A combinatorial aggregation algorithm
for computing the stationary distribution
of a large Markov chain
(Extended abstract)

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

[1] Institute of Informatics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
email: `aniag@mimuw.edu.pl`
[2] Institute of Applied Mathematics, Warsaw University
Banacha 2, 02-097 Warsaw, Poland
email: `pokar@mimuw.edu.pl`

Abstract. A new aggregation algorithm for computing the stationary distribution of a large Markov chain is proposed. This algorithm is attractive when the state space of Markov chain is large enough so that the direct and iterative methods are inefficient. It is 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. The correctness of the combinatorial aggregation is justified by the method of *forest expansions* developed recently in [14, 15]. In contrast to existing methods our approach is based on combinatorial and graph-theoretic framework and can be seen as an algorithmization of famous Matrix Tree Theorem. The general method is illustrated by an example of computing the stationary distribution. We establish also some preliminary results on the complexity of our algorithm. Numerical experiments on several benchmark examples show the potential applicability of the algorithm in real life problems.

1 Introduction

When considering complex systems, e.g. communication networks, the usual way to compute their important parameters (like throughput) is to study a probabilistic markovian model instead of the system itself. The usual task is to compute some of its characteristics (like stationary distribution), corresponding to the parameters of a modeled system. To this aim an appropriate system of linear equation must be solved.

Our attention is restricted to finite discrete-time Markov chains (although presented approach is valid also for continuous time). Such a Markov chain over a state space $\mathcal{S}$ is usually represented by a **transition probability matrix P** of order n, where n is the number of states in $\mathcal{S}$. The (i, j)-th element of **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. **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. [11, 13]. Besides stationary distribution, some other characteristics of a Markov chain are also of interest, 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).

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 and the only way is to solve the problem numerically [18]. 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.

Example 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 [4, 18]; it is considered again in more detail in Appendix. 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
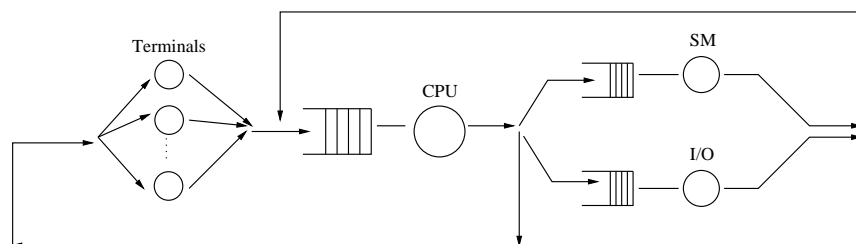

Fig. 1. Illustration for Interactive Computer System

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. 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: i. e. $S = \{(x, y, z) \in \mathbb{N}^3 : x + y + z \leq n\}$ for n users; three coordinates correspond to the number of processes in three queues. The state space is large but sparse in sense of connections; there are maximum 6 transitions going out from any state, corresponding to entering or exiting a queue by a process.

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 [4, 18]). 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 [19]). The applicability of a method depends strongly on the structure of a Markov chain considered.

For solving the chain of medium size, direct methods could be applied. In Section 3 the GTH (Grassmann-Taksar-Heyman [9]) algorithm which is a modification of the standard Gaussian elimination, is adopted 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 (see discussion in [18, 19] and the references therein). 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 focus on *nearly uncoupled* or *nearly completely decomposable* Markov chains (see [2, 5, 18]). 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 methods has been proposed. They are jointly classified as *iterative aggregation/disaggregation* [12] methods, and based on a 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.

Our combinatorial aggregation approach can be seen as a generalization of existing aggregation algorithms. The most important advantages over previous methods are:

— The presented algorithm uses combinatorial properties of *directed forests* in the underlying graph of a Markov chain. Combinatorial and graph-theoretic approach simplifies the description of algorithm and proof of correctness which relies on certain facts about forest expansions of solutions of linear equation systems. These facts are formulated in Matrix Tree Theorem and extensions of it proved in [1, 14].
— The applicability of traditional aggregation methods is restricted to Markov chains with a regular NCD structure[1]; 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) are problematic (such states are very common in large Markov chain resulting from practical examples). Algorithms derived from our method work correctly when such states are present.

