WARSAW UNIVERSITY

FACULTY OF MATHEMATICS,
INFORMATICS AND MECHANICS

Combinatorial Methods in Approximation Algorithms for Markov Chains with Large State Space

Ph.D. Thesis

Anna Gambin

Thesis supervisor:
prof. dr hab. Wojciech Rytter

Warsaw, October 1999

Acknowledgements

First of all, I want to express my gratitude to Sławek Lasota, my husband, for always being available and wiling to discuss problems, for his encouragement and his interest in my work.

My special thanks go to my advisor Wojciech Rytter for his good advice and guidance.

I am most grateful to Piotr Pokarowski for getting me interested in the field and for many helpful discussions on Markov models. Parts of the work presented here were done jointly with him what was both instructive and inspiring experience.

Some of the material presented in this thesis has been done during my scientific stay abroad. I would like to thank all members of Lehrstuhl II University of Dortmund for a nice atmosphere and hospitality, special thanks are to Martin Dietzfelbinger for the cooperation on Multistage Interconnection Networks. I wish to thank Heinz Nixdorf Institut für Informatik in Paderborn for the invitation and hospitality.

The stimulating working atmosphere of Algorithms and Complexity Group made the writing of this thesis a great pleasure. I wish to thank all members of our group and in particular Adam Malinowski for the excellent cooperation.

Partial financial support was provided by the KBN grant 8 T11C 039 15. Parts of this work was based on research supported by Deutscher Akademischer Austauschdienst (DAAD).

Contents

Abstract

We investigate efficiency and structure of algorithms for solving various problems occurring in Markov modeling of complex systems: communication networks, matroids and queueing models. The considered Markov chains can be classified into three different classes depending on their size and complexity.

The first class consists of 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 by Pokarowski [52]. Our original contribution is an improved algorithmic design theoretical/experimental analysis of Pokarowski's method and demonstration of practicality of designed algorithms in typical applications.

The second class of Markov chains is related to problems of counting matroid bases. The problems of *exact* counting is in general $\#P$-complete and the polynomial-time approximation algorithms are possible due to Markov chain simulation. The states of the chain are in this case the bases of the matroid and the transitions correspond to the exchanges of basis by single elements. We study the expansion property for the bases-exchange graph of some special class of matroids (*exchange-preserving*), which have not been investigated before. Our results imply an effective randomized approximation scheme for counting the number of bases. It was shown in [19] that the expansion property holds for the class of so-called *balanced matroids*, including all regular matroids. Our contribution is a corresponding result for all matroids satisfying *exchange-preservation property*, defined in graph-theoretic terms as a condition on the family of *internal exchange graphs*. We apply our results to a subclass of transversal matroids.

The third class of Markov chains considered in the thesis corresponds to the dynamic behavior of massively parallel communication networks, in particular *Multistage Interconnection Networks* (MIN's). We introduce a mathematical framework for the structural description of massively parallel networks. This allows to prove several properties of Markov chains related to such systems, e.g. the ergodicity and relationships between stationary distributions of different models. The method is applied to the analysis of dynamic routing in MIN's with the butterfly structure.

Chapter 1

Introduction

1.1 Overview of the thesis

Theory of Markov chains is related to computational problems at least in two relevant ways.

First, 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. Hence, a starting point is a Markov chain, and a task is to compute some of its characteristics (like stationary distribution), corresponding to the parameters of a modeled system.

Second, certain algorithms solving hard computational problems can be modeled by Markov chains. In such situations, analysis of a chain, especially of its asymptotic behavior, is crucial for the cost estimation.

This thesis is mainly devoted to a study of combinatorial methods for approximation of hard computational problems. In Chapters 2, 3, 4 and 6 we concentrate on relevant characteristics of Markov chains modeling complex computer systems and our aim in Chapter 5 is the analysis of the Markov chain underlying an algorithm counting the number of bases of a matroid.

We investigate three different approaches: *combinatorial graph decomposition*, *graph expansion* and *algebraic decomposition*.

The first approach (i.e. graph decomposition) is applicable when the state space of Markov chain is large enough so that the direct methods (e.g. Gauss elimination for solving linear equations) 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. This idea forms a background for hierarchical aggregation algorithms (see [61] for detailed treatment), used to solve the problems related to Markov chains. To distinguish formally those chains which are tractable in this approach, we use the notion of *powerly perturbed*

Markov chains, introduced in [52]. It turns out that the aggregation approach in that case is attractive, because the structure of transition matrix implies that the standard iterative methods converge very slowly (see discussion in [61, 62] and the references therein).

The presented algorithms use combinatorial properties of *directed forests* in the underlying graph of a Markov chain. The correctness of the algorithms relies on certain facts about forest expansions of solutions of linear equation systems. We analyze the complexity of the algorithms (and demonstrate that it strongly depends on a specific structure of a modeled system). We also report several experiments which have been performed on benchmark examples studied in the literature. The results presented in this part of thesis are an extension and continuation of the work presented in [52] and [26].

If the state space of a Markov chain is too large for the above approach to work, it might be still possible to approximate its stationary distribution if it possesses a property called *rapid mixing.* It is known that this is the case, if the underlying graph has so-called *expansion property.* We show in Chapter 5 that Markov chains induced by base-exchange graphs of *exchange preserving* matroids have the expansion property. This implies the existence of a polynomial time approximation scheme for counting the bases of these matroids. The preliminary version of this result was published as [25].

The last approach (i.e. algebraic decomposition) applies even to Markov chains of exponential size. It enables to define complex Markov chains, which model a system of interest as a composition of small homogeneous components having simpler structure than the whole system. Such an approach gives a possibility of keeping the description of a chain on a reasonable level of complexity, even when the state space of a Markov chain is huge (it is often exponential w.r.t. the size of the system) and the direct description by transition matrix becomes non-effective and intractable. Roughly, our approach enables to "compress" the description of a chain. We develop a calculus for composing smaller components into bigger systems. Most importantly, this calculus permits to deduce properties of composed Markov chains without "decompressing".

