Aggregation Algorithms for Markov Chains with Large State Space*

Anna Gambin and Piotr Pokarowski
Institute of Informatics, Institute of Applied Mathematics
Warsaw University, ul Banacha 2 02-097, Warsaw Poland
{aniag,pokar}@mimuw.edu.pl

Abstract

We consider large Markov chains which posses specific decomposable structure, the so called *Nearly Completely Decomposable Chains* (NCD chains). A new theoretical approach for approximate computations of NCD chains has been recently introduced in [25]. The method of *forest expansions* gives raise to aggregation algorithms, which approximate effectively the characteristics of Markov chain. (i.e. the solutions of the system $\mathbf{Lx} = \mathbf{b}$ or $\mathbf{L}^T\mathbf{x} = \mathbf{b}$, where $\mathbf{L}$ is derived from transition matrix). This novel approach allows us to treat both types of problems in the unified manner. To our knowledge for the first time an aggregation scheme was used to calculate Markov chain characteristics other than stationary distribution.

In this paper we present the efficient algorithms for these problems. The algorithms are based on grouping the states of a Markov chain in such a way that the probability of changing the state inside the group is of greater order of magnitude than interactions between groups.

In contrast to existing methods our approach is based on combinatorial and graph-theoretic framework and can be seen as an algorithmization of famous Markov Chain Tree Theorem.

We establish some preliminary results on the complexity of our algorithms. Numerical experiments on several benchmark examples show the potential applicability of the method in real life problems.

*Preliminary version of some of the results described here has been presented at the workshop *Computer Algebra in Scientific Computing* CASC 2001 [14]

1 Introduction

It is often possible to represent the behavior of a physical system by describing all the different states which it can occupy and by indicating how it moves from one state to another in time. If the future evolution of the system depends only on its current state, the system may be represented by a Markov process. When the state space is discrete, the term "Markov Chain" is employed.

Markov chains are used so frequently in various areas of computer science and in other disciplines that it is not difficult to justify their importance. Making use of a suitable defined Markov chain which model the system of interest one is able to locate bottlenecks in communication network; to assess the benefit of increasing the number of CPUs in multiprocessor systems and to quantify the effect of scheduling algorithms on throughput [30].

Markov modeling, in particular reliability modeling, is used to estimate the mean time to failure of components in systems as diverse as software system and aerospace systems. The applicability of Markov chain approach is not restricted to computer science and engineering; other disciplines which benefit from it are biology, genetics, agriculture, economics, demographic and education [9].

In this paper our attention is restricted to finite discrete-time Markov chains (although methods from Section 3 are valid also for continuous time Markov chains). A finite discrete-time Markov chain over a state space $\mathcal{S}$ is usually represented by a **one-step transition probability matrix** $\mathbf{P}$ of order n, where n is the number of states in $\mathcal{S}$. The (i,j)-th element of $\mathbf{P}$, denoted $p_{i,j}$, is the one-step transition probability of going from state i to state j.

In what follows, boldface capital letters (e.g. $\mathbf{P}$) denote matrices, boldface lowercase letters (e.g. $\boldsymbol{\pi}$) denote column vectors, italic lowercase and uppercase letters (e.g. a) denote scalars and italic letters (e.g. $\mathcal{S}$) denote sets.

For a transition probability matrix $\mathbf{P}$, any vector $\boldsymbol{\pi}$ satisfying

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T \mathbf{P}, \quad \sum_{i \in \mathcal{S}} \pi_i = ||\boldsymbol{\pi}||_1 = 1, \tag{1}$$

is called a **stationary probability distribution** cf. [18, 20]. Intuitively, a Markov chain initialized with a stationary distribution follows this distribution at all points in time. On the other hand, the long-run behavior of a

Markovian system starting from any state is revealed through the solution of (1). It is crucial to compute a stationary distribution 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. In any case, the measures may be computed exactly if $\boldsymbol{\pi}$ is available.

Besides stationary distribution, some other characteristics of a Markov chain 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 similar to (1) (cf. Section 2).

The most elegant way to deal with (1) is to find the analytical formulas for the solution of the system. Unfortunately, it is usually impossible to obtain the analytic solutions for models that incorporate the characteristics sought by modelers; hence the only way is to solve the problem numerically [30]. Problems arise from the computational point of view because of the large number of states which systems may occupy. It is not uncommon for thousands of states to be generated even for simple applications. On the other hand these Markov chains are often sparse and possess specific structure (cf. Section 1.1).

To illustrate the applications of Markov chains let us consider a simple model of an interactive computer system. Figure 1 represents the architecture of a time-shared, paged, virtual memory computer. This model was widely studied in the literature [8, 30]; it is considered again in more detail in Section 4. The system consists of:

- 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 FCFS (First Come First Served). When the command is generated, the user at the terminal remains inactive until the system responds.

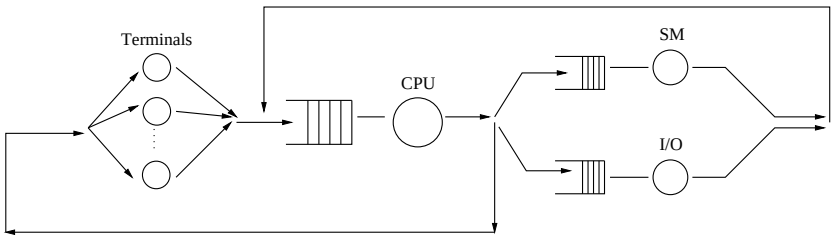

Figure 1: Illustration for Interactive Computer System

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 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 SM or I/O queue return to the CPU queue. Completion of a command is represented by a departure of the process from the CPU to the terminals.

States of the Markov chain corresponding to our model are determined by numbers of processes in all queues. The state space is large but sparse in sense of connections; there are maximum 6 transitions going out from any state. Figure 2 illustrates the structure of a Markov chain (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. At the moment we mention only how large the state space of the chain is — a more detailed analysis is deferred to Section 4. It turns out that the model for only 20 users yields the transition probability matrix of order $1,771$ with $11,011$ non-zero elements and for 50 users the matrix defining Markov chain is of order $23,426$ with $156,026$ non-zero elements.

1.1 Algorithms based on decompositional methods

A lot of research has been done concerning the numerical solutions of some linear equations that occur when one studies Markov chains (see for example [8, 17, 30]). Almost all 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 [31]). The applicability of a method depends strongly on the structure of a Markov chain considered.

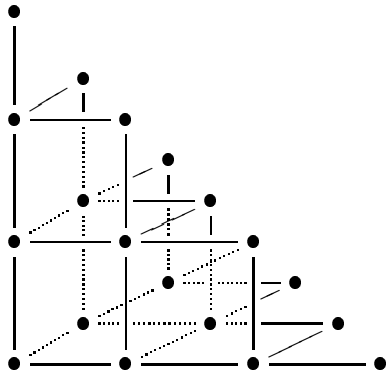

Figure 2: The 3D structure of Markov chain model of Interactive Computer System

For solving the chain of medium size, direct methods could be applied. Among others, algorithm, called GTH (Grassmann-Taksar-Heyman [15]) in the sequel, is adopted in Section 3 for dealing with small subsystems arising from decomposition of large Markov chain.

When the state space of the chain is large, even if it has sparse structure, for most of direct solving methods, elimination of one nonzero element of the matrix, produce several nonzero elements in positions which previously contained zero. This negative phenomenon is called fill-in and the amount of it can be so extensive that available memory is quickly exhausted.

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 do not alter the form of the matrix. For these reason, iterative methods (such as Gauss-Seidel iteration or Successive Overrelaxation) 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. In the case of direct methods the upper bound on the time required to obtain the solution may be determined a priori.

In this paper we consider *nearly uncoupled* or *nearly completely decomposable* Markov chains. 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 Markov chains, a family of new methods has been proposed recently. They are jointly classified as *iterative aggregation/-disaggregation* [19, 28] methods, and based on a very attractive decompositional approach. The idea follows well known **divide and conquer** principle — if the model is too large or complex to analyze, it is divided into separate subproblems. Ideally, subproblems can be solved independently and the global solution is obtained by "merging" the subproblem solutions together. Such an ideal situation occurs when the Markov chain is completely decomposable i.e. its matrix has a block structure in which all off-diagonal blocks has only zero elements. If the nonzero elements in off-diagonal blocks are small compared with those of the diagonal blocks we say about nearly completely decomposable (NCD) Markov chain (see [6, 30, 9]).