---

[1] See conditions $6.1 - 6.4$ in [18] pp. 335.

– 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, our approach offers possibility of designing procedures for other characteristics too.

Structure of the paper In Section 2 a mathematical theory behind the algorithms is sketched (see [14, 15] for more detailed treatement). The following section contains algorithm themselves. The complexity analysis and several case studies can be find in Section 4. In Appendix we report results of experiments we have performed — they are very promising and clearly justify the applicability of the algorithm in practical problems.

2 Directed forests method

Consider a directed graph $G = (S, E)$; let the set of vertices (called also states) $S = \{1, 2, \ldots s\}$, for some $s \geq 1$. The classification of states in a graph follows the Markov chain terminology (see [11, 13] for a detailed treatment of Markov chain theory). State j is reachable from state i if there exists a path 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**. The Markov chain and its graph are called **irreducible**. when $i \to j$ and $j \to i$ for all states i, j.

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**. 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 (a 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 $R = \{i_1, i_2, \ldots i_m\}$ we use the notation $\mathcal{F}(i_1, \ldots i_m)$ instead of $\mathcal{F}(R)$. For fixed $i \notin R$ and $j \in R$, $\mathcal{F}_{ij}(R) \subseteq \mathcal{F}(R)$ denotes the set of all 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 $\mathbb{R}$ induces a graph $G(A)$ with states $\{1, 2, \ldots, s\}$ 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) and $w(\emptyset) := 0$ (empty set of forests).

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 [14], 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. 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 non-positive real off-diagonal entries. Such matrices are known in graph theory and combinatorics (see discussion in [14] and the references therein).

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 of the following form[2]:

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

where R is a subset of states and $\mathbf{b}$ is a nonnegative vector of size $s - |R|$; $\mathbf{L}^T$ denotes transposed matrix. As an example of (2), 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:

$$\boldsymbol{\pi}^T \mathbf{L} = \mathbf{0}_s^T. \tag{3}$$

Assuming for simplicity that the states numbering implies $\pi_s > 0$, one of possible ways of solving (3) is to compute the solution $\mathbf{x}$ of a system of the form (2):

$$\mathbf{L}_{ss}{}^T \mathbf{x} = -(l_{11}, \ldots, l_{1s-1})^T \tag{4}$$

and then to normalize the vector $(\mathbf{x}^T, 1)$. On the other hand solving system (2) can be reduced to computing the solution of (3) for appropriately defined laplacian $\mathbf{L}$.

We express the solution of a system of linear equations as a rational function of directed forest weights (called the **forest expansion**) cf. [14]. For stationary distribution this is formulated in the following theorem (proved independently by many authors, among them [7, 16]):

Theorem 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:*

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

Nearly completely decomposable (NCD) Markov chains (see [2, 5, 18]) 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.

---

[2] Our method can be also adapted for a non-transposed case $\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}$; this result is to be reported elsewhere.

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 [10]). In [14] a wide class of Markov chains has been defined, subsuming previously known classes. 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 $d_{ij} \in \mathbb{R} \cup \{\infty\}$, 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:

$$-l_{ij}(\varepsilon) \sim \delta_{ij} \varepsilon^{d_{ij}}. \tag{5}$$

We assume that $d_{ij} = \infty$ if and only if $\delta_{ij} = 0$. In the following, we also use 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 $\boldsymbol{\zeta} = (\zeta_i)_{i=1}^{u}$ and $\mathbf{z} = (z_i)_{i=1}^{u}$, with $\zeta_i \geq 0$, $z_i \in \mathbb{R} \cup \{\infty\}$, for $i = 1, \dots, u$, the following holds:

$$b_i(\varepsilon) \sim \zeta_i \varepsilon^{z_i}. \tag{6}$$

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 set $\mathcal{F}$ of forests in $G^*(\mathbf{D})$ we study parameters:

