficients η_i and h_i computed in the subgraph restricted to some closed class I^k. Following this convention, $\eta^k(I^{k-1}|I^k)$ and $h^k(I^{k-1}|I^k)$ correspond to the η and h coefficient for the aggregated state I^{k-1} computed during the kth phase for the subgraph of G^k restricted to a closed class I^k. The following recursive relation can be shown:

$$\pi_i(\varepsilon) \sim \eta^1(i|I^1)\varepsilon^{h^1(i|I^1)}\eta^2(I^1|I^2)\varepsilon^{h^2(I^1|I^2)} \cdot \ldots \cdot \eta^n(I^{n-1}|I^n)\varepsilon^{h^n(I^{n-1}|I^n)}\pi(I^n), \tag{3.3}$$

where $\pi(I^n) = 1$ is the stationary probability of being inside the class I^n. This relation holds due to the iterative upgrade scheme for asymptotic coefficients (Step 12), the correctness of which follows by Proposition 3.1:

$$\begin{aligned} h(i|I^k) &= h(i|I^{k-1}) + h^k(I^{k-1}|I^k), \\ \eta(i|I^k) &= \eta(i|I^{k-1})\eta^k(I^{k-1}|I^k). \end{aligned} \tag{3.4}$$

Coefficients $h^k(I^{k-1}|I^k)$ can be computed using the value of $h^*(I^k)$ (Step 10):

$$h^k(I^{k-1}|I^k) = \max_{I \subseteq I^k} h^*(I) - h^*(I^{k-1}).$$

So, we have only to consider all vertices I aggregated during the previous step, which belong to the closed class I^k.

We discuss the effective way to compute $\eta^k(I^{k-1}|I^k)$. Recall that we have assumed that our MC possesses specific block structure. Hence the size of all considered closed classes I^k are small compared with the size of the whole state space. It opens the possibility of using direct methods for solving systems of the form $\mathbf{L}^T\mathbf{x} = \mathbf{b}$, inside each class in parallel. From the solution (say $\mathbf{x} = (x_1, x_2, \ldots)$), having already computed h^k and putting some fixed ε_0, the corresponding coefficients η^k are derived as the solution of the equation:

$$\eta^k(I^{k-1}|I^k) = x_{I^{k-1}}\varepsilon_0^{-h^k(I^{k-1}|I^k)}.$$

3.3. Mean hitting time. Let us start with the main aggregation procedure (Procedure 2 below), which will be a basis for the mean hitting time computation, summarized in Algorithm 3 at the end of this section.

We describe here an approximation algorithm for another important characteristic of the MC—the mean hitting time. According to section 2.1, this is an interesting example of the linear equations with a nontransposed matrix. It is parametrized by a subset of states $R \subseteq \mathcal{S}$, and the task is to approximately calculate the expected number of steps before reaching this set. A formal definition can be found in section 2; recall that $\mathbf{m}_i(R)$ satisfies the following equality:

$$m_i(R) = 1 + \sum_{j \notin R} p_{ij} m_j(R).$$

PROCEDURE 2. Aggregation.

construct $G^0 = (S^0, E^0)$
$k := 1$
repeat
  find partition of G^{k-1} into closed classes (ignore edges leading to u^*)
  construct S^k
  for each closed class in S^k, say I^k, s.t. $I^k \neq u^R$ **do**
    construct Laplacian $\mathbf{L}_k$
    compute stationary distribution, i.e., solve the system $\mathbf{L}_k^T \mathbf{x} = \mathbf{b}$
    compute $h^*(I^k)$
    for each aggregated state I^{k-1} in I^k **do**
      compute $\eta^k(I^{k-1}|I^k)$ and $h^k(I^{k-1}|I^k)$
    end for
    for all neighbors of class I^k **do**
      determine shortest edges
    end for
  end for
  construct new set of aggregated edges E^k (update edges leading to u^*)
  $G^k := (S^k, E^k)$
  $k := k + 1$
until G^k has the same number of states as G_{k-1}

In the matrix form ($\mathbf{m} = (m_i(R))_{i \in S \setminus R}$):

$$\mathbf{L}(R)\mathbf{m} = \mathbf{e}, \tag{3.5}$$

where $\mathbf{e} = (1, 1, \ldots, 1)^T$. Hence, computing mean hitting time corresponds to approximating the solution of the system $\mathbf{L}(R)\mathbf{x} = \mathbf{e}$. The following forest expansion is given for the vector $\mathbf{m}_i(R)$:

$$m_i(R) = \frac{\sum_{j \notin R} w(\mathcal{F}_{ij}(R \cup \{j\}))}{w(\mathcal{F}(R))}.$$

In the case of a powerly perturbed MC it was proved [33] that

$$m_i(R)(\varepsilon) \sim \beta_{iR} \varepsilon^{b_{iR}}, \tag{3.6}$$

where asymptotic coefficient β and b are defined as follows:

$$a_i := \min_{j \notin R} [d(\mathcal{F}_{ij}(R \cup \{j\}))], \tag{3.7}$$

$$b_{iR} := a_i - d(\mathcal{F}(R)), \tag{3.8}$$

$$\beta_{iR} := \frac{\sum_{j \notin R : d(\mathcal{F}_{ij}(R \cup \{j\})) = a_i} \delta(\mathcal{F}_{ij}(R \cup \{j\}))}{\delta(\mathcal{F}(R))}. \tag{3.9}$$

The algorithm for the mean hitting time performs the aggregation process (Procedure 2) starting from a graph $G^0 = (S^0, E^0)$, which differs slightly from that in Algorithm 1. Namely, all states from R are replaced by a some single new state u_R; it has no outgoing edges and for any $i \notin R$, an edge from i to u_R is given by $\min_{j \in R} d_{ij}$ (this means that we are interested in reaching *any* of the states of R). Moreover, we

add one more state u^*, and we set $d_{iu^*} := 0$, $\delta_{iu^*} := 1$ for all $i \notin R \cup \{u_R\}$. Intuitively, this new state corresponds to the right-hand side vector e in (3.5), forming a fragment of the matrix $(comp(\mathbf{L}(R)^T, e))^T$. Notice that edges leading to u^* are ignored during the construction of a new set of aggregated states in Steps 4 and 5. However, they are updated in Step 17 according to the same rule as other edges (cf. formula (3.2) in section 3.1). Finally, if the graph constructed we now denote by G, as G_0, we take $G_{\min}$, similarly as previously.

Algorithm 3 consists of two phases. The first one runs the aggregation scheme, similarly as before; however, due to the artificial states u_R and u^*, being sinks in the graph, the aggregation can leave several closed classes of the original graph not lumped together. Then, the second one calculates coefficients b_{iR} and β_{iR} for these closed classes. Computed values are then propagated throughout each class.

ALGORITHM 3. Calculate asymptotic coefficients b_{iR} and β_{iR}, i.e., approximate the mean hitting time $m_i = \beta_{iR}\varepsilon^{b_{iR}}$.