As a example consider a NCD Markov chain defined by the following matrix:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{12} & \dots & \mathbf{L}_{1N} \\ \mathbf{L}_{21} & \mathbf{L}_{22} & \dots & \mathbf{L}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{N1} & \mathbf{L}_{N2} & \dots & \mathbf{L}_{NN} \end{pmatrix}$$

where $\mathbf{L}_{11}, \mathbf{L}_{22}, \dots, \mathbf{L}_{NN}$ are square subblocks. The stationary distribution of $\boldsymbol{\pi}$ can be partitioned such that $\boldsymbol{\pi} = (\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \dots \boldsymbol{\pi}_N)$. Assume that $\mathbf{L}$ is of the form:

$$\mathbf{L} = \mathrm{diag}(\mathbf{L}_{11}, \mathbf{L}_{22}, \dots, \mathbf{L}_{NN}) + \mathbf{E}$$

where component $\mathbf{E}$ incorporates all off-diagonal blocks. The quantity

$$||\mathbf{E}||_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |e_{ij}|$$

is often referred to as **degree of coupling** and is taken to be a measure of the decomposability of the matrix (see [24]). If it is zero then the Markov chain is reducible (i.e. completely decomposable).

Another concept closely related to decomposability is that of *lumpability*. In [9] it was proved that the concept of NCD Markov chains coincides with

quasi lumpability property. We define a Markov chain to have the lumpability property if it can be divided into independent subchains and consequently its state space can be significantly reduced. In many cases, performance measures we are interested in depend on the probability of being in certain groups of states, i.e., the probability needs to be computed at a coarser level. In such a case, solving much smaller lumped chain will give us all needed information. Instead of formal definition, consider a simple example of a lumpable Markov chain defined by the probability transition matrix $\mathbf{P}$:

$$\mathbf{P} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} 0.2 & 0.3 & 0.4 & 0.1 \\ 0.3 & 0.1 & 0.4 & 0.2 \\ 0.5 & 0.1 & 0.1 & 0.3 \\ 0.5 & 0.3 & 0.2 & 0 \end{pmatrix} \end{array}$$

It has four states which can be partitioned into two groups $\mathcal{S}_1 = \{1, 3\}$ and $\mathcal{S}_2 = \{2, 4\}$ in such a way that the transition probability from each state in a given group to another group is the same. (For a fixed state, the probability of making the transition to a whole group is obtained by summing up all transitions probabilities from this state to each state inside the group.) Treating the partitions as new aggregated states we can form a smaller, lumped chain, given by the following matrix:

$$\begin{array}{c} \\ \mathcal{S}_1 \\ \mathcal{S}_2 \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{S}_1 & \mathcal{S}_2 \end{array} \\ \begin{pmatrix} 0.2 + 0.4 = 0.5 + 0.1 = 0.6 & 0.3 + 0.1 = 0.1 + 0.3 = 0.4 \\ 0.3 + 0.4 = 0.5 + 0.2 = 0.7 & 0.1 + 0.2 = 0.3 + 0.0 = 0.3 \end{pmatrix} \end{array}$$

If we are looking for all stationary probabilities then aggregation algorithms should be used. They 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 groups of closely related states together and solving the lumped Markov chain, treating each lump as a single state. This latter approach forms also a background of a new class of aggregation algorithms proposed in [25].

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 find in Section 4. We report results of experiments we have performed — they are very promising and clearly justify the

applicability of the algorithms in practical problems. This novel approach, joining successfully research from different areas such as combinatorics, linear algebra and numerical analysis, leads to algorithms which have several advantages over previous aggregation methods:

- The applicability of traditional aggregation methods is restricted to Markov chains 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 Markov chain resulting from practical examples). Algorithms derived from the method presented here work correctly when such states are present.

- Most of decomposition methods considered in the literature are designed only to solve the problem of stationary distribution, and other characteristics of Markov chain are neglected. In contrast, one of the algorithms presented in Section 3 allows to compute *mean hitting time*; designing the procedures for other characteristics is discussed latter.

2 Directed forests method

This Section is devoted to mathematical preliminaries crucial for aggregation algorithms considered in Section 3. The structure of different subjects discussed in this Section may be explained by the diagram presented in Figure 3. We discuss several known results enabling to express the solution of systems of linear equations by means of weighted directed forests (in the underlying graph). Having such *forests expansion* of a characteristics under consideration (such as stationary distribution), we are looking for an effective procedure to compute it. To this end the concept of *powerly perturbed Markov chain* is used (see [25]), as in this case there exist recursive formulas enabling to evaluate effectively the forest expansions. This procedure yields an approximation of characteristics considered.

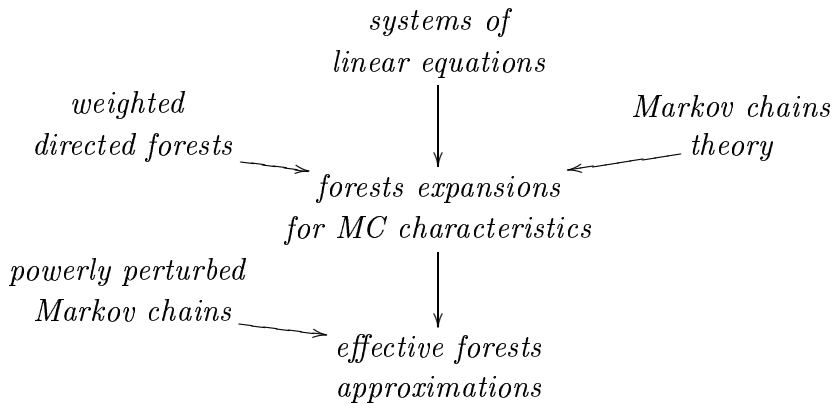

Figure 3: The structure of this Section.

2.1 Preliminaries

Consider a directed graph $G = (S, E)$, where S is nonempty finite set of vertices (called also states) and $E \subseteq S \times S$ is a set of its edges. In this section let $S = \{1, 2, \ldots s\}$, for some $s \geq 1$.

The classification of states in a graph follows the Markov chain terminology (see [18, 20] for a detailed treatment of Markov chain theory). State j is **reachable** from state i if there exists a path (nonempty sequence of edges) leading from i to j (we use the short notation $i \to j$). A state i is called **recurrent** if, for any state j, $i \to j$ implies $j \to i$; otherwise, i is called **transient**. We say that states i and j **communicate** ($i \leftrightarrow j$) when $i \to j$ and $j \to i$; state i is **absorbing** if there is no $j \neq i$ reachable from i. When all states communicate, the Markov chain and its graph are called **irreducible**.

A **strong component** of G is any maximal subgraph C of G, with the property that $i \to j$ for any two states i, j of C. A strong component is **absorbing** if it has no outgoing edges. Strong absorbing components are also called **closed classes** in the sequel. An underlying graph of each Markov chain has at least one strong absorbing component.

An acyclic subgraph $f = (S, E_f)$ of G containing all 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 **directed spanning tree**. In this section,

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 is identified with the set of its edges). We will omit the subscript G when it is obvious from the context. For readability, if $R = \{i_1, i_2, \dots i_m\}$ we use the notation $\mathcal{F}(i_1, \dots i_m)$ instead of $\mathcal{F}(R)$. 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.

Now we enrich directed graphs with weights, corresponding to the probability of changing the state in a Markov chain. A square matrix A of size s with elements from $\mathbb{C}$ (field of complex numbers) induces a graph $G(A)$ with states $\{1, 2, \dots, s\}$ and edges between all pairs (i, j) with $a_{ij} \neq 0$. 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

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

For completeness, we put $w((S, \emptyset)) = 1$ (empty forest, $R = S$) and $w(\emptyset) = 0$ (empty set of forests).

Formally, a Markov chain with the state space $S = \{1, \dots, s\}$ is defined as a sequence $X = (X_t)_{t \in \mathbb{N}}$ of random variables such that the probability of $X_i = j$ depends only on X_{i-1} (for details, we refer to [12, 18, 20]). Thorough this paper, we assume that Markov chains are given by a probability transition matrices — the only difference between these two ways of introducing a Markov chain is that given X, we have a distinguished initial distribution X_0, which in not given in a probability transition matrix.

It was observed, that many facts are valid simultaneously for both discrete and continuous time Markov chains. To deal with them at the same time we use a **laplacian matrix**, i.e. 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 $i = 1, \dots, s$.

Denote by $\mathbf{I}$ the identity matrix of size s. Let $\mathbf{P}$ be the transition probability matrix of a Markov chain X. It is easy to verify that matrix $\mathbf{L} = \mathbf{I} - \mathbf{P}$ is a laplacian matrix induced by $\mathbf{P}$. From now on, we assume that the Markov

chain is introduced by the **Markov chain laplacian matrix** i.e. laplacian matrix with nonpositive real off-diagonal entries. Such matrices are known in graph theory and combinatorics.