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

(ii) $\quad \left\{ \begin{array}{l} d(\mathcal{F}) := \displaystyle\min_{f \in \mathcal{F}} d(f) \\ \delta(\mathcal{F}) := \displaystyle\sum_{f \in \mathcal{F}: d(f)=d(\mathcal{F})} \delta(f) \end{array} \right\}$ $\qquad$ 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 fact is easy to prove:

Fact 2. *Consider a powerly perturbed Markov chain defined by $\mathbf{L}$, with matrices $\boldsymbol{\Delta}$ and $\mathbf{D}$ such that (5) 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 asymptotic behavior of solutions of system $\mathbf{L}^T(R|R)\mathbf{x} = \mathbf{b}$, 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. Proof omitted here can be found in [14].

Theorem 3. *Let matrices $\boldsymbol{\Delta}$ and $\mathbf{D}$ be such that (5) 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 $\boldsymbol{\zeta}$ and $\mathbf{z}$ of size $u := s - |R|$ be such that (6) holds, for a powerly perturbed vector $\mathbf{b}$. Suppose that there exist a forest with the root R in $G^*(\mathbf{D})$. Then 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

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

where the coefficients η_i, h_i are some constants, $i = 1, \dots, u$.

In the special case of stationary distribution, (i. e., equation (4)) we have:

$$
\begin{aligned}
h_i &:= d\left(\mathcal{F}(\{i\})\right) - \min_{j \in S} d\left(\mathcal{F}(\{j\})\right), \\
\eta_i &:= \delta\left(\mathcal{F}(\{i\})\right) \Big/ \sum_{j : h_j = 0} \delta\left(\mathcal{F}(\{j\})\right).
\end{aligned}
\tag{7}
$$

Unfortunately, all obtained expressions for asymptotic coefficients (vectors $\mathbf{h}$ and $\boldsymbol{\eta}$ from Theorem 3) are computationally non-tractable, at least directly, because of their exponential size. We discuss the aggregation approach, yielding effective and accurate procedures for computing the asymptotic coefficients and approximate values of the stationary probability vector of NCD Markov chain.

3 Combinatorial aggregation

Asymptotic coefficients and exact solutions. Before describing the algorithm for computing asymptotic coefficients $\mathbf{h}$ and $\boldsymbol{\eta}$, we explain how it can be used to obtain the approximation of stationary distribution vector.

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 $\boldsymbol{\Delta}$ and $\mathbf{D}$ such that:

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

2. run Algorithm 1 to compute vectors $\boldsymbol{\eta}$ and $\mathbf{h}$;
3. set $\pi_i(\varepsilon) := \eta_i \varepsilon^{h_i}$;

When $\varepsilon < \min_{ij} -(l_{ij})$, we have $d_{ij} = 0$ (for all i, j), hence $\mathbf{L} = \boldsymbol{\Delta}$ and Algorithm 1 gives the exact solution (in particular, $h_i = 0$, for all i). In that case, no aggregation can be done and 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 time/space efficiency of the algorithm and precision of the approximation.

Fast computation of forest expansions. The algorithm reduces 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 matrices $\mathbf{D}$ and $\mathbf{\Delta}$ are 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. The solutions of these systems are used to upgrade the values of asymptotic coefficients computed for each state of the original Markov chain. Before passing to a next phase, an aggregation procedure is performed, lumping all states in each closed class into a new, aggregated state.

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 later how to calculate η_i).

Algorithm 1 Calculate asymptotic coefficient η and h.