1: **call** Aggregation (see Procedure 2)
2: **for** each closed class in S^k, say I^k, s.t. $I^k \neq u^R$ **do**
3: compute $b_{I^k R} := d_{I^k u^*} - h^*(I^k)$
4: **end for**
5: **repeat**
6: remove from S^k classes I^k having minimal $b_{I^k R}$ value, denote the set of removed classes by M
7: **for** each $I^k \in M$ and any state $i \in S^0$ belonging to I^k **do**
8: $b_{iR} := b_{I^k R}$
9: **end for**
10: **for** each I^k remaining in S^k **do**
11: let $n(I^k) := \min_{J^k \in M}(d_{I^k J^k} - h^*(I^k) + b_{J^k R})$
12: compute $b_{I^k R} := \min(b_{I^k R}, n(I^k))$
13: **end for**
14: **until** the set $S^k = \{u_R\}$
15: construct Laplacian $\mathbf{L}$, taking S^k as the set of states
16: solve the system $\mathbf{L}_{\{u_R\},\{u_R\}}\mathbf{x} = \mathbf{b}^*$, where $\mathbf{b}^*_{I^k}$ equals the probability of moving from I^k to u^* in G_k
17: **for** each $I^k \in S^k$ different than u_R and each state $i \in S^0$ belonging to I^k **do**
18: compute $\beta_{iR} := x_{I^k}\varepsilon_0^{-b_{I^k R}}$
19: **end for**

3.4. Improvements and extensions. The construction of Laplacian $\mathbf{L}$ in Step 3 of Algorithm 3 requires taking into account all edges between aggregated states, not only the shortest ones. Hence, coefficients β_{iR} computed by these algorithms do not coincide precisely with those needed in (3.6); fortunately, the computed values, say, β'_{iR}, are asymptotically equivalent:

$$m_i(R)(\varepsilon) \sim \beta'_{iR}\varepsilon^{b_{iR}}.$$

A similar modification of Algorithm 1 is also possible. The procedure solving a system of equations in a closed class (while calculating $\boldsymbol{\eta}$) can take into account all edges between vertices inside this class rather than only the shortest ones. After such a modification the algorithm computes no more the asymptotic coefficients—however,

it gives a closer approximation of stationary distribution. In certain situations it can significantly improve the accuracy of approximation.

Algorithm 3 can be easily applied to the calculation of other characteristics of MCs such as $\mathbf{p} = p_{ik}(R)$—the probability distribution (in R) in the hitting time and and $\boldsymbol{\mu} = \mu_{ij}(R)$—the mean number of visits before absorption.

3.5. Complexity issues. We estimate here the time and space complexity of Algorithms 1 and 3. The upper bound on computational complexity (later denoted for short by T) of Algorithms 1 and 3 presented before is $\mathcal{O}(n^3)$, where n is the number of states. The upper bound on memory needed (denoted by S) is $\mathcal{O}(n^2)$. However, the actual complexity of algorithms depends strongly on the structure of an MC under consideration. In this section we study in detail some important cases. The main conclusion is that if we can profit from a specific structure of a matrix (e.g., if we choose an appropriate ε), time $\mathcal{O}(n^2)$ is sufficient. Moreover, when a matrix is sparse, i.e., the number of edges m is significantly smaller that $\mathcal{O}(n^2)$, the algorithms use only space $\mathcal{O}(n+m)$. This is crucial, since matrices appearing in applications are often sparse, and it is not rare that $m = \Theta(n)$.

Consider a Laplacian with n states and m edges (i.e., m is the number of nonzero entries in the probability transition matrix). Assume that, in a step of the aggregation process, the underlying graph is divided into k closed classes of size $n_1, n_2, \ldots, n_k$, respectively (i.e., $n_1 + n_2 + \cdots + n_k = |\mathcal{S}|$, transient state are singleton closed classes).

THEOREM 3.3. *The computational complexity T and space requirements S of a single phase of aggregation (in both algorithms) are as follows:*

$$T = \mathcal{O}(n + m + \sum_{i=1}^{k} {n_i}^3),$$
$$S = \mathcal{O}(n + m + \max_{1 \le i \le k} {n_i}^2).$$

Proof. The computational cost of $n+m$ is due to Tarjan's algorithm for finding strongly connected components, used to determine closed classes. The construction of aggregated edges (i.e., edges in the graph used in the next phase) can be performed in time proportional to m. Finally, $\sum_{i=1}^{k} {n_i}^3$ is the upper bound of the cost of running a direct solver k times, for each closed class separately. For the space cost, it is determined by the size of the representation of a matrix together with at most quadratic space needed by a direct solver. ∎

The cost of the algorithm for stationary distribution is equal to the total cost of all aggregation phases. It is difficult to foresee, in general, the number of phases and the number of closed classes in each phase. This is why we study below some special cases—the aim is to demonstrate that the complexity is strongly dependent on the degree of aggregation. In the following let p denote the number of phases; when necessary, we use superscripts, like $n^{(i)}$, $k^{(i)}$, $m^{(i)}$, and ${n_j}^{(i)}$ to denote the number of states, etc., in phase i; $n^{(1)} = n, m^{(1)} = m$, etc. Below S_{total} and T_{total} denote the total computational and space cost of all aggregation phases, respectively.

1. $p = 1, k = 1$; a pessimistic case—all states are aggregated during the first step; a direct solver has to be used on the whole matrix; $T_{total} = \mathcal{O}(n^3), S_{total} = \mathcal{O}(n^2)$.
2. $p = n-1$, $k^{(i)} = n-i$; "lazy" aggregation—after step i there are still $n-i-1$ isolated states; $T_{total} = \mathcal{O}(n^2), S_{total} = \mathcal{O}(n+m)$.

3. The "ripple" aggregation: here we assume that $n = b \cdot c$; there are $p = b$ aggregation phases. In the first phase c states are aggregated into a closed class while other states are isolated; in each consecutive phase a group of c states is added to the closed class while remaining states are isolated. The computational complexity of the first phase is then

$$T = \mathcal{O}(n + m + (n - c + c^3)).$$

If we take $c = \Theta(\sqrt{n})$, then the cost of whole computation is $T_{total} = \mathcal{O}(n^2)$, $S_{total} = \mathcal{O}(n + m)$.

4. The "ideal" equilibrated aggregation, i.e., $k^{(i)} = \sqrt{n^{(i)}}$. Assume that $n^{(1)} = 2^{2^l}$ for some $l \geq 1$. We have then $p = l$, $k^{(i)} = n^{(i+1)} = \sqrt{n^{(i)}}$, ${n_j}^{(i)} = \sqrt{n^{(i)}}$, $j \leq \sqrt{n^{(i)}}$. It is an easy calculation to show that $T_{total} = \mathcal{O}(n^2), S_{total} = \mathcal{O}(n + m)$.