For $U, W \subseteq S$ and a square matrix $\mathbf{A}$ of size s, let us denote by $\mathbf{A}(U|W)$ the submatrix of $\mathbf{A}$ resulting from deletion of rows and columns indexed by U and W respectively. For the simplicity of notation we write $\mathbf{A}_{ij}$ instead of $\mathbf{A}(\{i\}|\{j\})$. Let $\mathbf{e}_s$ and $\mathbf{0}_s$ denote column vectors with each component equal to 1 and 0 respectively.

Many characteristics of Markov chains are solutions of systems of linear equations in one of the following form:

$$\mathbf{L}(R|R)\mathbf{x} = \mathbf{b} \tag{2}$$

$$\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}, \tag{3}$$

where R is a subset of states and $\mathbf{b}$ is a nonnegative vector of size $s - |R|$. As an example of (3), consider computing the stationary distribution. By (1), the **stationary distribution** of a Markov chain defined by a laplacian matrix $\mathbf{L} = (l_{ij})_{i,j=1}^{s}$ is a nonnegative, normalized vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_s)^T$, being the solution of following system:

$$\boldsymbol{\pi}^T\mathbf{L} = \mathbf{0}_s^T. \tag{4}$$

Assuming for simplicity that the states numbering implies $\pi_s > 0$, one of possible ways of solving (4) is to compute the solution $\mathbf{a}_s$ of a system of the form (3):

$$\mathbf{L}_{ss}{}^T\mathbf{a}_s = -(l_{11}, \ldots, l_{1s-1})^T$$

and then to normalize the vector $\mathbf{a}^T = (\mathbf{a}_s^T, 1)$. On the other hand solving system (3) can be reduced to computing the solution of (4) for appropriately defined laplacian $\mathbf{L}$.

Another interesting characteristics of the Markov chain, is parametrized by a subset of states $R \subseteq S$. The **mean hitting time** is an expected number of steps before reaching the set R. More formally, for $R \subseteq S$, $i, j \notin R$, and $k \in R$ we have the following definitions:

$\tau_R = \min\{t \geq 0 : X_t \in R\}$ — the **hitting time** of the set R,

$\mathbf{Pr}_i(A) = \mathbf{Pr}(A|X_0 = i)$,

$\mathbf{E}_i A = \mathbf{E}(A|X_0 = i)$ — expectation conditioned by starting from state i,

$m_i(R) = \mathbf{E}_i \tau_R$ — the **mean hitting time** of R.

The following equality is easy to verify (cf. [18]):

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

it can be reformulated in the matrix form (2) as follows ($\mathbf{m} = (m_i(R))_{i \in S \setminus R}$):

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

2.2 Markov chain tree theorem

We express the solution of a system of linear equations as a rational function of directed forest weights (called the **forest expansion**) [4]. For stationary distribution this is formulated in Markov Chain Tree Theorem (Theorem 2.1). We also give analogous expansions for other characteristics. Unfortunately, expressions obtained could contain an exponential number of summands. In Section 2.5 we present recursive formulas yielding the effective procedures for evaluating forest expansions in the special case of powerly perturbed Markov chains.

Without loss of generality assume that states are numbered in such a way that $R = \{s - |R| + 1, \ldots, s\}$; 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 have been proved in [11] and [25], respectively:

$$\det \mathbf{L}(R|R) = w(\mathcal{F}(R));$$
$$\text{if } w(\mathcal{F}(R)) \neq 0 \text{ then} \quad \mathbf{L}(R|R)^{-1} = \left[\frac{w(\mathcal{F}_{ij}(R \cup \{j\}))}{w(\mathcal{F}(R))}\right]_{i,j \in S \setminus R} \tag{5}$$

(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 (3):

$$\mathbf{A}^T \mathbf{x} = \mathbf{b} \tag{6}$$

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

$$\mathbf{L} = comp(\mathbf{A}, \mathbf{b}) = \begin{pmatrix} \mathbf{A} & \mathbf{l} \\ -\mathbf{b}^T & b \end{pmatrix}, \quad where \tag{7}$$

$$\mathbf{l} = \left(-\sum_{j=1}^{s-1} a_{1j}, \ldots, -\sum_{j=1}^{s-1} a_{s-1,j} \right)^T \quad and \quad b = \sum_{j=1}^{s-1} b_j. \tag{8}$$

Recall that $\mathcal{F}(i)$, $i = 1, \ldots, s$ denotes the set of spanning trees rooted in the state i in the graph $G(\mathbf{L})$. The following facts can be concluded from (5) 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 (6) is given by $x_i = w(\mathcal{F}(i))/w(\mathcal{F}(s))$.

Consider again the problem of computing the stationary distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_s)^T$ of a Markov chain defined by the laplacian matrix $\mathbf{L} = (l_{ij})_{i,j=1}^{s}$. Assume that an underlying graph of the Markov chain has exactly one absorbing strong component and the states are numbered in such a way that $\pi_s > 0$. Now, putting $\mathbf{A} = \mathbf{L}_{ss}$, $\mathbf{x} = (\frac{\pi_1}{\pi_s}, \ldots, \frac{\pi_{s-1}}{\pi_s})^T$ and $\mathbf{b} = (-l_{s,1}, \ldots, -l_{s,s-1})^T$ in (6) leads to the following theorem (cf. equation (2.1)).

Theorem 2.1 (Markov chain tree theorem) *If the underlying graph of a Markov chain has exactly one absorbing strong component, then the stationary distribution is given by:*

$$\pi_i = \frac{w(\mathcal{F}(i))}{\sum_{j \in S} w(\mathcal{F}(j))}, \quad for\ i = 1, \ldots s;$$

This was proved independently by many authors, among them [13, 27]; Aldous [1] calls it "the most often rediscovered result in probability theory". A proof by Matrix Tree Theorem was proposed in [25]. Analogously, 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 are:

$\mu_{ij}(R) = \mathbf{E}_i \left[\sum_{0 \le t < \tau_R} \mathbf{1}(X_t = j)\right]$ — the **mean number of visits before absorption**,

$p_{ik}(R) = \mathbf{Pr}_i\{X_{\tau_R} = k\}$ — the **probability distribution in the hitting time** of R.

$$\text{where} \qquad \mathbf{1}(\phi) = \begin{cases} 1 & \text{if } \phi \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

They can be computed by solving systems of the form (2). For a Markov chain 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 $i, j \in S \setminus R$, $k \in R$:

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

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 Markov chains — the wide class of Markov chains, for which there exists an effective way to approximate the solution of the system $\mathbf{Lx} = \mathbf{b}$ or $\mathbf{L}^T\mathbf{x} = \mathbf{b}$.

2.3 Powerly perturbed Markov chains

Nearly completely decomposable (NCD) Markov chains (see [6, 9, 30]) are defined by laplacian matrices that can be ordered so that the matrix has a block structure in which the nonzero elements of the off-diagonal blocks are small compared with those of the diagonal blocks. Such matrices often arise in queueing network analysis, large scale economic models and computer systems performance evaluation.

Formally, NCD Markov chain is defined as a family of Markov chains $\{\mathbf{L}(\varepsilon),\ \varepsilon \in$

$(0, \varepsilon_1)\}$, 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 Markov chains, aiming in expressing several different orders of magnitude of interaction strength (see for example [16]):

1. The **linearly perturbed** Markov chains

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

   where $\mathbf{L}_0$ and $\mathbf{L}_1$ are laplacian matrices;

2. The **polynomially or analytically perturbed** Markov chains

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

   where every $\mathbf{L}_n$ is an laplacian matrix, $N \leq \infty$;

In [25] a wider class of perturbed Markov chains 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 $\varepsilon \in (-\varepsilon_1, \varepsilon_1)$, $A(\varepsilon) = 0$.

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** Markov chain, if there exist matrices $\mathbf{\Delta} = (\delta_{ij})_{i,j \in S}$, 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:

$$-l_{ij}(\varepsilon) \sim \delta_{ij} \varepsilon^{d_{ij}}. \tag{9}$$

We assume that $d_{ij} = \infty$ if and only if $\delta_{ij} = 0$. In the following, we use also the concept of **powerly perturbed** nonnegative vector which is defined as the 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:

$$b_i(\varepsilon) \sim \zeta_i \varepsilon^{z_i}. \tag{10}$$

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 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 facts are easy to prove:

Fact 2.2 *Consider a powerly perturbed Markov chain defined by* $\mathbf{L}$*, with matrices* $\mathbf{\Delta}$ *and* $\mathbf{D}$ *such that (9) 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 solutions of systems $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{A}^T\mathbf{x} = \mathbf{b}$, related to powerly perturbed Markov chains, 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 2.3 ([25]) *Let matrices* $\mathbf{\Delta}$ *and* $\mathbf{D}$ *be such that (9) above holds, for a powerly perturbed Markov chain* $\{\mathbf{L}(\varepsilon), \varepsilon < \varepsilon_1\}$*; 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 (10) 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*

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

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*

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

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 Markov Chain Tree Theorem in three respects. First, powerly perturbed Markov chains are considered. Second, a general class of problems is taken into account. And third, part *(2)* deals with non-transposed matrices.

