A New Combinatorial Algorithm for Large Markov Chains
(Extended Abstract)

Anna Gambin[1][*] and Piotr Pokarowski[2]

[1] Institute of Informatics, email: `aniag@mimuw.edu.pl`
[2] Institute of Applied Mathematics, email: `pokar@mimuw.edu.pl`
Warsaw University Banacha 2, 02-097 Warsaw, Poland
tel: +48 22 55 444 06   fax: +48 22 55 444 00

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 [11, 12]. The method of *forest expansions* gives raise to aggregation algorithms, which approximate effectively the characteristics of Markov chain. 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 [5] the algorithm approximating stationary distribution (described by transposed system $\mathbf{L}^T \mathbf{x} = \mathbf{b}$, where $\mathbf{L}$ is derived from transition matrix) was presented; in this paper we illustrate that combinatorial aggregation in the case of non-transposed system (defining e.g. *mean hitting time*) is not less effective. 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.

Keywords: Markov chains, decomposability, aggregation, mean hitting time.

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 [16]. 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.

We propose a new combinatorial aggregation approach which is applicable when the state space of Markov chain is large enough so that the direct methods (e.g. Gauss elimination) are inefficient. It is based on lumping together closely related states. This approach is attractive in that case, because the structure of transition matrix implies that standard iterative methods converge very slowly (see discussion in [16, 17] and the references therein).

Formally, a Markov chain with the state space $S = \{1, \ldots, s\}$ 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 [8, 10]). 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 is 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, following [11], a **laplacian matrix**, i.e. matrix $\mathbf{L} = (l_{ij})_{i,j=1}^{s}$, $l_{ij} \in \mathbb{R}$ satisfying $l_{ii} = -\sum_{j:j \neq i} l_{ij}$ for $i = 1, \ldots, s$. Denote by $\mathbf{I}$ the identity matrix of size s. Let $\mathbf{P}$ be the **transition probability matrix** of a Markov chain. 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. 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 non-positive real off-diagonal entries.

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

$$\mathbf{L}(R|R)\mathbf{x} = \mathbf{b} \tag{1}$$

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

where R is a subset of states, $\mathbf{b}$ is a nonnegative vector of size $s - |R|$ and $\mathbf{L}(R|R)$ is a submatrix of $\mathbf{L}$ resulting from deletion of rows and columns indexed by R; $\mathbf{L}^T$ denotes transposition. The most elegant way to deal with (1) and (2) is to find the analytical formulas for the solution of the system. Unfortunately, it is usually impossible and the only way is to solve the problem numerically [16]. 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.

Motivation and related research. 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 [2, 16]). 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 [17]). Some of them turn out to be quite efficient, others have serious drawbacks. The applicability of a method depends strongly on the structure of the Markov chain considered (see discussion in [3]). In this paper we focus on *nearly uncoupled* or *nearly completely decomposable* Markov chains (see [1, 16]).

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.

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

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

where $\mathbf{L}_{11}, \mathbf{L}_{22}, \ldots, \mathbf{L}_{NN}$ are square subblocks. Assume that $\mathbf{L}$ is of the form:

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

where component $\mathbf{E}$ incorporates all off-diagonal blocks. In the sequel we assume that the submatrices $\mathbf{L}_{ii}$ are of moderate size, hence we apply the direct method (a modification of the standard Gaussian elimination called GTH after Grassmann-Taksar-Heyman [6]) to solve the systems of the form $\mathbf{L}_{ii}\mathbf{x} = \mathbf{b}$, arising from decomposition of large NCD Markov chain. In Section 3 we discuss the possibility to speedup the computation by choosing some iterative method instead.

For solving NCD Markov chains the methods known as *aggregation* and *iterative aggregation/disaggregation* are the most suitable. Our combinatorial aggregation approach can be seen as a generalization of existing aggregation algorithms (c. f. [9]). In [4, 5] we summarize the most important advantages over previous methods. Here we emphasize the following aspects:

- All known aggregation 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 to this, we propose the procedure approximating the **mean hitting time**. Similar algorithms can be obtained for other characteristics being the solution of (1). For the first time the aggregation approach is successful in computing Markov chain characteristics which are solutions of **non-transposed** systems of equations.
- Comparison with GTH procedure shows the height precision level of approximation computed by our algorithms.