5. Consider an abstract algorithm consisting of several phases. The cost of each phase is polynomial w.r.t. the size of input data for this phase. Assume that the size of input data decreases at least twice in each phase. Then, it is easy to see that the cost of the algorithm is of the same order of magnitude as the cost of its first phase. Now, look at the generalized ideal aggregation: $k^{(i)} = (n^{(i)})^{1-\varepsilon}, {n_j}^{(i)} = (n^{(i)})^{\varepsilon}$, $j \leq k^{(i)}$, $n = n^{(1)} = 2^{\alpha^l}$, where $\alpha = \frac{1}{1-\varepsilon}$ and $0 < \varepsilon < 1$. The size of problem in the ith phase of aggregation satisfies the inequality

$$n^{(1-\varepsilon)^i} < \frac{n}{2^i};$$

hence the cost of whole computation is the same as a cost of the first phase:

$$T_{total} = \mathcal{O}(m + n^{(1+2\varepsilon)}).$$

For the mean hitting time, the complexity must be increased by $\mathcal{O}(k^3)$, where $k = k^{(p)} = n^{(p)}$ is the number of closed classes when the aggregation processes stops; the space requirements are increased by $\mathcal{O}(k^2)$.

4. Numerical experiments and case studies. In this section we describe the results obtained when combinatorial aggregation algorithms are used to compute stationary distribution and mean hitting time of several Markov models. We compare our combinatorial aggregation method with a direct GTH method in the case of small size examples and with the BSOR in the case of large models. The latter was chosen because it is widely accepted as the best known method calculating the approximate solution of problems related to large and sparse MCs (see for example [10, 31]). Two problems investigated in sections 4.2 and 4.3 are real-life examples and had been extensively studied by many authors (see, e.g., [10, 8, 41]). We conclude with section 4.5, which discusses the pros and cons of applying the aggregation algorithm to a Google matrix problem [25].

For each small example considered in what follows we compute the relative error of the solution (denoted by $\boldsymbol{\pi}^*$ and $\mathbf{m}^*$) compared with the outcome of the GTH procedure (vectors: $\boldsymbol{\pi}$ and $\mathbf{m}$, respectively).

Two other measures of algorithms' accuracy correspond to a mean number of correct most significant digits and are given by the following formulas (n is the size

of the matrix):

$$\mathbf{Prec}_1 := -\frac{1}{n}\sum_{i=1}^{n}\log_{10}\frac{|\pi_i^* - \pi_i|}{|\pi_i|} + \log_{10} 5,$$
$$\mathbf{Prec}_2 := -\log_{10}\frac{||\boldsymbol{\pi}^* - \boldsymbol{\pi}||_2}{||\boldsymbol{\pi}||_2} + \log_{10} 5.$$

In the case of combinatorial aggregation we discuss also the aggregation process: the value of ε, the number of phases, and the number of aggregated states in each phase.

Another part of the tests was to verify if the aggregation produces a better (than the commonly used uniform distribution) starting vector for a regular iterative solver, such as BSOR or the power method. It should be noted that, while we always used in the experiments a full-matrix GTH solver, in real computations it would be profitable to replace it with a sparse solver, e.g., a couple of iterations of an iterative method. The rationale for doing this is that, since the resulting vector is only an approximation, it is not necessary to solve the subproblems to full available accuracy.

The experiments show several advantages of our algorithms over existing methods:

- For the first time the aggregation approach is successful in computing the MC characteristics which are solutions of nontransposed systems of equations, e.g., the mean hitting time.
- The regular NCD structure[1] is not obligatory—the algorithms deal also with *asymptotically transient states*.
- A comparison with the GTH procedure shows that the precision of approximation computed by our algorithms is on the level of $\log \frac{1}{\varepsilon}$ for the prespecified parameter ε.
- A very promising application of our algorithms is to use the approximate solution yielded by them as a starting point for some iterative methods, e.g., block successive over-relaxation.

4.1. Block successive over-relaxation (BSOR). The BSOR method belongs to the class of stationary iterative methods which can be expressed in the simple form [10]:

$$x^{(k+1)} = \mathbf{A}x^{(k)} + c, \qquad k = 0, 1, \ldots,$$

where neither $\mathbf{A}$ nor c depend on the iteration step k. In particular, the BSOR procedure (see Algorithm 4) is parametrized by the following:

- starting solution vector;
- relaxation parameter ω $(0 < \omega < 2)$;
- block partitioning of the matrix;
- stop criterion.

Algorithm 4 assumes the partitioning of the Laplacian matrix into N blocks; ith diagonal block denoted by $\mathbf{L}_{ii}$ is of size n_i. In what follows we consider five block partitioning strategies:

1. **SCC**. We are looking for strongly connected components in the underlying graph of the MC. Edges weighted with probability less than ε (prespecified decomposability parameter) are ignored. This partitioning coincides with one resulting from the near-decomposability test of the Markov Chain Analyzer (MARCA) [40].

[1] for a formal definition see conditions 6.1–6.4 in [38] pp. 335.

ALGORITHM 4. BSOR method for solving the system $\mathbf{L}^T(R|R)\boldsymbol{\pi} = \mathbf{0}$.

repeat
for $i = 1$ to N **do**

$$\mathbf{z}_i^{(k+1)} := (1-\omega)\mathbf{L}_{ii}^T\mathbf{x}_i^{(k)} - \omega(\sum_{j=1}^{i-1}\mathbf{L}_{ji}^T\mathbf{x}_j^{(k+1)} + \sum_{j=i+1}^{N}\mathbf{L}_{ji}^T\mathbf{x}_j^{(k)})$$

Solve (e.g., using GTH method) system of equations:

$$\mathbf{L}_{ii}^T\mathbf{x}_i^{(k+1)} = \mathbf{z}_i^{(k+1)}$$

end for
normalize vector $\mathbf{x} := (\mathbf{x}_1^T, \ldots, \mathbf{x}_N^T)$, where $\mathbf{x}_i^T = (x_{i1}, x_{i2}, \ldots, x_{in_i})^T$:

$$\pi_{ij} := \frac{x_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{n_i} x_{ij}}$$

until stop criterion succeeds

2. **CC**. Consider a graph obtained from underlying graph of the chain by replacing each directed edge having probability greater than ε by an undirected one. Blocks for partitioning are then the connected components of this graph.
3. **CC***. As before partitioning is induced by connected components, but additionally all singletons are grouped into a single component. This strategy was used in [10, 7].
4. **Aggr**. We take a partitioning into closed classes computed during the first phase of combinatorial aggregation. Recall that in the aggregation process the subgraph of shortest edges leaving each state is considered.
5. **Asymp**. The complete aggregation process which approximates the stationary distribution induces a partition of states into blocks according to the values of asymptotic coefficients: each block groups states with the same value of h_i. Such a partitioning is illustrated in Figure 4.4 for a two-dimensional (2D) MC model.

4.2. Interactive computer system. To illustrate the applications of MCs let us consider a simple model of an interactive computer system (ICS). Figure 4.1 represents the architecture of a time-shared, paged, virtual memory computer. This model was widely studied in the literature [8, 38]. The system consists of the following:

- a set of terminals from which users generate commands;
- a central processing unit (CPU);
- a secondary memory device (SM);
- an I/O device (I/O).