From the proof of this theorem we can deduce the asymptotic forest expansions for Markov chain 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

$$\begin{aligned} h_i &= d\left(\mathcal{F}(\{i\})\right) - \min_{j \in S} d\left(\mathcal{F}(\{j\})\right), \\ \eta_i &= \delta\left(\mathcal{F}(\{i\})\right) \Big/ \sum_{j:\, h_j = 0} \delta\left(\mathcal{F}(\{j\})\right). \end{aligned} \tag{11}$$

For the other case *(2)*, more complicated formulas express the asymptotic coefficients η_i' and h_i':

$$\begin{aligned} a_i &= \min_{j \in S \setminus R} [d(\mathcal{F}_{ij}(R \cup \{j\})) + z_j] \\ h_i' &= a_i - d(\mathcal{F}(R)), \\ \eta_i' &= \left(\sum_{j \in S \setminus R : d(\mathcal{F}_{ij}(R \cup \{j\})) + z_j = a_i} \delta(\mathcal{F}_{ij}(R \cup \{j\})) \cdot \zeta_j \right) / \delta(\mathcal{F}(R)). \end{aligned} \tag{12}$$

While in (11) it is sufficient to consider spanning trees rooted in some state i, in (12) it is necessary to take into account spanning forests rooted in $R \cup \{j\}$.

Unfortunately, all obtained expressions for asymptotic coefficients are computationally non-tractable, 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 NCD Markov chain.

2.4 Asymptotic coefficients and exact solutions

In Section 3 we describe algorithms which compute asymptotic coefficients $\mathbf{h}$ and η from Theorem 2.3. Now, we explain how they can be used to obtain the approximation of e. g. stationary distribution vector (case *(1)* from Theorem 2.3).

The algorithm takes as an input laplacian $\mathbf{L} = (l_{ij})$ defining Markov chain and parameter ε and consists of three steps:

1. construct matrices $\mathbf{\Delta}$ and $\mathbf{D}$ such that:

$$-l_{ij} = \delta_{ij} \varepsilon^{d_{ij}};$$

2. run Algorithm 2 to compute vectors η and $\mathbf{h}$;

3. set $\pi_i(\varepsilon) = \eta_i \varepsilon^{h_i}$.

This algorithm is correct in a sense that for sufficiently small ε, the computed value $\boldsymbol{\pi}(\varepsilon)$ can be arbitrarily close to an exact value. Even more, when $\varepsilon < \min_{ij} -l_{ij}$, we have $d_{ij} = 0$ (for all i, j), hence $\mathbf{L} = \mathbf{\Delta}$ and Algorithm 2

gives the exact solution (in particular, $h_i = 0$, for all i). In that case, no aggregation can be done and algorithm runs a direct (GTH) method. On the other hand, larger ε's allow to profit from a specific block structure of a laplacian matrix, which improves efficiency. Hence, there exists a tradeoff between the time and space efficiency of the algorithm and a precision of the approximation.

2.5 Fast computation of forest expansions

The algorithms discussed in Section 3 use the combinatorial approach to compute the asymptotic coefficients (vectors $\mathbf{h} = (h_i)_{i=1}^s$ and $\boldsymbol{\eta} = (\eta_i)_{i=1}^s$ from Theorem 2.3), hence they approximate 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 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 Markov chain 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 interested characteristic computed for each state of the original Markov chain. Such an iterative upgrade scheme, enabling to calculate effectively the asymptotic coefficients in the case of stationary distribution and mean hitting time, is discussed in detail in Section 3.

The task of computing exponents h_i is of quite different nature than the task of computing the coefficients η_i. While the former can be performed using purely combinatorial methods (hence precisely), the latter uses a procedure of solving a system of linear equations, exposed to numerical errors. Although calculating coefficients η_i is of crucial importance, in the sequel we concentrate on h_i only (we explain how to calculate η_i in Section 3.2). We overview the process of aggregation (it is treated again in more detail in Section 3) and state some facts on **shortest forests**, useful in computing exponents h_i.

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

$$m(i) = \min_j d_{ij}. \tag{13}$$

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:

$$\begin{aligned} d_{IJ} &= \min_{i \in I, j \in J} (d_{ij} + h(i|I)) \quad \text{where} \\ h(i|I) &= \max_{k \in I} m(k) - m(i). \end{aligned} \tag{14}$$

Values $h(i|I)$ are computed in Algorithm 2 — they correspond to coefficients h_i in a graph induced by a closed class I. Fact 2.4 below justifies such an aggregation scheme in order to calculate h_i — recall from (11) that to this aim we need $d(\mathcal{F}(i))$. From this fact (and from an accompanying fact for η_i) one derives correctness of (19) 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.,

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

Fact 2.4 *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 non-transposed case, we need a similar fact for forests, related to (12).

Fact 2.5 *Let R be a subset of states of G 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'):*

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

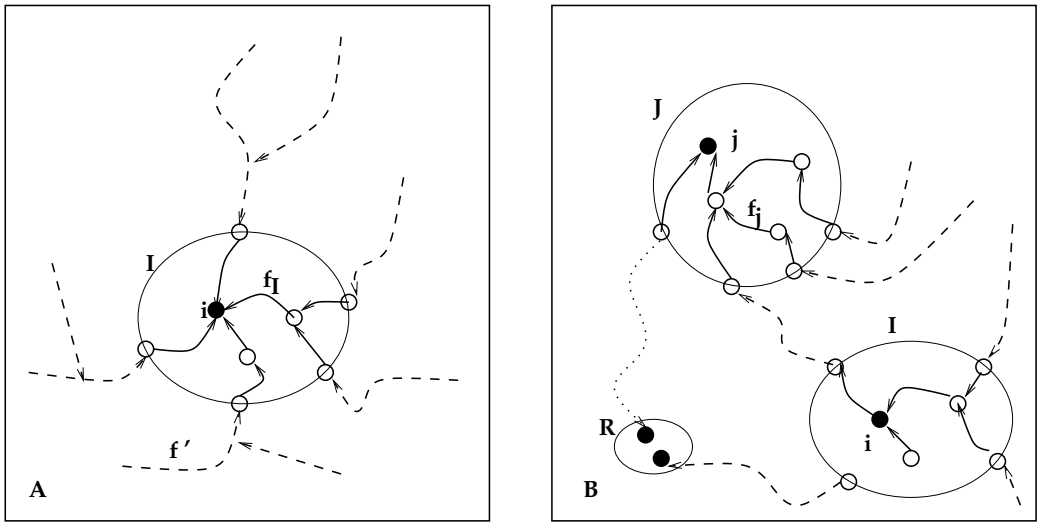

Figure 4: A: shortest tree rooted in i from Fact 2.4, B: shortest forest rooted in $R \cup j$ from Fact 2.5

We finish here the discussion about the mathematical background of algorithms — these are presented themselves in the next Section 3. Due to 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 the bibliography pointed in the text.

Figure 2.5 illustrates concept of shortest forests factorization from Facts 2.4 and 2.5.

3 Combinatorial aggregation algorithms

We present two algorithms — one approximates the stationary distribution and the other gives the approximation of the mean hitting time. They represent two classes of methods, designed to calculate approximation of solutions of linear systems build up of transpose or non-transpose matrices (cf. Section 2.1). The main idea is to reduce the size of a Markov chain state space by performing the aggregation of closely related states. During the computation

process one needs to solve small Markov chains, corresponding to one aggregate (i.e. to solve the system of linear equations of the form $\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}$ or $\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}$). To this aim the GTH method is adopted below; it can solve both kinds of systems, while keeping the numerical errors on a relatively low level (cf. Section 3.1).

3.1 Direct methods

If one has to solve the system with specific structure, the standard LU-decomposition is often not the best choice. Similar situation appears in the case of stochastic matrices, arising when one studies linear systems for some Markov chain problems. Then the **state-reduction method** seems to be the most suitable.