Structure of the paper. In Section 2 a mathematical theory behind the algorithms is sketched (see [11, 12] for more detailed treatment). The following section contains the description of the algorithm and the discussion of its complexity. In Section 4 we report results of numerical experiments we have performed — they are very promising and clearly justify the applicability of our approach for NCD Markov chains. Final remarks and some open problems can be found in the last section.

2 Mathematical Preliminaries

This section is devoted to mathematical preliminaries crucial for aggregation algorithms. We define here the Markov chain characteristics we are interested in. It is argued that to compute them, one needs to solve certain systems of linear equations. Then, we recall 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, we are looking for an effective procedure to compute it. To this end the concept of *powerly perturbed Markov chain* [11] is used, as in this case there exist recursive formulas enabling to evaluate effectively the forest expansions. This procedure yields an approximation of characteristics considered.

Let $G = (S, E)$ be a directed graph with the set of vertices (called also states) $S = \{1, 2, \ldots s\}$, for $s \geq 1$. The classification of states in a graph follows the Markov chain terminology (c.f. [8, 10] for a detailed treatment of Markov chain theory). A **strong component** of G is any maximal subgraph C of G, with the property that for any two states i, j of C there exists a path from i to j. A strong component is **absorbing** if it has no outgoing edges. Strong absorbing components are also called **closed classes** in the sequel. 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**. We write shortly forest (tree) instead of directed spanning forest (tree).

Let $\mathcal{F}(R)$ denote the set of all forests in G having the root R (a forest is identified with the set of its edges). For fixed $i \notin R$ and $j \in R$, $\mathcal{F}_{ij}(R) \subseteq \mathcal{F}(R)$ denotes the set of all forests with the root R, containing a path from i to j.

Now we assume G to be an underlying graph of Markov chain induced by a square real matrix A of size s (i. e. there are edges between all pairs (i, j) with $a_{ij} \neq 0$). We define the **(multiplicative) weight** of a forest $f = (S, E_f)$ and the weight of a set $\mathcal{F}$ of forests as follows:

$$w(f) = \prod_{(i,j) \in E_f} (-a_{ij}), \qquad w(\mathcal{F}) = \sum_{f \in \mathcal{F}} w(f).$$

Let $R \subseteq S$, $i, j \notin R$, and $k \in R$; Markov chain characterisctics we consider in the sequel are defined as follows:

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

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

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

$$p_{ik}(R) := \mathbf{Pr}_i\{X_{\tau_R} = k\} \text{ --- the \textbf{probability distribution in the hitting time}}$$
$$\text{of } R,$$
$$\mathbf{Pr}_i(A) := \mathbf{Pr}(A \,|\, X_0 = i),$$
$$\tau_R := \min\{t \geq 0 : X_t \in R\} \text{ --- the hitting time of the set } R,$$
$$\mathbf{E}_i A := \mathbf{E}(A | X_0 = i) \text{ --- expectation conditioned by starting from state } i.$$

The **mean hitting time** of R can be calculated by solving following system of linear equation of the form (1) ($\mathbf{m} = (m_i(R))_{i \in S \setminus R}$):

$$\mathbf{L}(R|R)\mathbf{m} = \mathbf{e}; \tag{3}$$

where $\mathbf{e} = (11 \ldots 1)^T$. Two other characteristics can be also presented as solutions of non-transposed systems of the form (1).

Famous Markov Chain Tree Theorem [14] characterizes stationary distribution by a rational function of directed forest weights called the **forest expansion**. A result of [11] extends MCT Theorem to the characteristics introduced above:

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

In Section 3 we present the combinatorial aggregation algorithms approximating vector $\mathbf{m} = m_i(R)$. The analogous constructions for $\mathbf{p} = p_{ik}(R)$ and $\boldsymbol{\mu} = \mu_{ij}(R)$ is discussed.