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
6:    construct S^k
7:    **for** each closed class in S^k, say I^k **do**
8:       construct laplacian L_k
9:       compute stationary distribution i. e., solve the system $\mathbf{L}_k^T \mathbf{x} = \mathbf{b}$
10:       compute $m(I^k)$ (cf. (8))
11:       **for** each aggregated state I^{k-1} in I^k **do**
12:          compute $h^k(I^{k-1}|I^k)$ and $\eta^k(I^{k-1}|I^k)$
13:          **for** each state i aggregated into state I^{k-1} **do**
14:             upgrade $\eta_i = \eta(i|I^k)$ and $h_i = h(i|I^k)$ according to (11)
15:          **end for**
16:       **end for**
17:       **for** all neighbors of class I^k **do**
18:          determine shortest edges
19:       **end for**
20:    **end for**
21:    construct new set of aggregated edges E^k
22:    $G^k := (S^k, E^k)$
23: **until** G^k has only one closed class

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

$$m(i) := \min_j d_{ij}. \qquad (8)$$

Shortest edges correspond to the largest order of magnitude of 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:

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

(9)

Values $h(i|I)$ are computed in Algorithm 1 — they correspond to coefficients h_i in a graph restricted to a closed class I. Lemma 4 below justifies such an aggregation scheme in order to calculate h_i — recall from (7) that to this aim we need $d(\mathcal{F}(i))$. From Lemma 4 (and from an accompanying fact for η_i) one derives correctness of (10) below. Let i denote any state of G such that there exists some tree rooted in i (i. e. $\mathcal{F}(i) \neq \emptyset$). By a **shortest tree** rooted in i we mean any tree f rooted in i such that $d(f)$ is minimal, i.e.,

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

Lemma 4 ([14]). *Let f be a shortest tree in G, rooted in i, and let I be a closed class in G_{min} containing i, $i \in I$. 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):*

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

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 (9). Now, as a new graph G_k we take $(G'_k)_{min}$, whose states are the same as in G'_k and whose edges are the shortest edges in G'_k. Notice that the main loop ends precisely when the partitioning of G_{k-1} results in the only one closed class together with possibly some transient states.

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

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

(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 was proved in [14]:

$$\pi_i(\varepsilon) \sim \eta^1(i|I^1)\varepsilon^{h^1(i|I^1)}\eta^2(I^1|I^2)\varepsilon^{h^2(I^1|I^2)} \cdot \ldots \cdot \eta^n(I^{n-1}|I^n)\varepsilon^{h^n(I^{n-1}|I^n)}\pi(I^n) \qquad (10)$$

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 14 in Algorithm 1), the

correctness of which follows by Lemma 4:

$$h(i|I^k) = h(i|I^{k-1}) + h^k(I^{k-1}|I^k)$$
$$\eta(i|I^k) = \eta(i|I^{k-1})\eta^k(I^{k-1}|I^k)$$

(11)

Coefficients $h^k(I^{k-1}|I^k)$ can be computed using the value of $m(I^k)$ (step 12):

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

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

Finally, we need to explain how to compute effectively $\eta^k(I^{k-1}|I^k)$ in step 12. 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 $\mathbf{L}_k^T \mathbf{x} = \mathbf{b}$, independently inside each class I^k. For the solution the following holds: $x_k(\varepsilon) \sim \eta^k \varepsilon^{h^k}$. Now having already computed h^k and putting some fixed ε, the corresponding coefficients η^k are derived as the solution of the equation:

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

4   Complexity analysis

The upper bound on time complexity of Algorithm 1 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 algorithm 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 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 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 5 ([8]). *The time and space costs of a single phase of aggregation (in both algorithms) are as follows:*

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

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. By S_{total} and T_{total} we denote below 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 stay 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, such that the cost of each phase is polynomial w.r.t. the size of input data for this phase. Assumed that the size of input data decreases at least twice in each phase, the total 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 scheme: $k^{(i)} = (n^{(i)})^{1-\varepsilon}, n_j^{(i)} = (n^{(i)})^{\varepsilon}$, $j \leq k^{(i)}$, $n = n^{(1)} = 2^{\alpha^l}$, where $\alpha = \frac{1}{1-\varepsilon}$ and $0 < \varepsilon < 1$. The size of the problem in the i-th phase of aggregation is $n^{(1-\varepsilon)^i}$. The following inequality holds: $n^{(1-\varepsilon)^i} < \frac{n}{2^i}$, hence the cost of the whole computation is dominated by the cost of the first phase: $T_{total} = \mathcal{O}(m + n^{(1+2\varepsilon)})$.