The state-reduction method (called here also GTH algorithm) was introduced in [15] by Grassmann, Taksar and Heyman, where it was applied to the equations yielding the stationary vector; applications to other characteristics of a Markov chain are proposed among others in [21, 25]. This approach has a probabilistic interpretation: each iteration can be viewed as a construction of a different Markov chain with one state less. Moreover, from its solution the solution of the original problem can be obtained. Numerically, this algorithm behaves surprisingly well: its numerical features are discussed in many papers (see [5, 17, 8]). The rest of this section is devoted to explain the idea of the GTH approach; we use here the notation from Section 2 and the presentation is based on [8, 25]. Consider the system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{15}$$

where the matrix $\mathbf{A}$ can be for example a transposed laplacian matrix:

$$\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}, \tag{16}$$

$\mathbf{b}$ is nonnegative vector of size $u := s - |R|$, s is the number of states of a Markov chain and R denotes subset of states forming root of the directed forest in the graph induced by the laplacian $\mathbf{L}$. From Section 2.2 (c.f. equation(5)) we know that the matrix $\mathbf{L}(R|R)$ is invertible. Assume that we are interested in computing the stationary probability vector i.e:

$$\mathbf{L}^T\boldsymbol{\pi} = \mathbf{0}, \quad \text{where} \quad \sum_{i=1}^{s} \pi_i = 1.$$

Then we can put $R = \{s\}$, as the induced graph is necessarily irreducible. Hence, equivalently, we can solve the system:

$$\mathbf{L}_{ss}^T \mathbf{a}_s = -(l_{11}, l_{13} \dots l_{1s-1})^T,$$

being an instance of (16) and then normalize the vector $\mathbf{a}^T := (\mathbf{a}_s^T, 1)$.

GTH algorithm has two variants: for a transposed and nontransposed laplacian $\mathbf{L}$, respectively. Below we describe the GTH variant dealing with transposed system (such as for the calculation of the stationary distribution); analogical approach, proposed for nontransposed variant in [21, 25], is mentioned later.

In the algorithm, we act similarly as in Gaussian elimination: the system of equations is transformed into equivalent one having upper diagonal matrix and then the solution is computed by backward substitution. The only (but crucial) difference lies in the manner we calculate pivots. It turns out that the specific structure of a matrix $\mathbf{A}$ having zero column sums allow us to calculate the solution using only nonnegative numbers and to avoid subtraction operations. Let

$$\mathbf{A} := \begin{bmatrix} a_{11} & \mathbf{y}^T \\ \mathbf{w} & \mathbf{A}_{22} \end{bmatrix}.$$

Then all columns of $\mathbf{A}$ sums to zero ($\mathbf{e}^T\mathbf{A} = 0$) and also $-a_{11} = \mathbf{e}^T\mathbf{w}$.

In the first elimination step we aim to clear the first column of $\mathbf{A}$. Assume

$$\mathbf{B}_1\mathbf{A} = \begin{bmatrix} a_{11} & \mathbf{y}^T \\ 0 & \mathbf{A}_{22}^* \end{bmatrix},$$

where

$$\mathbf{B}_1 = \begin{bmatrix} 1 & \mathbf{0} \\ -\frac{\mathbf{w}}{a_{11}} & \mathbf{I} \end{bmatrix} \quad \mathbf{B}_1^{-1} = \begin{bmatrix} 1 & \mathbf{0} \\ \frac{\mathbf{w}}{a_{11}} & \mathbf{I} \end{bmatrix}$$

then $\mathbf{A}_{22}^*$ has the same features as $\mathbf{A}$: it has zero column sums, moreover it corresponds to Markov chain with one state less. To prove this notice that we have:

$$\mathbf{e}^T\mathbf{A} = \mathbf{e}^T(\mathbf{B}_1^{-1}\mathbf{B}_1)\mathbf{A} = 0,$$

$$\mathbf{e}^T\mathbf{B}_1^{-1} = (1, 1, \dots, 1) \begin{bmatrix} 1 & \mathbf{0} \\ \frac{\mathbf{w}}{a_{11}} & \mathbf{I} \end{bmatrix} = (0, 1, \dots, 1),$$

$$(0, 1, \dots, 1)(\mathbf{B}_1\mathbf{A}) = 0 \quad \Longrightarrow \quad \mathbf{e}^T\mathbf{A}_{22}^* = 0$$

By easy induction it follows that for $k = 1, 2, \ldots$, matrix $\mathbf{A}^*_{kk}$ is a transposed laplacian matrix of smaller size. All diagonal elements of this matrix (except the last one) are positive, hence the following equality is satisfied by pivots:

$$u_k = \sum_{j=k+1}^{u+1} w_j, \tag{17}$$

where w_j are elements of vector $\mathbf{w}$ (cf. step 6 in Algorithm 1).

Hence, we modify the elimination step as follows: rather then calculating pivot in the usual way, we replace the pivot with the negated sum of the off-diagonal elements in the unreduced part of the same column.

As a conclusion from the above discussion we formulate Algorithm 1. It is presented in a more general setting i.e. for solving (16); recall that $l_{iR} := \sum_{j \in R} l_{ij}$ for $i = 1, \ldots, u$.

Algorithm 1 GTH method for solving the system $\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}$

1: **for** $k = 1$ to u **do**
2: $\quad l_{k,u+1} := l_{kR}$
3: $\quad l_{u+1,k} := b_k$
4: **end for**
5: **for** $k = 1$ to u **do**
6: $\quad u_k := -\sum_{j=k+1}^{u+1} l_{kj}$
7: $\quad$ **for** $i, j = k+1,\ i \neq j$ to $u+1$ **do**
8: $\quad\quad l_{ij} := l_{ij} - l_{ik} l_{kj} / u_k$
9: $\quad$ **end for**
10: **end for**
11: **for** $k = u$ downto 1 **do**
12: $\quad x_k := \left(l_{u+1,k} - \sum_{j=k+1}^{u} l_{jk} x_j \right) / u_k$
13: **end for**

The GTH variant of Gaussian elimination can be also used to solving the system of the form:

$$\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}, \tag{18}$$

where $\mathbf{b}$ is nonnegative vector of size u. Systems like (18) are used to determine for instance the vector of mean hitting time (cf. Section 3.3) — from the equations $\mathbf{L}(R|R)\mathbf{m} = \mathbf{e}$, as well as the distribution matrix at the moment reaching the set R — from the equations $\mathbf{L}(R|R)\mathbf{A} = -\mathbf{L}(R|S \setminus R)$. The only difference here is dealing with non-transposed matrix; now the pivots are calculated as a upper-diagonal row sums.

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 Markov chain characteristics defined by the system with non-transposed $\mathbf{L}$ are discussed in Section 3.3.

Recall that for the powerly perturbed Markov chain $\mathbf{L}(\varepsilon)$ 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 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 (14). 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.

Algorithm 2 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 $m(I^k)$ (cf. (13) in Section 2.5)
    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 (20)
      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

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):

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

(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 n-th 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 k-th phase for the subgraph of G^k restricted to a closed class I^k. The following recursive relation can be shown:

$$\pi_i(\varepsilon) \sim \eta^1(i|I^1)\varepsilon^{h^1(i|I^1)}\eta^2(I^1|I^2)\varepsilon^{h^2(I^1|I^2)} \cdot \ldots \cdot \eta^n(I^{n-1}|I^n)\varepsilon^{h^n(I^{n-1}|I^n)}\pi(I^n) \tag{19}$$

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 Fact 2.4:

$$\begin{aligned} h(i|I^k) &= h(i|I^{k-1}) + h^k(I^{k-1}|I^k) \\ \eta(i|I^k) &= \eta(i|I^{k-1})\eta^k(I^{k-1}|I^k) \end{aligned} \tag{20}$$

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

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

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 Markov chain 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 independently. 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

Procedure 3 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 $m(I^k)$
    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}

We describe here an approximation algorithm for another important characteristics of the Markov chain — the mean hitting time. According to Section 2.1, this is an interesting example of the linear equations with a non-transposed 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. Formal definition can be found in Section 2; recall that

$\mathbf{m}_i(R)$ satisfies the following equality:

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

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

$$\mathbf{L}(R|R)\mathbf{m} = \mathbf{e}, \tag{21}$$

where $\mathbf{e} = (11 \ldots 1)^T$. Hence, computing mean hitting time corresponds to approximating the solution of the system $\mathbf{L}(R|R)\mathbf{x} = \mathbf{e}$. 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 Markov chain it was proved that:

$$m_i(R)(\varepsilon) \sim \beta_{iR} \varepsilon^{b_{iR}}, \tag{22}$$

where asymptotic coefficient β and b are defined as follows:

$$a_i := \min_{j \notin R} [d(\mathcal{F}_{ij}(R \cup \{j\}))], \tag{23}$$

$$b_{iR} := a_i - d(\mathcal{F}(R)), \tag{24}$$

$$\beta_{iR} := \frac{\sum_{j \notin R : d(\mathcal{F}_{ij}(R \cup \{j\})) = a_i} \delta(\mathcal{F}_{ij}(R \cup \{j\}))}{\delta(\mathcal{F}(R))}. \tag{25}$$

The algorithm for the mean hitting time performs the aggregation process starting from a graph $G^0 = (S^0, E^0)$, which differs slightly from that in Algorithm 2. 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 states of R). Moreover, we add one more state u^*, and 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 equation (21), forming a fragment of the matrix $(comp(\mathbf{L}(R|R)^T, e))^T$. 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 (14) in Section 2.5). Finally, if the graph constructed by now we denote by G, as G_0 we take G_{min}, similarly as previously.