The concept of powerly perturbed Markov chains [11] is crucial for effective approximation of the solution of system $\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}$ ($\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}$). It subsumes all previously known generalizations of NCD Markov chains, aiming in expressing several different orders of magnitude of interaction strength (see for example [7]).

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

$$-l_{ij}(\varepsilon) \sim \delta_{ij}\varepsilon^{d_{ij}}. \tag{4}$$

Powerly perturbed nonnegative vector is defined as the family $\{\mathbf{b}(\varepsilon), \ \varepsilon \in (0, \varepsilon_1)\}$ of nonnegative vectors of size u, such that for some vectors $\boldsymbol{\zeta} = (\zeta_i)_{i=1}^u$ and $\mathbf{z} = (z_i)_{i=1}^u$, with $\zeta_i \geq 0$, $z_i \in \mathbb{R}$, for $i = 1, \ldots, u$, the following holds:

$$b_i(\varepsilon) \sim \zeta_i\varepsilon^{z_i}. \tag{5}$$

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

$$G^*(\mathbf{D}) = (S, \{(i, j) \in S \times S : \delta_{ij} \neq 0\}).$$

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

$$
(i) \quad \left\{ \begin{array}{l} d(f) := \displaystyle\sum_{(i,j)\in f} d_{ij} \\[2mm] \delta(f) := \displaystyle\prod_{(i,j)\in f} \delta_{ij} \end{array} \right\} \qquad \text{an asymptotic weight of the forest } f.
$$

$$
(ii) \quad \left\{ \begin{array}{l} d(\mathcal{F}) := \displaystyle\min_{f\in\mathcal{F}} d(f) \\[2mm] \delta(\mathcal{F}) := \displaystyle\sum_{f\in\mathcal{F}:\, d(f)=d(\mathcal{F})} \delta(f) \end{array} \right\} \qquad \text{an asymptotic weight of the set of forests } F.
$$

We describe the asymptotics of solutions of systems $\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}$ and $\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}$, related to powerly perturbed Markov chains, in terms of directed forests expansions (i.e. the rational function of forests' weights). The following theorem, says that a solution of a system of linear equations, for a perturbed chain, can be treated as a perturbed vector.

Theorem 1 ([11]). *Let matrices $\mathbf{\Delta}$ and $\mathbf{D}$ be such that (4) 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 (5) 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 $x_i(\varepsilon) \sim \eta_i \varepsilon^{h_i}$, for some constants η_i, 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 $x_i(\varepsilon) \sim \beta_i \varepsilon^{b_i}$, for some constants β_i, b_i.

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 $\delta(\mathcal{F})$. Both cases *(1)* and *(2)*, although described similarly, differ substantially in difficulty (cf. [4, 5]). — this is visible in different structures of algorithms. While in *(1)* it is sufficient to consider spanning trees rooted in some state i, in *(2)* it is necessary to take into account spanning forests rooted in $R \cup \{j\}$.

Unfortunately, as we will see all obtained expressions for asymptotic coefficients (i.e. constants η, h, β, b from Theorem above) are computationally non-tractable, at least directly, because of their exponential length. In the next section 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.

3 Aggregation Algorithm for Mean Hitting Time

The mean hitting time ($\mathbf{m}$) is an important characteristics of the Markov chain defined by the system of linear equations with a non-transposed matrix. It is characterized 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.

Applying Theorem 1 we derive $m_i(R)(\varepsilon) \sim \beta_{iR} \varepsilon^{b_{iR}}$, where asymptotic coefficients $\boldsymbol{\beta} = (\beta_{iR})$ and $\mathbf{b} = (b_{iR})$ are defined as follows:

$$a_i := \min_{j \notin R} [d(\mathcal{F}_{ij}(R \cup \{j\}))], \qquad\qquad b_{iR} := a_i - d(\mathcal{F}(R)),$$

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

We present the algorithm which computes asymptotic coefficients $\boldsymbol{\beta}$ and $\mathbf{b}$ — they can be used to approximate $\mathbf{m}$ (c. f. Section 5). The main idea of the algorithm is to 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 each phase graphs induced by matrix $\mathbf{D}$ is considered. The algorithm groups states in each closed class of the graph and solves the system of linear equations restricted to this class. Smaller size, hence tractable, systems of equations can be solved by a direct method. Before passing to a next phase, an aggregation procedure is performed, lumping all states in each closed class into a new, aggregated state.