5 Summary and open problems

We presented a new approximation algorithm based on the combinatorial approach proposed by Pokarowski [14] for computing the stationary distribution of a Markov chain. The complexity of our algorithm was analyzed in detail. Both analytic and experimental results obtained by us classify this new method as a potentially very useful tool in practice. Some case studies of Markov models for communication systems are reported in Appendix.

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

- comparison with GTH procedure shows that precision of approximation computed by our algorithm is on the quite acceptable level of $\log \frac{1}{\varepsilon}$ for the prespecified parameter ε;
- a very promising application of our algorithm is to use the approximate solution it yields as a starting point for some iterative methods e.g. block successive over relaxation (cf. Appendix).

An important unsolved problem is to develop a method of choosing an appropriate value of decomposability parameter ε. In our approach some preprocessing phase is assumed to perform this task.

References

1. Chaiken, S.: A combinatorial proof of all minors matrix tree theorem, *SIAM J. Alg. Disc. Math.*, **vol. 3**, 1982, pp. 319-329.
2. Courtois, P. J.: *Decomposability: Queueing and Computer System Applications*, Academic Press, New York, 1977.
3. 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.
4. 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.
5. 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.
6. 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*.
7. Freidlin, M. I. and Wentzell, A. D.: On small random perturbations of dynamical systems, *Russian Math. Surveys*, 1970, **vol. 25(1)**, pp. 1-55.
8. Gambin, A.: Combinatorial Methods in Approximation Algorithms for Markov Chains with Large State Space. *PhD thesis*, Institute of Informatics, Warsaw University 1999.
9. 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.
10. Hassin, R. and Haviv, M.: Mean passage times and nearly uncoupled Markov chains, *SIAM Journal of Disc. Math.*,1992, **vol. 5**, pp. 386-397.
11. Iosifescu, M.: *Finite Markov Processes and Their Applications*, Wiley and Sons, 1980.
12. 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.
13. Kemeny, J.G. and Snell, J.L.: *Finite Markov Chains*. Van Nostrand, Princeton, 1960.
14. Pokarowski, P. Directed forests and algorithms related to Markov chains, *PhD thesis* Institute of Mathematics, Polish Academy of Sciences, 1998.
15. Pokarowski, P. Directed forests with applications to algorithms related to Markov chains, *Applicationes Mathematicae*, **vol. 26, no. 4**, 1999, pp. 395-414.
16. 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.
17. Stewart, W. J.: A Comparison of Numerical Techniques on Markov Modeling, *Comm. ACM* , **vol. 21**, 1978, pp. 144-152.
18. Stewart, W. J.: *Introduction to the numerical solution of Markov chains*, Princeton University Press, 1994.
19. 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.
20. 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.

Appendix

We report here outcome of our algorithm used to compute stationary distribution of two Markov models: interactive computer system (mentioned already in Introduction) and two-dimensional telecommunication model. To compare our method with existing ones we perform the following experiment: stationary distribution vector is approximated by our algorithm and the obtained solution is used then as a starting point for Block Iterative Over Relaxation (BSOR) method. We measure the speedup (in the number of iteration) compared with BSOR stared from the uniform distribution until the prespecified precision is reached.

5.1   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 [6]:

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

where neither A nor c depend on the iteration step k. In particular, BSOR procedure (see Algorithm 2) is parametrized by:

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

Algorithm 2 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 block partitioning strategies:

Algorithm 2 BSOR method for solving the system $\mathbf{L}^T(R\big|R)\boldsymbol{\pi} = \mathbf{0}$

---

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

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

    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 $\mathbf{x} := (\mathbf{x}_1^T, \dots, \mathbf{x}_N^T)$ where $\mathbf{x}_i^T = (x_{i1}, x_{i2}, \dots, x_{in_i})^T$:

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

until stop criterion succeeds

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) [20].

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 [6, 3]

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 3 for two-dimensional Markov chain model.