A queue of requests is associated with each device, and the scheduling is assumed to be first-come first-served. When the command is generated, the user at the terminal remains inactive until the system responds. Symbolically, a user having generated a command enters the CPU queue. The behavior of the process in the system is characterized by a computing time followed by a page fault, after which the process

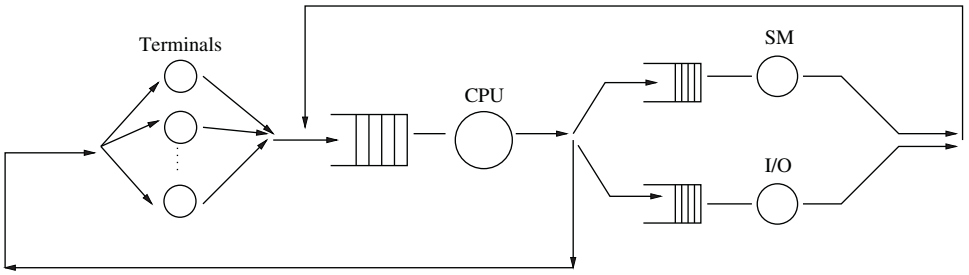

FIG. 4.1. *An illustration for interactive computer system.*

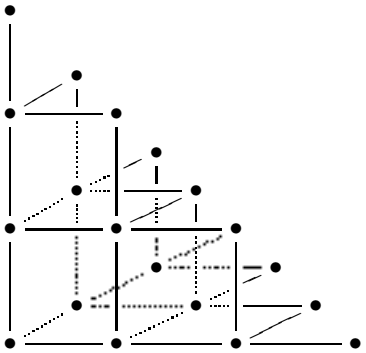

FIG. 4.2. *The three-dimensional structure of an MC model of interactive computer system.*

enters the SM queue, or an input/output (file request), in which case the process enters the I/O queue. Processes which terminate their service at a SM or I/O queue return to the CPU queue. The completion of a command is represented by a departure of the process from the CPU to the terminals.

The states of the MC corresponding to our model are determined by the numbers of processes in all queues. The state space is large but sparse in the sense of connections; there are maximum 6 transitions going out from any state. Figure 4.2 illustrates the structure of an MC (for 3 users in the system): three coordinates correspond to the number of processes in three queues; for example, state (0,0,3) denotes three tasks waiting in I/O queue. It turns out that the model for only 20 users yields the transition probability matrix of order $1,771$, with $11,011$ nonzero elements, and for 50 users the matrix defining the MC is of order $23,426$, with $156,026$ nonzero elements.

In order to perform numerical experiments we assign specific values for the parameters of the model such as

- the number of users in the system $N = 3, 10, 20, 50$;
- *Belady–Kuehner lifetime function* $q(\alpha, M, k)$, where M denotes the size of primary memory and α, k are some constants which depend on program characteristics and memory management strategy [37];
- the mean thinking time of a user at the terminal $\lambda^{-1} = 10s$;
- the mean service time of a SM $u_1^{-1} = 5ms$;
- the mean service time of an I/O device $u_2^{-1} = 20ms$;

- the mean compute time I/O requests $r = 20ms$;
- the mean compute time of the process $c = 500ms$.

We fix all parameters, except the number of users in the system (N), according to [37]. Recall that the state of the system is coded by a triple $\mathbf{z} = (z_1, z_2, z_3)$ of nonnegative numbers, where z_1 denotes the number of users thinking or busy at their terminals, z_2 and z_3 denote, respectively, the number of processes in the queue of SM and I/O. Obviously $z_1 + z_2 + z_3 \leq N$. There are at most six transitions which can be made from any state, i.e., from (z_1, z_2, z_3) to states:

$(z_1 - 1, z_2, z_3)$, with rate $\lambda z_1 = 0.0001 z_1$;
$(z_1 + 1, z_2, z_3)$, with rate $c^{-1} = 0.002$;
$(z_1, z_2 - 1, z_3)$, with rate $u_1 = 0.2$;
$(z_1, z_2 + 1, z_3)$, with rate $q(z_1); = \alpha^{-1}(\frac{N-z_1}{M})^K = 100(\frac{(N-z_1)}{128})^{\frac{3}{2}}$;
$(z_1, z_2, z_3 - 1)$, with rate $u_2 = 0.05$;
$(z_1, z_2, z_3 + 1)$, with rate $r^{-1} = 0.05$.

We study the behavior of the algorithm approximating mean hitting time and stationary distribution for 3, 10, and 20 users. In each case we analyze the number of phases in the aggregation process (denoted by p), the number of aggregates in the first step (k), and the precision measures. In the array, n states for the size of the matrix, and m is the number of its nonzero entries. Parameters T_{agr} and T_{gth} correspond to the time cost of aggregation algorithm and GTH procedure, respectively. Set $R = \{0, 0, 0\}$, i.e., all processes are in the CPU queue as follows:

N	n	m	$\mathrm{Prec}_1(\mathbf{m})$	$\mathrm{Prec}_2(\mathbf{m})$	p	k	T_{agr}	T_{gth}
3	20	60	4.16	3.87	1	14	$0.01s$	$0.003s$
10	286	1320	4.21	4.21	1	77	$0.97s$	$25s$
20	1.771	11.011	4.97	4.97	1	252	$89s$	$> 8h$

The results for the stationary distribution are summarized in the following array:

N	n	m	$\mathrm{Prec}_1(\boldsymbol{\pi})$	$\mathrm{Prec}_2(\boldsymbol{\pi})$	p	k	T_{agr}	T_{gth}
3	20	60	3.4	4.49	2	4	$0.02s$	$0.003s$
10	286	1320	2.97	3.7	2	11	$0.37s$	$24s$
20	1.771	11.011	3.16	6.0	2	21	$30s$	$> 8h$

We conclude that combinatorial aggregation in the case of a nontransposed system of equations (mean hitting time) is not less effective than in the case of stationary distribution. Up to now the aggregation approach was used only in solving problems like stationary distribution. Combinatorial aggregation algorithms allow us to treat both type of problems in a unified manner.

For an ICS model with 50 users we perform several experiments with the iterative method. The solution computed by combinatorial aggregation is used as a starting vector for the BSOR algorithm (the relaxation parameter $\omega = 1.0$ as in [10] Table C.7). The speed of convergence, measured in the number of iterations, is compared with BSOR starting from the uniform distribution. We investigate three block partitionings:

1. **SCC** for $\varepsilon = 0.0002$, which gives the same partition as **Aggr** into 51 strongly connected components.
2. **CC*** for $\varepsilon = 0.1$ yields 1221 blocks (connected components).
3. **CC** for $\varepsilon = 0.003$ results in 1326 blocks.

TABLE 4.1
Numerical results for an ICS example.