The algorithm consists of two phases. The first one runs the aggregation scheme, similarly as for transposed case (c.f. [4, 5]); however, the aggregation can leave several closed classes of the original graph not lumped together. Then, the second phase calculates coefficients b_{iR} and β_{iR} for closed classes resulted from first phase (see the description of Algorithm2). Computed values are then propagated throughout each class.

Correctness of the algorithm. The interesting observation is that the task of computing exponents b_{iR} (h_i) is of quite different nature than the task of computing the coefficients β_{iR} (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 β_{iR} is of crucial importance, in the sequel we concentrate on b_{iR} and h_i. We overview here the process of aggregation and state some facts on **shortest forests**, useful in computing exponents h_i (b_{iR}). Note that, the coefficients h_i and η_i play an auxiliary role in the algorithm — they are necessary during the aggregation phase and alow to construct an aggregated underlying graph.

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

$$m(i) := \min_j d_{ij}. \tag{6}$$

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:

$$d_{IJ} := \min_{i \in I, j \in J} (d_{ij} + h(i|I)) \quad \text{where} \quad h(i|I) := \max_{k \in I} m(k) - m(i). \tag{7}$$

Values $h(i|I)$ are computed in aggregation precedure — they correspond to coefficients h_i in a graph induced by a closed class I. Lemma 1 below justifies such an aggregation scheme in order to calculate h_i. 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., $d(f) = \min_{f' \in \mathcal{F}(i)} d(f') = d(\mathcal{F}(i))$.
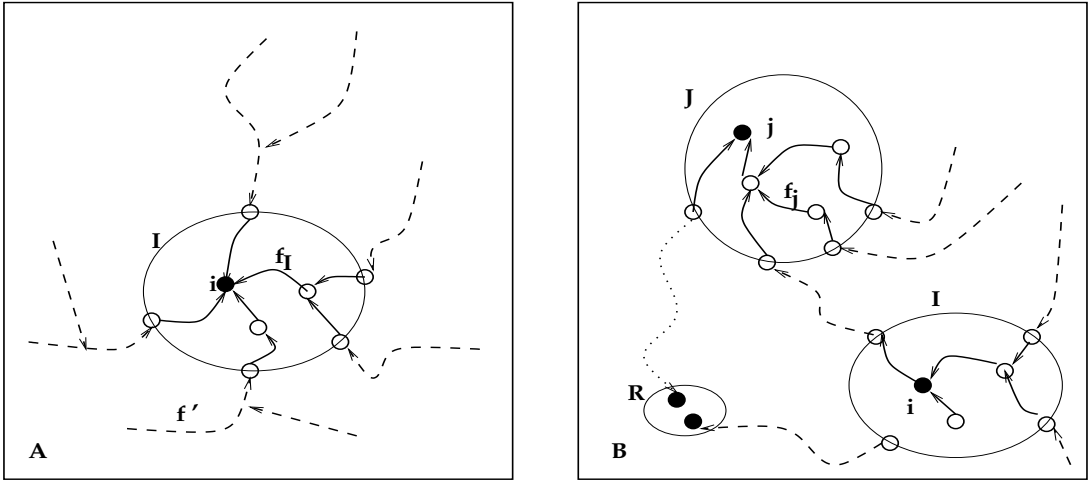

Fig. 1. A: shortest tree rooted in i from Lemma 1, B: shortest forest rooted in $R \cup j$ from Lemma 2

Lemma 1 ([11]). *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.