Algorithm 4 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 4 Calculate asymptotic coefficients b_{iR} and β_{iR} i.e. approximate the mean hitting time $m_i = \beta_{iR}\varepsilon^{b_{iR}}$

1: **Aggregation**
2: **for** each closed class in S^k, say I^k, s.t. $I^k \neq u^R$ **do**
3: compute $b_{I^k R} := d_{I^k u^*} - m(I^k)$
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 $n(I^k) := \min_{J^k \in M}(d_{I^k J^k} - m(I^k) + b_{J^k R})$
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 4 of Algorithm 4 requires taking into account all edges between aggregated states, not only the shortest ones. Hence, coefficients β_{iR} computed by this algorithms do not coincide precisely with those needed in (22); fortunately, the computed values, say β'_{iR}, are asymptotically equivalent:

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

Originally, in step 4 of Algorithm 4 only shortest edges were considered — the necessity of the modification was observed during experiments. A similar modification of Algorithm 2 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 then 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 4 can be easily applied to calculation of other characteristics of a Markov chains, 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.

4 Analysis of algorithms and case studies

We analyze combinatorial aggregation algorithms, in particular we estimate time and space complexity of Algorithms 2 and 4. Then we point out that the efficiency of algorithms depends strongly on the value of parameter ε which determines the number of phases of algorithm and the structure of aggregation process in each phase.

Complexity analysis. The upper bound on time complexity of Algorithms 2 and 4 presented in previous chapter is $\mathcal{O}(n^3)$, where n is the number of states. The upper bound on memory needed is $\mathcal{O}(n^2)$. However, the complexity of algorithms depends strongly on the structure of a Markov chain 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 $m = \Theta(n)$. These estimates strongly motivate further studies of methods to establish an appropriate value of parameter ε.

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. $n_1 + n_2 + \ldots + n_k = |\mathcal{S}|$, transient state are singleton closed classes).

Theorem 4.1 *The time and space costs 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 time cost of $n + m$ is due to Tarjan's algorithm for finding strongly connected components, used to determine closed classes. 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 cost of running GTH procedure k times, for each closed class separately.

For the space cost, it is determined by the size of representation of a matrix together with a quadratic space needed in GTH algorithm. □

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 $n^{(i)}$, $k^{(i)}$, $m^{(i)}$, ${n_j}^{(i)}$, 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 time and space cost of all aggregation phases, respectively.

1. $p = 1, k = 1$; a pessimistic case — all states are aggregated during the first step; GTH procedure is performed 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 time cost 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. to 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 i-th 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 time cost 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 cost is increased by $\mathcal{O}(k^2)$.

Degree of coupling There are several approaches aiming at defining a measure of decomposability of a transition matrix (see for example [24]). Such a measure is usually closely related to the parameter ε and could be probably used to find an optimal value for ε. Moreover, a right value of ε can significantly improve precision of approximation as well as time and space needed for the algorithm.

In our approach it is assumed that some preprocessing phase is needed to find an appropriate value of ε. It seems to be a challenging task to design an automatic procedure which would propose a value for ε, that would guarantee a desired accuracy and time/space effectiveness of aggregation.

4.1 Numerical experiments

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 block successive over relaxation (BSOR) in the case of large models. The latter one was chosen because it is widely accepted as the best known method calculating the approximate solution of problems related to large and sparse Markov chains (see for example [10, 23]). First we concentrate on small examples which appear frequently in the literature (cf. [6, 30]). Although the sizes of the matrices considered are quite small, it is still instructive to examine the effect of using our algorithms in such cases. Two problems investigated in Sections 4.4 and 4.5 are real life examples and had been extensively studied by many authors (see e. g. [10, 8, 33]).

For each small example considered in the sequel we compute the relative error of the solution (denoted by $\boldsymbol{\pi}^*$ and $\mathbf{m}^*$) compared with outcome of the GTH procedure (vectors: $\boldsymbol{\pi}$ and $\mathbf{m}$, respectively). Errors are computed w.r.t. GTH algorithm, despite that it is also exposed itself to numerical errors. GTH is suitable for this purpose as it has an *a priori* error estimation [5, 25].

Another two 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.

For readability, instead of laplacians we show probability transition matrices (zero entries are marked by dots). The experiments show several advantages of our algorithms over existing methods:

- for the first time the aggregation approach is successful in computing the Markov chain characteristics which are solutions of non-transposed 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*;
- comparison with GTH procedure shows that 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.

4.2 Small examples

Courtois matrix

The first problem considered in this section is the 8×8 Courtois matrix studied already in [6] as an example of NCD Markov chain. The parameter ε is chosen to be 0.001. During computing stationary distribution first phase of aggregation results in three aggregated states, $I = \{1, 2, 3\}$, $J = \{4, 5\}$

[1] for a formal definition see conditions $6.1 - 6.4$ in [30] pp. 335.

and $K = \{6, 7, 8\}$, which are aggregated in the following step into one closed class.

$$\begin{bmatrix}
.85 & . & .149 & .0009 & . & .00005 & . & .00005 \\
.1 & .65 & .249 & . & .0009 & .00005 & . & .00005 \\
.1 & .8 & .0996 & .0003 & . & . & .0001 & . \\
. & .0004 & . & .7 & .2995 & . & .0001 & . \\
.0005 & . & .0004 & .399 & .6 & .0001 & . & . \\
. & .00005 & . & . & .00005 & .6 & .2499 & .15 \\
.00003 & . & .00003 & .00004 & . & .1 & .8 & .0999 \\
. & .00005 & . & . & .00005 & .1999 & .25 & .55
\end{bmatrix}$$

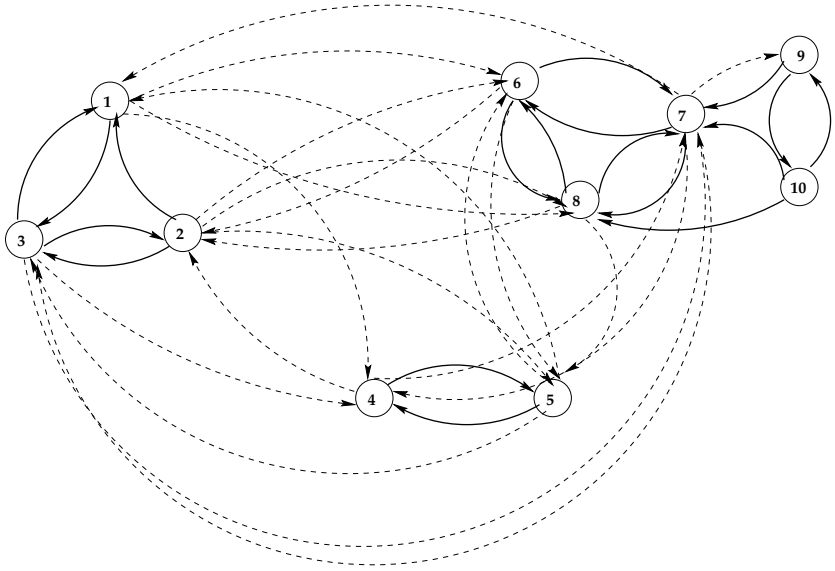

Figure 5: Courtois matrix with two asymptotically transient states.