Starting $\boldsymbol{\pi}$	Partition	ICS example ($N = 50$)			
		# bl.	# it.	$\|\|\Delta\boldsymbol{\pi}\|\|_\infty$	$\|\|\boldsymbol{\pi}^T\mathbf{L}\|\|_\infty$
Uniform	**SCC=Aggr**	51	22	$0.45e-10$	$0.59e-16$
	CC*	1221	161	$0.92e-10$	$0.16e-14$
	CC	1326	252	$0.99e-10$	$0.26e-14$
Aggregation	**SCC=Aggr**	51	4	$0.97e-10$	$0.69e-16$
	CC*	1221	48	$0.95e-10$	$0.17e-14$
	CC	1326	48	$0.95e-10$	$0.17e-14$

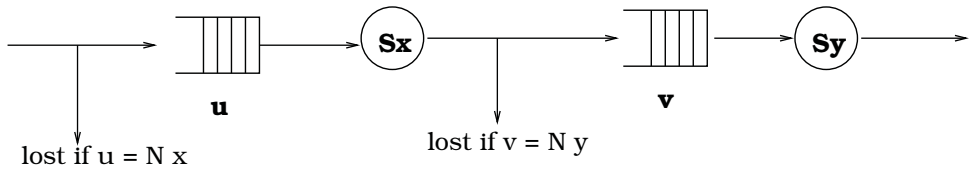

FIG. 4.3. *Telecommunication model for a 2D example.*

The stopping criterion we use in BSOR is $||\boldsymbol{\pi}^{(k)} - \boldsymbol{\pi}^{(k-1)}||_\infty \leq$ *stop_tol*, where stopping tolerance *stop_tol* is set to 10^{-10}. In Table 4.1, $||\Delta\boldsymbol{\pi}||_\infty$ is the infinity norm of the difference between the last two iterates, and $||\boldsymbol{\pi}^T\mathbf{L}||_\infty$ is the true residual upon termination.

Notice (cf. Table 4.1) that starting from an approximate solution computed by combinatorial aggregation allows us to reduce the number of iterations about $4-5$ times. In some cases, e.g., for **SCC** partition the time cost of Algorithm 1 is comparable with the cost of the first iteration of BSOR. One could have additionally accelerated the combinatorial aggregation by using an iterative method instead of a direct solver for solving subproblems inside closed classes.

4.3. A 2D MC model. We consider here a 2D MC, studied, e.g., in [10, 40]. It is a simple telecommunication model illustrated in Figure 4.3. There are two servers; each has a queue of waiting tasks of prespecified maximum size. A task arrives, waits for the first server, then waits for the second and finally leaves the network.

The states are pairs (u, v), where u ranges from 0 through N_x and v ranges from 0 through N_y. States correspond to the size of two queues. In this model we assume transitions to the south, east and northwest. From any nonboundary state (u, v) there are the following values assigned to the transitions:

$(u, v-1)$ (south),	with rate	v	task leaves the network;
$(u+1, v)$ (east),	with rate	2025.0	task arrives to first queue;
$(u-1, v+1)$ (northwest),	with rate	u	task enters second queue.

The state space of the Markov chain is of size $(N_x+1)(N_y+1)$. In larger experiments the values of N_x and N_y are both set to 128, yielding a matrix of order $16,641$, with $66,049$ nonzero elements. For this model we perform several experiments (the relaxation parameter ω in BSOR varies according to the partitioning scheme, cf. Table D.4 in [10]):

1. The approximate solution is calculated using a combinatorial aggregation algorithm for $\varepsilon = 0.06$.
2. The solution is computed by BSOR procedure with **CC** partitioning ($\varepsilon = 0.06$) and **Asymp**. We start from the uniform distribution.

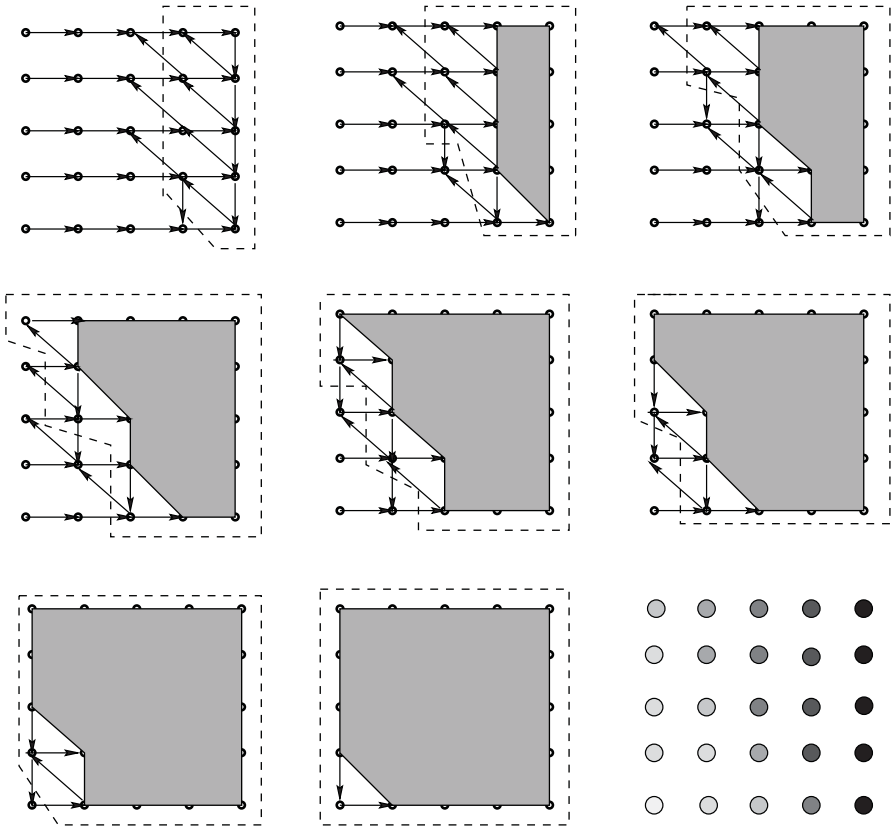

FIG. 4.4. *Aggregation in a* $2D$ *example.*

3. The solution is computed using BSOR starting from an approximation obtained by combinatorial aggregation; the same partitionings are considered.

Figure 4.4 illustrates the process of aggregation in combinatorial aggregation algorithms for $N_x = N_y = 4$. There are 8 phases of aggregation, each is shown in a separate figure. We see a subgraph of shortest edges $G_{\min}$ in every phase; a dashed line surrounds the only nonsingleton closed class appearing in that phase. In the last figure the states are colored according to the values of their asymptotic coefficients yielding **Asymp** partition for BSOR. These correspond to consecutive aggregation phases, i.e., those states which are aggregated earlier have bigger stationary probability. Despite that during every step there is only one nonsingleton closed class, i.e., the aggregation does not proceed in parallel, the time needed for the whole computation in only $\mathcal{O}(n^2)$ (cf. "ripple" aggregation).

The numerical results are summarized in Table 4.2. We observe about a 25% speedup of the BSOR method started from an approximate solution w.r.t. BSOR started from the uniform solution. **CC** partition behaves better than **Asymp**, but in the other hand the profit from choosing a starting vector is greater in **Asymp**.

4.4. The choice of an appropriate ε parameter. The efficiency of our approximation algorithms depends strongly on the value of the parameter ε, which determines the number of phases of the algorithm and the structure of the aggregation process in each phase. It seems to be a challenging task to design a fully automatic

TABLE 4.2
Numerical results for a 2D example.