Lemma 2 ([11]). *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'):*

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

Procedure 1 Aggregation

1: construct $G^0 = (S^0, E^0)$
2: $k := 0$
3: **repeat**
4:　　$k := k + 1$
5:　　find partition of G^{k-1} into closed classes (ignore edges leading to u^*)
6:　　construct S^k
7:　　**for** each closed class in S^k, say I^k, s.t. $I^k \neq u^R$ **do**
8:　　　　construct laplacian $\mathbf{L}_k$
9:　　　　compute stationary distribution i. e. solve the system $\mathbf{L}_k^T \mathbf{x} = \mathbf{b}$
10:　　　　compute $m(I^k)$
11:　　　　**for** each aggregated state I^{k-1} in I^k **do**
12:　　　　　　compute $\eta^k(I^{k-1}|I^k)$ and $h^k(I^{k-1}|I^k)$
13:　　　　**end for**
14:　　　　**for** all neighbors of class I^k **do**
15:　　　　　　determine shortest edges
16:　　　　**end for**
17:　　**end for**
18:　　construct new set of aggregated edges E^k (update edges leading to u^*)
19:　　$G^k := (S^k, E^k)$
20: **until** G^k has the same number of states as G_{k-1}

We 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 Procedure 1. Initially, define the graph G^0 which differs slightly from that in transposed case. Namely, all states from R are replaced by a new state u_R; it has no outgoing edges and for any $i \notin R$, an edge from i to u_R is weighted 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 $\mathbf{e}$ in equation (3). Notice that edges leading to u^* are ignored, during the construction of a new set of aggregated states (steps 5 and 6 in Procedure 1). However they are updated in step 18 according to the same rule as other edges (cf. formula (7)). For $k = 1, 2, \ldots$, we define inductively $G'_k = aggr(G_{k-1})$. 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.

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.

From Lemmas 1 and 2 one can derive following recursive relation:

$$h(i|I^k) = h(i|I^{k-1}) + h^k(I^{k-1}|I^k), \quad \eta(i|I^k) = \eta(i|I^{k-1})\eta^k(I^{k-1}|I^k). \tag{8}$$

Coefficients $h^k(I^{k-1}|I^k)$ can be computed using the value of $m(I^k)$ (step 12): $h^k(I^{k-1}|I^k) = \max_{I \subseteq I^k} m(I) - m(I^{k-1})$. Using (8) we can justify the aggregation scheme: in a given iteration we have only to consider all vertices I aggregated during the previous step, which belong to the closed class I^k. Procedure 1 results in aggregated graph $G^k = (S^k, E^k)$ having the set of closed classes as vertices ($S^k = \{I^k, J^k \ldots\}$) and aggregated

edges ($d_{I^k J^k} \in E^k$). Aiming in computing coefficients $\mathbf{b}$ and β (i. e. approximating $\mathbf{m}$) we run the second phase of Algorithm 2. Recall that $m(I^k)$ denote the weight of the shortest edge leaving I^k.

Algorithm 2 Calculate asymptotic coefficients b_{iR} and β_{iR} i.e. approximate the mean hitting time $m_i = \beta_{iR}\varepsilon^{b_{iR}}$

1: run **Procedure 1**
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**

Finally, we need to explain how to compute effectively $\eta^k(I^{k-1}|I^k)$ in step 12 of Procedure 1 and β_{iR} in step 18 of Algorithm 2. Recall that we have assumed that the Markov chain under consideration possesses a specific block structure, namely the sizes of all 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 $\mathbf{L}_k\mathbf{x} = \mathbf{b}$ ($\mathbf{L}_k^T\mathbf{x} = \mathbf{b}$), independently inside each class I^k. For the solution the following holds: $x_k(\varepsilon) \sim \beta_{kR}\varepsilon^{b_{kR}}$ ($x_k(\varepsilon) \sim \eta^k\varepsilon^{h^k}$). Now having already computed b_{kR} (h^k) and putting some fixed ε_0, the corresponding coefficients η^k are derived as the solution of the equation: $\beta_{I^k R} = x_{I^k R}\varepsilon_0^{-b_{I^k R}}$ ($\eta^k(I^{k-1}|I^k) = x_{I^{k-1}}\varepsilon_0^{-h^k(I^{k-1}|I^k)}$).