5.2 Interactive Computer System

We come back to the model described in Figure 1 (for a detailed treatment see [17]). 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 according to [17]. Recall that the state of the system is coded by a triple (x, y, z) of non-negative numbers, where x denotes the number of users thinking or busy at theirs terminals, y and z denote, respectively, the number of processes in the queue of SM and I/O. Obviously $x + y + z \leq N$. There are at most six transition which can be made from any state, to states obtained by increasing or decreasing one of three coordinates.

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.

For ICS model with 50 users we perform several experiments with iterative method. In this case matrix defining Markov chain is of order $23,426$ with $156,026$ non-zero elements. 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 $||\boldsymbol{\pi}^{(k)}-\boldsymbol{\pi}^{(k-1)}||_\infty \le stop\_tol$ where stopping tolerance $stop\_tol$ 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 1 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.

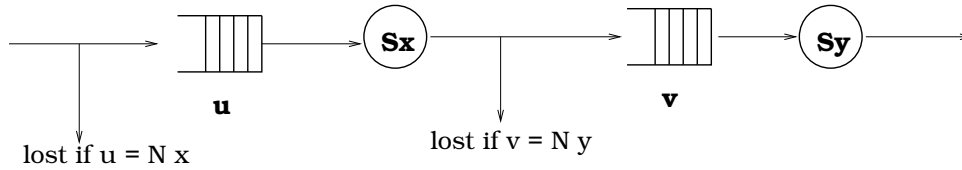## 5.3   A Two-Dimensional Markov Chain Model

Fig. 2. Telecommunication model for 2D example

We consider here a two dimensional Markov chain, studied e.g. in [6, 20]. It is a simple telecommunication model illustrated in Figure 2. 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. The states are pairs (u, v) where u ranges from

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.

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. 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:
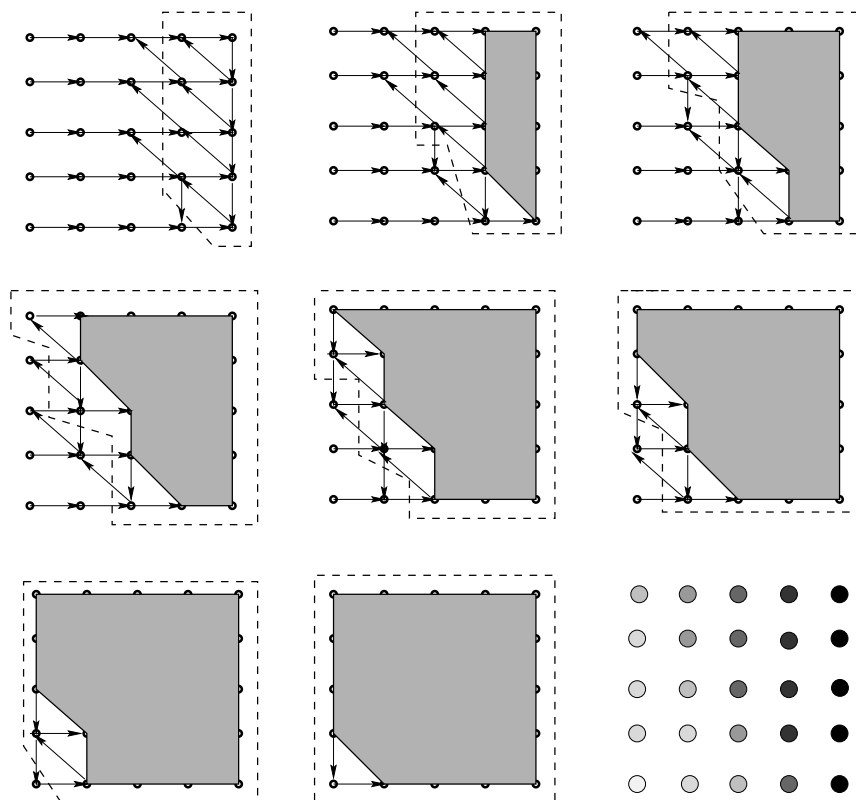
Fig. 3. Aggregation in 2D example.

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 considered.

Figure 3 illustrates the process of aggregation 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 $\mathcal{O}(n^2)$.

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**.