Starting $\boldsymbol{\pi}$	Partition	2D example ($N_x = N_y = 64$)			
		# bl.	# it.	$\|\|\Delta\boldsymbol{\pi}\|\|_\infty$	$\|\|\boldsymbol{\pi}^T\mathbf{L}\|\|_\infty$
Uniform	**CC**	65	505	$0.96e-10$	$0.25e-11$
	Asymp	129	595	$0.98e-10$	$0.31e-11$
Aggregation	**CC**	65	410	$0.98e-10$	$0.25e-11$
	Asymp	129	414	$0.97e-10$	$0.31e-11$
Starting $\boldsymbol{\pi}$	partition	2D example ($N_x = N_y = 128$)			
		# bl.	# it.	$\|\|\Delta\boldsymbol{\pi}\|\|_\infty$	$\|\|\boldsymbol{\pi}^T\mathbf{L}\|\|_\infty$
Uniform	**CC**	129	1030	$0.99e-10$	$0.51e-11$
	Asymp	257	1212	$0.99e-10$	$0.59e-11$
Aggregation	**CC**	129	868	$0.99e-10$	$0.52e-11$
	Asymp	257	876	$0.99e-10$	$0.58e-11$

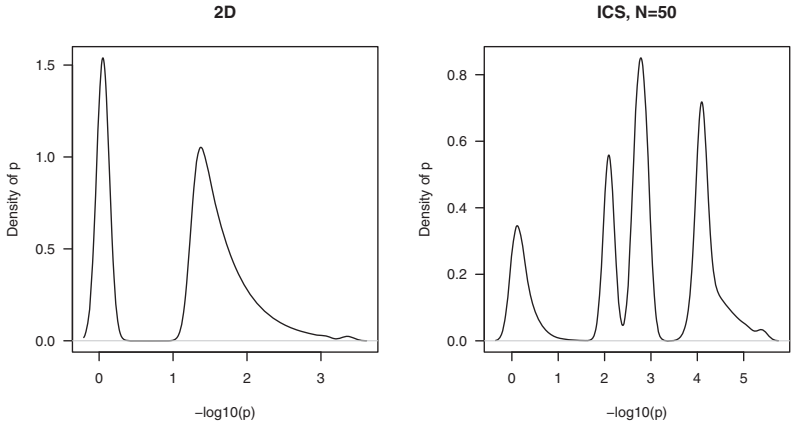

FIG. 4.5. *The distribution of orders of magnitude for Laplacian elements.*

procedure which would return a value for ε, with a guaranteed accuracy level and a time/space effectiveness of aggregation.

However, in many applications it is relatively easy to determine the value of ε in a semiautomatic manner as follows. We begin with the analysis of the elements of the Laplacian. In Figure 4.5, the distribution of $\{-\log(l_{ij})\}_{i\neq j}$ for selected MCs considered in the previous sections is plotted. It allows us to observe the different orders of magnitude of Laplacian elements. The Laplacian values were normalized such that $\max_i \sum_{j\neq i}(-l_{ij}) = 1$. In both plots the partition of Laplacian elements into several groups is clearly visible. For the 2D model we have two groups corresponding to different orders of magnitude. We can therefore select ε in such a way that elements from the first group have exponent 0 and elements from the second group have exponent 1 in the numeral system of radix ε (cf. section 3). On the other hand, for the interactive computer system model, a plot of density $\{-\log(l_{ij})\}_{i\neq j}$ is not so instructive: we can distinguish 2 groups as in the previous example or 4 groups corresponding to the local maxima of the density function.

4.5. Markov model of the World Wide Web. The problem of the efficient computation of the PageRank vector [30], also known as the Google matrix problem [25], has recently brought the attention of several scientists, e.g., [17, 20, 21, 22, 26, 36].

The matrix $\mathbf{L}$ of the Google problem is very large, sparse, and highly unstructured. It also has several interesting mathematical properties, which can be exploited in the algorithms, see e.g., [22]. The aim of our tests was to verify whether the combinatorial aggregation yields a better (than usually used uniform distribution) starting vector for the power method to compute the PageRank vector.

Running tests on a "true" Google matrix was far beyond our capabilities. Instead, we considered the problem of finding a PageRank vector for a collection of web pages hosted by some portal. We tried to obtain the link structure for pages reachable from the home page of the portal, dropping any links leading to or from pages outside the portal. For efficiency reasons, we stopped the process after reaching a prescribed number of portal pages.

In order to make the model realistic, we supplemented the collection of pages hosted by the portal with an artificial web page, called "the world." Our rationale was to take into account the actual behavior of the web surfer, who usually does not spend all the time browsing through the inner pages of the portal, but rather, after some time, leaves it and goes to another portal. In consequence, we included an out-link to the world outside on each portal page and decided that the only in-link from the world outside leads exactly to the home page of the portal. We found it reasonable to specify a free (and small) parameter $\varepsilon \in (0, 1)$ to measure the probability of going from any page in the portal to the outside world, assuming that in real life, the web surfer leaves the portal very quickly. For a given web page on the portal, containing n links to other portal pages, we set the probability of going from this page to another equal to ε/n. This approach corresponds to a very specific, nonuniform "personalization vector" of the original Google matrix [25]. Note that such an approach removes the common problems of web-rings [30] and dangling pages [21]. Let us also remark that, for similar reasons, the original Google matrix problem also contains a parameter-dependent modification, where the genuine link-based transition matrix $\mathbf{L}$ is replaced with $(1-\varepsilon)\mathbf{L} + \varepsilon E$, where E denotes a (usually uniform) teleportation matrix [21].

Ordering the states so that the first state corresponds to the world, and the second state is the home page of the portal we have, using the notation of Algorithm 1, that $h_1 = 1$ and $h_2 = 1$. Other coefficients of the stationary vector are expected to be extremely small. It is interesting that the exact, easy interpretable formula for coefficients h_i exists. Indeed, for a given page i hosted by the portal, we have $h_i = 1 + \mathrm{dist}(\mathrm{home}, i)$, where $\mathrm{dist}(\mathrm{home}, i)$ is the length of the shortest path in the link graph from the home page to i. Then, in the limiting case $\varepsilon = 0$ the stationary state π is simply the unit vector.

Therefore, in some sense, the problem of finding the stationary vector π becomes numerically difficult: most of the coordinates in π may be numerically irrelevant relative to the first two coordinates of π. Moreover, the usual stopping criterion $||\pi^{(k+1)} - \pi^{(k)}|| \leq \mathit{stop\_tol}$ turns out to be essentially useless for the same reason; some of the smallest coefficients may also suffer numerical underflow. This is where our method shows an advantage over the straightforward numerics, because the former is capable of ranking the pages even for any small ε. Indeed, having computed for the ith and the jth state values of h_i, η_i and h_j, η_j (see Algorithm 1), we are able to decide that the ith state has a higher rank than the jth state if $h_i < h_j$, or, in the case when $h_i = h_j$, when $\eta_i > \eta_j$. (An identical approach could have been, in principle, applied also in the case of the augmented Google matrix $(1-\varepsilon)\mathbf{L} + \varepsilon E$, with ε treated as a free and small parameter.)