Courtois matrix with two additional states

We have modified the Courtois matrix by adding two **asymptotically transient** states. It is worth noting that this matrix does not satisfy NCD conditions, as not all off-diagonal elements have small values, but it is still feasible for our aggregation algorithms. The parameter ε is chosen to be 0.001. In the first step of aggregation we obtain three closed classes (similarly as before) and two transient states 9 and 10. Second step of aggregation results in one

closed class lumping together all states (cf. Fig.5).

$$\begin{bmatrix}
.85 & . & .149 & .0009 & . & .00005 & . & .00005 & . & . \\
.1 & .65 & .249 & . & .0009 & .00005 & . & .00005 & . & . \\
.1 & .8 & .0996 & .0003 & . & . & .0001 & . & . & . \\
. & .0004 & . & .7 & .2995 & . & .0001 & . & . & . \\
.0005 & . & .0004 & .399 & .6 & .0001 & . & . & . & . \\
. & .00005 & . & . & .00005 & .6 & .2499 & .15 & . & . \\
.00003 & . & .00003 & .00002 & . & .1 & .8 & .0999 & .00002 & . \\
. & .00005 & . & . & .00005 & .1999 & .25 & .55 & . & . \\
. & . & . & . & . & . & .33 & . & . & .67 \\
. & . & . & . & . & . & .05 & .625 & .325 & .
\end{bmatrix}$$

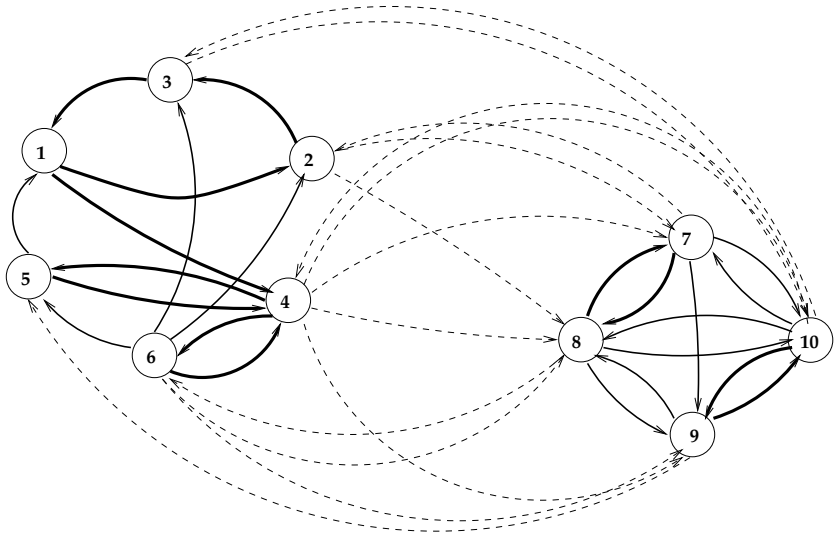

Figure 6: Aggregation in Stewart matrix.

Stewart matrix

The last small example is that proposed by Stewart [30]. Note the impact of

different ε values on accuracy of the result.

$$\begin{bmatrix}
. & .5 & . & .5 & . & . & . & . & . & . \\
. & . & .999994 & . & . & . & .000006 & . & . & . \\
.999995 & . & . & . & . & . & . & . & . & .000005 \\
. & . & . & .019999 & .44 & .44 & .000003 & .000002 & .000003 & .000002 \\
.0001 & . & . & .99 & .0099 & . & . & . & . & . \\
. & .0002 & .0002 & .99 & .00959 & . & . & .000005 & .000005 & . \\
. & .000001 & . & . & . & . & .49 & .49 & .001 & .00099 \\
. & .000003 & . & . & . & .000007 & .49999 & .49 & .006 & .004 \\
. & . & . & . & .000001 & . & . & .009999 & . & .99 \\
. & . & .000005 & .000005 & . & . & .00999 & .01 & .98 & .
\end{bmatrix}$$

The following array summarizes results of experiments. Recall that precision measures are accurate indicators of the number of correct digits in the approximate solution. Observe the effect of choosing smaller ε in the case of Stewart example. In general, calculated precision measures are on the level $\log \frac{1}{\varepsilon}$ — the same situation occurs in larger examples.

problem type	$\mathbf{L}(R\|R)\mathbf{x} = \mathbf{b}$		$\mathbf{L}^T(R\|R)\mathbf{x} = \mathbf{b}$	
	$\mathrm{Prec}_1(\mathbf{m}^*, \mathbf{m})$	$\mathrm{Prec}_2(\mathbf{m}^*, \mathbf{m})$	$\mathrm{Prec}_1(\boldsymbol{\pi}^*, \boldsymbol{\pi})$	$\mathrm{Prec}_2(\boldsymbol{\pi}^*, \boldsymbol{\pi})$
Courtois	4.98	4.81	4.45	4.23
modified Courtois	4.88	4.86	4.48	4.26
Stewart				
$\varepsilon = 0.01$	4.05	4.85	1.97	2.03
$\varepsilon = 0.00001$	5.95	5.49	4.20	4.79

4.3 Block Successive Over Relaxation (BSOR)

Block successive over relaxation 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 A nor c depend on the iteration step k. In particular, BSOR procedure (see Algorithm 5) is parametrized by:

- starting solution vector;

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

Algorithm 5 assumes the partitioning of laplacian matrix into N blocks; i-th diagonal block denoted by $\mathbf{L}_{ii}$ is of size n_i. In the sequel we consider five

---

Algorithm 5 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

---

block partitioning strategies:

SCC We are looking for strongly connected components in the underlying graph of the Markov chain. 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) [32].

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.

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]

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.

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 8 for two-dimensional Markov chain model.

4.4 Interactive Computer System

In this section we come back to the model described in Figure 1 in Section 1 (for a detailed treatment see [29]). We recall that the model represents a time-shared multiprogrammed, paged, virtual memory computer system, modeled as a closed queueing network. In order to perform numerical experiments we assign specific values for the parameters of the model such as:

- the number of user 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 [29],
- the mean think 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 a 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 [29]. Recall that the state of the system is coded by a triple

$\mathbf{z} = (z_1, z_2, z_3)$ of non-negative numbers, where z_1 denotes the number of users thinking or busy at theirs 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 transition which can be made from any state, i.e. from (z_1, z_2, z_3) to states:

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

We present outcome of the algorithm approximating mean hitting time and stationary distribution for 3, 10 and 20 users. In each case we analyze number of phases in 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.

N	n	m	$\text{Prec}_1(\mathbf{m})$	$\text{Prec}_2(\mathbf{m})$	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	$\text{Prec}_1(\boldsymbol{\pi})$	$\text{Prec}_2(\boldsymbol{\pi})$	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 non-transposed system of equations (mean hitting time) is not less effective than in the case of stationary distribution. Up to now aggregation approach was used only in solving problems like stationary distribution. Combinatorial aggregation algorithms allow us to treat both type of problems in the unified manner.

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	$0.95e-10$	$0.17e-14$

Table 1: Numerical results for ICS example

For ICS model with 50 users we perform several experiments with iterative method. The solution computed by combinatorial aggregation is used as a starting vector for BSOR algorithm. 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.

The stopping criterion we use in BSOR is $||\boldsymbol{\pi}^{(k)} - \boldsymbol{\pi}^{(k-1)}||_\infty \leq \mathit{stop\_tol}$ where stopping tolerance $\mathit{stop\_tol}$ is set to 10^{-10}. In the table $||\Delta\boldsymbol{\pi}||_\infty$ 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 1) that starting from 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 2 is comparable with the cost of the first iteration of BSOR. One can additionally accelerate combinatorial aggregation by using iterative method (e.g. SOR) instead of GTH for solving subproblems inside closed classes.

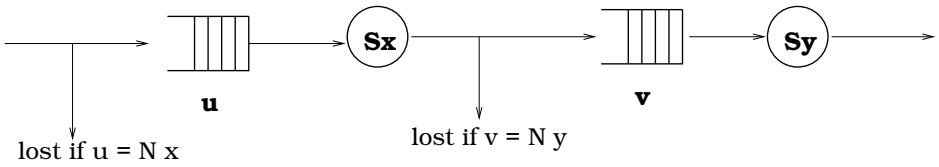

Figure 7: Telecommunication model for 2D example

4.5 A Two-Dimensional Markov Chain Model

We consider here a two dimensional Markov chain, studied e.g. in [10, 32]. It is a simple telecommunication model illustrated in Figure 7. There are two servers; each has a queue of waiting tasks of prespecified maximum size. A task arrives, wait for the first server, then waits for the second and finally leaves the network.

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	$0.99e-10$	$0.58e-11$

Table 2: Numerical results for 2D example.