Complexity issues. The upper bound on time cost of combinatorial aggregation 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 cost of algorithm depends strongly on the structure of a Markov chain under consideration. In [4] 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 algorithm uses only $\mathcal{O}(n+m)$ space. 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

the issue of establishing an appropriate value of parameter ε, which has a significant impact on time and space cost.

In our analysis we have assumed that all closed classes (i.e. diagonal blocks of NCD Markov chain) are treated by a precise direct method (GTH). One can achieve better bound by choosing an iterative method to solve the subsystems on the desired level of accuracy.

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 2. *The time and space costs of a single phase of aggregation are as follows:*

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

The cost of the Algorithm 2 is equal to the total cost of all aggregation phases increased by $\mathcal{O}(c^3)$, where c is the number of closed classes when the aggregation processes stops; the space cost is also increased by $\mathcal{O}(c^2)$.

4 Numerical Experiments

We describe here the outcomes obtained when combinatorial aggregation algorithms are used to compute stationary distribution ($\boldsymbol{\pi}$) and mean hitting time ($\mathbf{m}$) of several Markov models. Stationary distribution (non-transposed case) [5], is concerned in order to demonstrate that computing the mean hitting time is of the same difficulty. We concentrate mainly on small examples which appear frequently in the literature (cf. [1, 16]). 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. In contrast to this, last problem investigated by us is a real life example and had been extensively studied by many authors (see e. g. [3]).

For each 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 [11].

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):

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

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).

Courtois matrix. The first problem considered in this section is the 8×8 Courtois matrix studied already in [1] 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, $I = \{1, 2, 3\}$, $J = \{4, 5\}$ and $K = \{6, 7, 8\}$, which are aggregated in the following step into one closed class.

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

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.
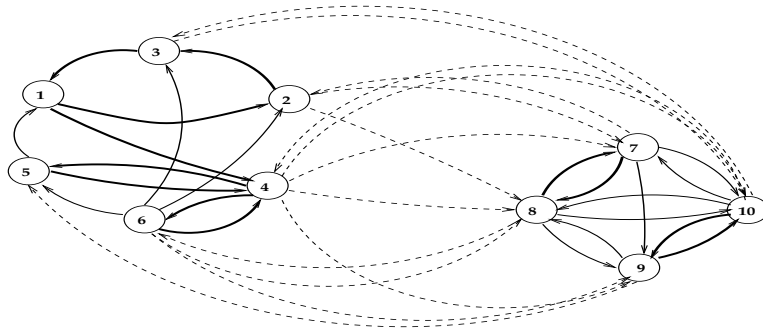

Fig. 2. Aggregation in Stewart matrix.

Stewart matrix. This example is proposed by Stewart [16]. 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 (c.f. [4]).

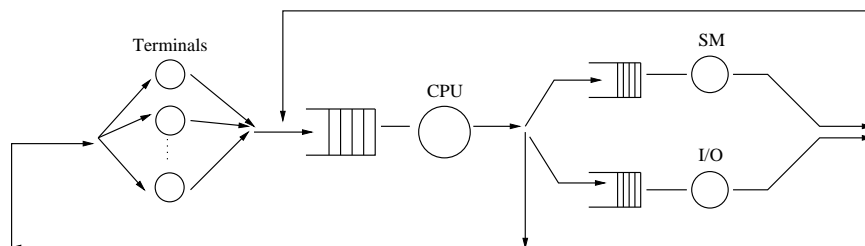| problem type	$\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}$		$\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}$	
	$\mathrm{Prec}_1(\mathbf{m}^*, \mathbf{m})$	$\mathrm{Prec}_2(\mathbf{m}^*, \mathbf{m})$	$\mathrm{Prec}_1(\boldsymbol{\pi}^*, \boldsymbol{\pi})$	$\mathrm{Prec}_2(\boldsymbol{\pi}^*, \boldsymbol{\pi})$
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

Fig. 3. Illustration for Interactive Computer System

Interactive Computer System. We consider the Markov model described in Figure 3 (for a detailed treatment see [15]). It 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 according to [15]. 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.

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.