Using a substantially modified MATLAB code originating from [28], we obtained the inner link structure (as of January, 2006) of several faculties' portals. In what

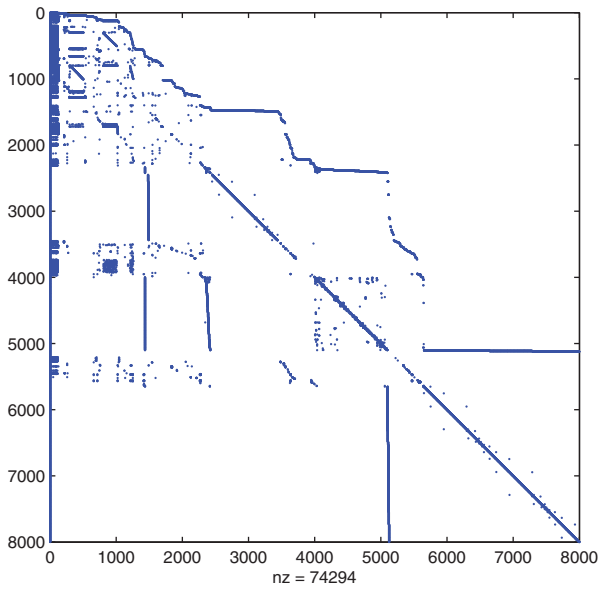

FIG. 4.6. *The link structure of the first 8000 pages hosted at http://www.mimuw.edu.pl.*

follows we shall focus on our faculty's portal `http://www.mimuw.edu.pl`, including 8000 pages (see Figure 4.6).

The aggregation went on `http://www.mimuw.edu.pl` smoothly and finished after 8 stages, with the largest subclass counting 2657 states. Next, for various values of ε, we computed the PageRank vector π with the power method. For the starting vector, we used either the uniformly distributed vector $\frac{1}{8001}[1, 1, \ldots, 1]^T$ (denoted "uniform" in the plots and tables) or the approximation obtained with the aggregation method. For the latter choice, we compared two starting vectors: "$aggr(\epsilon)$," which was the aggregation-based starting vector calculated by the formula from section 3, with the actual value of ϵ, and a starting vector corresponding to an arbitrarily chosen value $aggr(0.5)$. The stopping criterion was, as previously, $||\pi^{(k)} - \pi^{(k-1)}||_\infty \leq stop\_tol$. The results are summarized in Table 4.3 for $stop\_tol = 10^{-5}$, which reportedly is the case encountered by Google [25].

Regardless of the value of ε, the aggregation-based vector always provided a better alternative to the uniform starting vector, and the number of the iterations was essentially identical for both $aggr(\varepsilon)$ and $aggr(0.5)$. The number of power method iterations required for $\varepsilon \approx 1$ was large, so it was quite surprising that the aggregation-based starting vector turned out robust even in this case and actually was superior to the uniform initial distribution, as shown in Table 4.3.

We also compared the stationary vectors obtained by the means of the power method, on one hand, and computed directly by using the asymptotic formula $\pi_i(\varepsilon) = \eta_i \varepsilon^{h_i}$ on the other hand, with η and $\mathbf{h}$ obtained from the aggregation procedure. Since the accurate solution was unavailable, instead of validating the correctness of the two rankings, we sought only for differences between them. As depicted in Figure 4.7, the rank vectors computed by the two methods differed significantly, despite both methods producing similar residual norms; cf. Table 4.3. A closer examination of the two vectors reveals that not only the ranges of the rank values are different, but also the generated page rankings.

TABLE 4.3
The performance of the power method for `http://www.mimuw.edu.pl`. *Stopping criterion* stop_tol $= 10^{-5}$.

ε	Starting vector	# it.	$\|\|\pi^T L\|\|_\infty$
0.00001	uniform	3	2.28e-11
0.00001	$aggr(0.5)$	3	1.26e-11
0.00001	$aggr(0.00001)$	1	1.08e-11
0.001	uniform	3	2.28e-11
0.001	$aggr(0.5)$	3	1.26e-11
0.001	$aggr(0.001)$	1	1.08e-11
0.1	uniform	5	4.96e-07
0.1	$aggr(0.5)$	4	7.72e-07
0.1	$aggr(0.1)$	3	2.13e-08
0.5	uniform	14	3.37e-06
0.5	$aggr(0.5)$	9	2.03e-06
0.9	uniform	69	8.59e-06
0.9	$aggr(0.5)$	30	5.93e-06
0.9	$aggr(0.9)$	32	7.90e-06
0.999	uniform	2513	9.94e-06
0.999	$aggr(0.5)$	64	9.58e-06
0.999	$aggr(0.999)$	68	9.48e-06

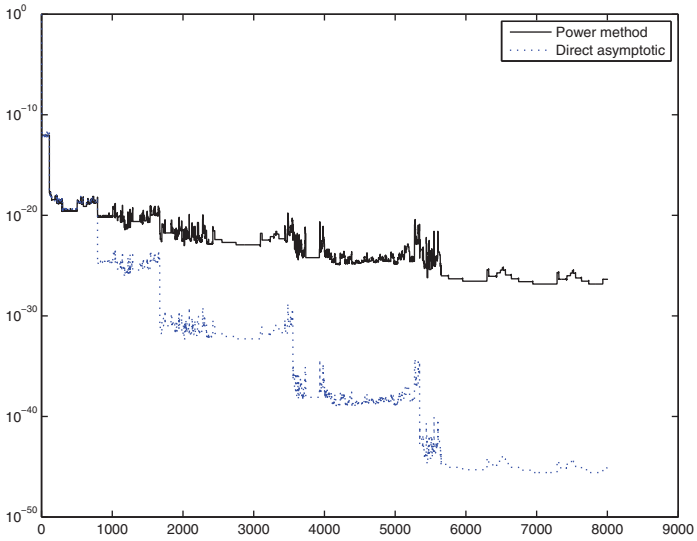

FIG. 4.7. *A comparison of the rank vectors* π *for* $\varepsilon = 10^{-5}$ *computed either by the power method or directly from the asymptotic formula.*

Let us finally mention here that when applied to Google-like matrices, our combinatorial aggregation method has some common ingredients with the block acceleration method developed, e.g., in [21] where, similarly to the present approach, local PageRanks are computed by the algorithm; see also [26]. However, our method not only makes use of the divide-and-conquer technique, but also gives some asymptotic answer regarding the solutions.

5. Final remarks. The rapid development of modern communication networks and the need of an algorithm solving hard computational problems resulted in the wide family of MC models. The most important common feature of these chains is the big

size of their state space. This often eliminates the possibility of finding the exact solution, and the approximation algorithms remain as the only way to solve the problem.

In this paper we studied a new class of approximation algorithms based on the combinatorial approach proposed in [33]. We analyzed the complexity of algorithms and studied their applicability on several Markov models of communication systems. Although the method has some limitations and there are cases where it is unsuccessful, several analytic and experimental results obtained by us classify this new method as a potentially useful tool in applications. The combinatorial aggregation method not only returns an approximate solution (which, if unsatisfactory, can be used as a reasonable starting point for a regular iterative solver), but it also provides an asymptotic formula which may be used to infer some properties of a perturbed MC.

At the end we discuss some possible extensions of the results presented here.

1. An important unsolved problem is to develop a method of choosing an appropriate value of the decomposability parameter ε. In our approach one has to assume some preprocessing phase that performs this task.
2. It would be interesting to check whether the algebraic decomposition approach proposed by us can be useful in proving some relevant properties of described MCs, e.g., rapid mixing.
3. Another challenge is an error analysis of our algorithm. This would involve a rather subtle analysis of how the approximation ratio depends on the aggregation structure and seems to be a very difficult task in general.