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 North-West. From any non-boundary state (u, v) there are 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
$(u-1, v+1)$ (North-West)	with rate	u	task enters second queue

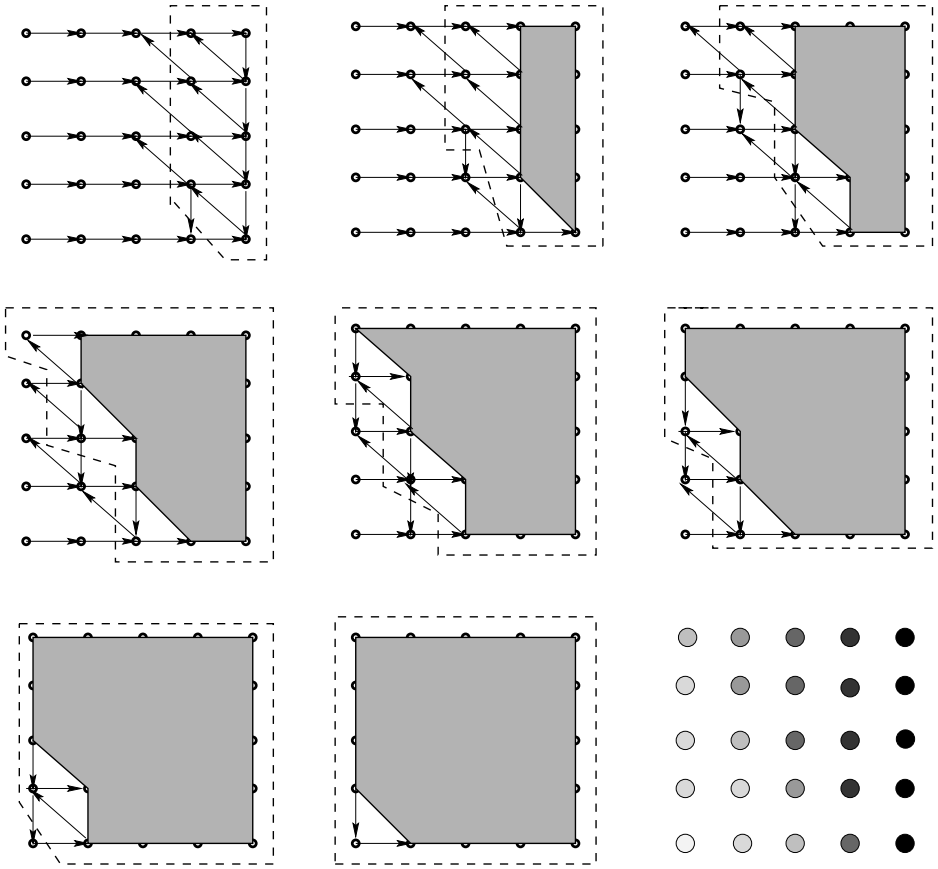

Figure 8: Aggregation in 2D example.

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:

1. the approximate solution is calculated using 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.

3. the solution is computed using BSOR starting from approximation obtained by combinatorial aggregation, the same partitionings are con-

sidered.

Figure 8 illustrates the process of aggregation in combinatorial aggregation algorithms for $N_x = N_y = 4$. There are 8 phases of aggregation each shown in a separate figure. We see a subgraph of shortest edges G_{min} in every phase; dashed line surrounds the only non-singleton 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 non-singleton 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 2. We observe about 25% speedup of BSOR method started from 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 starting vector is greater in **Asymp**.

5 Final remarks

The rapid development of modern communication networks and the need of algorithm solving hard computational problems resulted in the wide family of Markov chain 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 [25]. We analyzed the complexity of algorithms and studied their applicability on several Markov models of communication systems. Both analytic and experimental results obtained by us classify this new method as a potentially very useful tool in practice.

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

Open problems:

(1) The combinatorial aggregation approach is a new very efficient algorith-

mic method for solving problems related to Markov chains, e.g. computing the stationary distribution. It would be useful to design procedures for other Markov chain characteristics based on this theory.

(2) An important unsolved problem is to develop a method of choosing an appropriate value of decomposability parameter ε. In our approach one has to assume some preprocessing phase that performs this task.

(3) It would be interesting to check, whether the algebraic decomposition approach proposed by us can be useful in proving some relevant properties of described Markov chains, e.g. rapid mixing.

(4) 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 structur e and seems to be a very difficult task.

References

[1] Aldous, D.: *Reversible Markov chains and random walks on graphs*, 1994, Preprint.

[2] Azar, Y., Broder, A.Z., and Frieze, A.M.: On the problem of approximating the number of bases of a matroid. *Inform. Process. Lett.*, **vol. 50**, 1994 pp. 9-11.

[3] Broder, A.Z.: Generating random spanning trees. *30th IEEE Symposium on Foundations of Computer Science*, 1989, pp. 442-447.

[4] Chaiken, S.: A combinatorial proof of all minors matrix tree theorem, *SIAM J. Alg. Disc. Math.*, **vol. 3**, 1982, pp. 319-329.

[5] O'Cinneide, C. A.: Relative-error bounds for the LU decomposition via the GTH algorithm, *Numerische Mathematik*, **vol. 73**, 1996, pp. 507-519.

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

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

[8] Dayar, T. and Stewart, W.J.: On the effects of using the Grassman-Taksar-Heyman method in iterative aggregation-disaggregation, *SIAM Journal on Scientific Computing*, **vol. 17**, 1996, pp. 287-303.

[9] Dayar, T. and Stewart, W.J.: Quasi-lumpability, lower-bounding coupling matrices and nearly completely decomposable Markov chains, *SIAM Journal on Matrix Analysis and Applications*, **vol. 18**, 1997, pp. 482-498.

[10] Dayar, T. and Stewart, W.J.: Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains, to appear in *SIAM Journal on Scientific Computing.*

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

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

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

[14] Gambin, A Pokarowski,P.: A new combinatorial algorithm for large Markov chains. *Computer Algebra in Scientific Computing - CASC 2001*, V.G. Ghanza, E.W. Mayr, E.V. Vorozthsov (Eds.), Springer, Berlin 2001, pp. 195-212.

[15] W.K. Grassmann, M.I. Taksar and D.P. Heyman. Regenerative analysis and steady–state distributio ns for Markov chains, *Operations Research*, **vol.33**, pp 1107-1116, 1985.

[16] Hassin, R. and Haviv, M.: Mean passage times and nearly uncoupled Markov chains, *SIAM Journal of Disc. Math.*,1992, **vol. 5**, pp. 386-397.

[17] Heyman, D. P. and Reeves, A.: Numerical solution of linear equations arising in Markov chain model. *ORSA Journal on Computing.*, 1989, **vol. 1**, pp 52-60.

[18] Iosifescu, M.: *Finite Markov Processes and Their Applications*, Wiley and Sons, 1980.

[19] Kafeety, D.D., Meyer, C.D. and Stewart, W.J.: A General Framework for Iterative Aggregation/Disaggregation Methods, *Proceedings of the Fourth Copper Mountain Conference on Iterative Methods*, 1992.

[20] Kemeny, J.G. and Snell, J.L.: *Finite Markov Chains.* Van Nostrand, Princeton, 1960.

[21] Kohlas, J.: Numerical Computation of Mean Passage Times and Absorption Probabilities in Markov and Semi-Markov Models, *Zeitschrift für Operatiions Research*, **vol. 30**, 1983, A197-A207.

[22] Patel, J. H.: Performance of processor-memory interconnections for multiprocessors, *IEEE Trans. on Computing*, **vol. C-30**, 1981, pp. 771-780.

[23] Philippe, B., Saad Y. and Stewart, W. J.: Numerical Methods in Markov Chain Modelling. *Operations Research*, **vol. 40**, **no. 6**, 1992, pp. 1156-1179.

[24] Pokarowski, P.: Uncoupling measures and eigenvalues of stochastic matrices, *Journal of Applied Analysis*, **vol. 4, no. 2**, 1998, pp 259-267.

[25] Pokarowski,P.: Directed forests with applications to algorithms related to Markov chains. *Applicationes Mathematicae*, **vol. 26(4)**, pp 395-414, 1999.

[26] Pfister, G.F., Brantley, W.C., George, D.A., Harvey, S.L., Kleinfelder, W.J., McAuliffe, K.P., Melton, E.A., Norton, V.A., Weiss, J. Introduction to the IBM Research Parallel Processor Prototype (RP3), in J.J. Dongarra, *Experimental Parallel Computing Architectures* Elsevier Amsterdam, 1987, pp. 123-140.

[27] Shubert, B.O.: A flow-graph formula for the stationary distribution of a Markov chain, *IEEE Trans. Systems Man. Cybernet.*, 1975,**vol. 5**, pp. 565-566.

[28] Spears, W. M.: *Evolutionary Algorithms* Natural Computing Series, Springer, 2000.

Comm. ACM , **vol. 21**, 1978, pp. 144-152.

[29] Stewart, W. J.: A Comparison of Numerical Techniques on Markov Modeling, *Comm. ACM* , **vol. 21**, 1978, pp. 144-152.

[30] Stewart, W. J.: *Introduction to the numerical solution of Markov chains*, Princeton University Press, 1994.

[31] Stewart, W. J.: Numerical methods for computing stationary distribution of finite irreducible Markov chains, *Chapter 3 of Advances in Computational Probability*. Edited by Winfried Grassmann; Kluwer Academic Publishers, 1997.

[32] Stewart, W. J.: MARCA: Markov Chain Analyzer. A software package for Markov modelling. In W.J. Steward (Ed.), *Numerical solution of Markov chains*, M. Dekker, Inc., New York, 1991, pp. 37-62.

[33] Stewart, W. J. and Wu W.: Numerical Experiments with Iteration and Aggregation for Markov chains, *ORSA Journal on Computing*, **vol 4, no. 3**, pp. 336-350, 1992.