5 Final Remarks

We proposed a new class of approximation algorithms based on combinatorial approach developed in [11]. We presented an algorithm approximating the mean hitting time. However after some minor modifications this algorithm is applicable to all non-transpose system, hence it can be used 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. We analyzed the complexity of algorithm and studied its applicability on several Markov models. Some outcomes of numerical experiments are reported. Both analytic and experimental results obtained by us classify this new method as a potentially very useful tool in practice. Our algorithm computes the asymptotic coefficients $\mathbf{b}$ and $\boldsymbol{\beta}$. It takes as an input laplacian $\mathbf{L} = (l_{ij})$ defining Markov chain; given additionally value ε it can be used to approximate mean hitting time (and other characteristics) as follows:

1. construct matrices $\boldsymbol{\Delta}$ and $\mathbf{D}$ such that: $-l_{ij} = \delta_{ij}\varepsilon^{d_{ij}}$, where $\varepsilon < \delta_{ij} \leq 1$;
2. run Algorithm 2 to compute vectors $\boldsymbol{\beta}$ and $\mathbf{b}$;
3. set $\mathbf{m}_i(R)(\varepsilon) := \beta_i\varepsilon^{b_i}$.

When $\varepsilon < \min_{ij} -(l_{ij})$, we have $d_{ij} = 0$ (for all i, j), hence $\mathbf{L} = \boldsymbol{\Delta}$ and Algorithm 2 gives the exact solution. On the other hand, larger ε's allow to profit from a specific block structure of a laplacian matrix to improve efficiency. In fact, there is a tradeoff between time/space efficiency of the algorithm and precision of approximation.

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 to calculate a value of ε, that would guarantee desired accuracy and time/space

effectiveness of aggregation. There are several approaches aiming at defining a measure of decomposability of a transition matrix (see for example [13]). Such measures are usually closely related to parameter ε and could be probably used to find an optimal value for it. Moreover, a right value of ε can significantly improve precision of approximation as well as time and space needed for the algorithm. Another challenge is an error analysis of our algorithm. We have some preliminary results for some restricted families of Markov chains. However the obtained analytical formulas are often non-optimal and yields to significant overestimation of error level.

References

1. Courtois, P. J.: *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, 1977.
2. 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.
3. 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*.
4. Gambin, A.: Combinatorial Methods in Approximation Algorithms for Markov Chains with Large State Space. *PhD thesis*, TR No. 260 Institute of Informatics, Warsaw University 1999.
5. Gambin, A. and Pokarowski, P.: A combinatorial aggregation algorithm for stationary distribution of a large Markov chain, accepted to FCT'2001.
6. W.K. Grassmann, M.I. Taksar and D.P. Heyman. Regenerative analysis and steady-state distributions for Markov chains, *Operations Research*, **vol.33**, pp 1107-1116, 1985.
7. Hassin, R. and Haviv, M.: Mean passage times and nearly uncoupled Markov chains, *SIAM Journal of Disc. Math.*,1992, **vol. 5**, pp. 386-397.
8. Iosifescu, M.: *Finite Markov Processes and Their Applications*, Wiley and Sons, 1980.
9. 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.
10. Kemeny, J.G. and Snell, J.L.: *Finite Markov Chains*. Van Nostrand, Princeton, 1960.
11. Pokarowski, P. Directed forests and algorithms related to Markov chains, *PhD thesis* Institute of Mathematics, Polish Academy of Sciences, 1998.
12. Pokarowski, P. Directed forests with applications to algorithms related to Markov chains, *Applicationes Mathematicae*, **vol. 26, no. 4**, 1999, pp. 395-414.
13. Pokarowski, P.: Uncoupling measures and eigenvalues of stochastic matrices, *Journal of Applied Analysis*, **vol. 4, no. 2**, 1998, pp 259-267.
14. 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.
15. Stewart, W. J.: A Comparison of Numerical Techniques on Markov Modeling, *Comm. ACM* , **vol. 21**, 1978, pp. 144-152.
16. Stewart, W. J.: *Introduction to the numerical solution of Markov chains*, Princeton University Press, 1994.
17. 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; To be published by Kluwer Academic Publishers, 1997.