REFERENCES

[1] D.A. Bini, G. Latouche, B. Meini, *Numerical Methods for Structured Markov Chains*, Oxford University Press, London, 2005.

[2] A.Z. Broder, *Generating random spanning trees*, in Proceedings of the 30th IEEE Symposium on Foundations of Computer Science, 1989, pp. 442–447.

[3] R. Bru, F. Pedroche, and D. Szyld, *Additive Schwarz iterations for Markov chains*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 445–458.

[4] S. Chaiken, *A combinatorial proof of all minors matrix tree theorem*, SIAM J. Alg. Discrete Math., 3 (1982), pp. 319–329.

[5] C.A. O'Cinneide, *Relative-error bounds for the LU decomposition via the GTH algorithm*, Numer. Math., 73 (1996), pp. 507–519.

[6] P.J. Courtois, *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, 1977.

[7] T. Dayar, *Permuting Markov chains to nearly completely decomposable form*, Technical report BU-CEIS-9808, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1998.

[8] T. Dayar and W.J. Stewart, *On the effects of using the Grassman–Taksar–Heyman method in iterative aggregation-disaggregation*, SIAM J. Sci. Comput., 17 (1996), pp. 287–303.

[9] T. Dayar and W.J. Stewart, *Quasi lumpability, lower-bounding coupling matrices and nearly completely decomposable Markov chains*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 482–498.

[10] T. Dayar and W.J. Stewart, *Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains*, SIAM J. Sci. Comput., 21 (2000), pp. 1691–1705.

[11] S. Fiedler and J. Sedlácek, *O w-basich orientovaných grafu*, Cas. Pest. Mat., 83 (1958), pp. 214–225.

[12] W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, New York, 1971.

[13] M.I. Freidlin and A.D. Wentzell, *On small random perturbations of dynamical systems*, Russian Math. Surveys, 25 (1970), pp. 1–55.

[14] W.K. Grassmann, M.I. Taksar, and D.P. Heyman, *Regenerative analysis and steady–state distributions for Markov chains*, Oper. Res., 33 (1985), pp. 1107–1116.

[15] R. Hassin and M. Haviv, *Mean passage times and nearly uncoupled Markov chains*, SIAM J. Discrete Math., 5 (1992), pp. 386–397.

[16] D.P. Heyman and A. Reeves, *Numerical solution of linear equations arising in Markov chain model*, ORSA J. Comput., 1 (1989), pp. 52–60.
[17] I. Ipsen and S. Kirkland, *Convergence analysis of a PageRank updating algorithm by Langville and Meyer*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 952–967.
[18] M. Iosifescu, *Finite Markov Processes and Their Applications*, John Wiley & Sons, New York, 1980.
[19] D.D. Kafeety, C.D. Meyer, and W.J. Stewart, *A general framework for iterative aggregation/disaggregation methods*, in Proceedings of the Fourth Copper Mountain Conference on Iterative Methods, U.S. Department of Energy, 1992.
[20] S. Kamvar, T. Haveliwala, and G. Golub, *Adaptive methods for the computation of PageRank*, Linear Algebra Appl., 386 (2004), pp. 51–65.
[21] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, and G.H. Golub, *Exploiting the Block Structure of the Web for Computing PageRank*, Stanford University Technical Report, Stanford University, Palo Alto, CA, 2003.
[22] S.D. Kamvar, T.H. Haveliwala, C.D. Manning, and G.H. Golub, *Extrapolation methods for accelerating the computation of PageRank*, in Proceedings of the Twelfth International World Wide Web Conference, Budapest, Hungary, 2003.
[23] J.G. Kemeny and J.L. Snell, *Finite Markov Chains*, Van Nostrand, Princeton, 1960.
[24] J. Kohlas, *Numerical Computation of Mean Passage Times and Absorption Probabilities in Markov and Semi-Markov Models*, Zeitschrift für Oper. Res., 30 (1983), pp. A197–A207.
[25] A.N. Langville and C.D. Meyer, *A survey of eigenvector methods for Web information retrieval*, SIAM Rev., 47 (2005), pp. 135–161.
[26] C. Pan-Chi Lee, G.H. Golub, and S.A. Zenios, *A Fast Two-stage Algorithm for Computing PageRank and Its Extensions*, Stanford University SCCM Technical Report, Palo Alto, CA, 2003.
[27] I. Marek and D. Szyld, *Algebraic Schwarz methods for the numerical solution of Markov chains*, Linear Algebra Appl., 386 (2004), pp. 67–81.
[28] C.B. Moler, *Numerical computing with MATLAB*, SIAM, Philadelphia, 2004; software available at http://www.mathworks.com/moler/.
[29] M.K. Ng, *Iterative methods for Toeplitz systems*, Numer. Math. Sci. Comput., Oxford University Press, New York, 2004.
[30] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Technical report, Stanford Digital Libraries Working Paper, Palo Alto, CA, 1998.
[31] B. Philippe, Y. Saad, and W.J. Stewart, *Numerical methods in Markov chain modelling*, Oper. Res., 40 (1992), pp. 1156–1179.
[32] P. Pokarowski, *Directed forests and algorithms related to Markov chains*, Ph.D. thesis, Institute of Mathematics, Polish Academy of Sciences, 1998 (in Polish); also available online from http://www.mimuw.edu.pl/∼pokar/Works/PokarowskiPhD.pdf.
[33] P. Pokarowski, *Directed forests with applications to algorithms related to Markov chains*, Appl. Math. (Warsaw), 26 (1999), pp. 395–414.
[34] B.O. Shubert, *A flow-graph formula for the stationary distribution of a Markov chain*, IEEE Trans. Systems Man. Cybernet., 5 (1975), pp. 565–566.
[35] W.M. Spears, *Evolutionary Algorithms*, Natural Computing Series, Springer, New York, 2000.
[36] S. Serra-Capizzano, *Jordan canonical form of the Google matrix: A potential contribution to the PageRank computation*, SIAM J. Matrix Anal. Appl., 27 (2005) pp. 305–312.
[37] W.J. Stewart, *A Comparison of Numerical Techniques on Markov Modeling*, Comm. ACM, 21 (1978), pp. 144–152.
[38] W.J. Stewart, *Introduction to the numerical solution of Markov chains*, Princeton University Press, Princeton, NJ, 1994.
[39] W.J. Stewart, *Numerical methods for computing stationary distribution of finite irreducible Markov chains*, in Advances in Computational Probability, W. Grassmann, ed., Kluwer Academic Publishers, Norwell, MA, 1997.
[40] W.J. Stewart, *MARCA: Markov Chain Analyzer. A software package for Markov modelling*, in Numerical Solution of Markov Chains, W.J. Stewart, ed., M. Dekker, Inc., New York, 1991, pp. 37–62.
[41] W.J. Stewart and W. Wu, *Numerical Experiments with Iteration and Aggregation for Markov chains*, ORSA J. Comput., 4 (1992), pp. 336–350.