The general method is illustrated by an example of the dynamic routing problem in multistage interconnection networks with butterfly interconnection structure. Markov chains are used in this context in modeling of the dynamic behavior of the network. Using our calculus, we are able to relate different models of packet flow thorough the network. These results are an extension of [15] and [24].

We obtain three kinds of results. These are:

- **Numerical results** in Chapter 4 prove effectiveness of the algorithms proposed in [52] for certain characteristics of Markov chains. We present outcome of some case studies; moreover, we estimate time-space complexity of the algorithms in some relevant cases.

- **Complexity results** concerning the Markov chain induced by a base-exchange graph of a matroid (Chapter 5). We show that there exists an effective approximation scheme for counting the number of bases of an exchange-preserving matroid. This applies e.g. to a subclass of transversal matroids.

- **Structural results** in Chapter 6 enable to present in a concise way a structure of a large interconnection network; this facilitates substantially treatment of the Markov chain description and permits to derive their various properties; moreover, this enables to relate different models of dynamic behavior of described systems.

1.2 Markov chain modeling

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

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

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

$$\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{P}, \quad \sum_{i\in\mathcal{S}} \pi_i = ||\boldsymbol{\pi}||_1 = 1, \tag{1.1}$$

is called a **stationary probability distribution** cf. [33, 40]. Intuitively, a Markov chain initialized with a stationary distribution follows this distribution at all points in time. On the other hand, the long-run behavior of a Markovian system starting from any state is revealed through the solution of (1.1). It is crucial to compute a stationary distribution in estimating performance measures for modeled systems. For queueing systems, these measures may be the average number of customers, the mean waiting time, or the blocking probability for a specific queue. In communication systems, these measures may be the total packet loss rate, the probability of an empty system or the communication throughput for packet switching networks (cf. Chapter 6). In any case, the measures may be computed exactly if $\boldsymbol{\pi}$ is available.

Besides stationary distribution, some other characteristics of a Markov chain are also considered, such as *first-passage time* between states or the *number of visits* *in a fixed state before absorption.* To compute them, one has also to solve a system of linear equations similar to (1.1) (cf. Chapter 2).

The most elegant way to deal with (1.1) is to find the analytical formulas for the solution of the system. Unfortunately, it is usually impossible to obtain the analytic solutions for models that incorporate the characteristics sought by modelers; hence the only way is to solve the problem numerically [61]. Problems arise from the computational point of view because of the large number of states which systems may occupy. It is not uncommon for thousands of states to be generated even for simple applications. On the other hand these Markov chains are often sparse and possess specific structure (cf. Section 1.3).

To illustrate the applications of Markov chains let us consider a simple model of an interactive computer system. Figure 1.1 represents the architecture of a time-shared, paged, virtual memory computer. This model was widely studied in the literature [11, 61]; it is considered again in more detail in Chapter 3. 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).

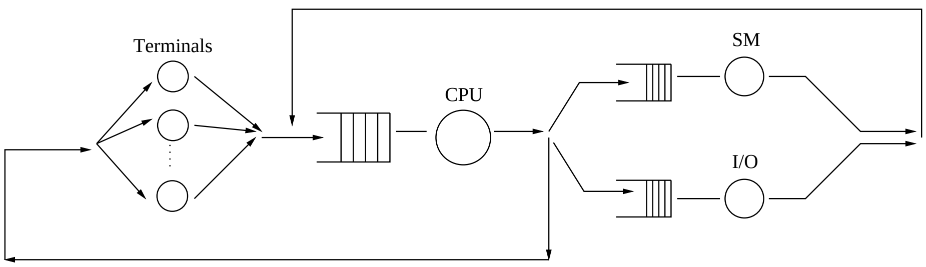

Figure 1.1: Illustration for Interactive Computer System

A queue of requests is associated with each device and the scheduling is assumed to be FCFS (First Come First Served). When the command is generated, the user at the terminal remains inactive until the system responds. Symbolically, a user having generated a command enters the CPU queue. The behavior of the process in the system is characterized by a computing time followed by a page fault, after which the process enters the SM queue, or an input/output (file request), in which case the process enters the I/O queue. Processes which terminate their service at SM or I/O queue return to the CPU queue. Completion of a command is represented by a departure of the process from the CPU to the terminals.

States of the Markov chain corresponding to our model are determined by numbers of processes in all queues. The state space is large but sparse in sense of connections; there are maximum 6 transitions going out from any state. The Figure 1.2 illustrates the structure of a Markov chain (for 3 users in the system): three coordinates correspond to the number of processes in three queues; for example state (0,0,3) denotes three tasks waiting in I/O queue. At the moment we mention

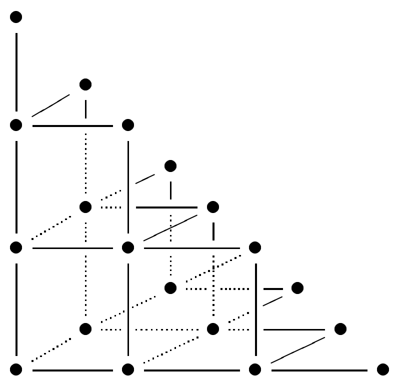

Figure 1.2: The 3D structure of Markov chain model of Interactive Computer System

only how large the state space of the chain is — a more detailed analysis is deferred to Chapter 4. It turns out that the model for only 20 users yields the transition probability matrix of order $1,771$ with $11,011$ non-zero elements and for 50 users the matrix defining Markov chain is of order $23,426$ with $156,026$ non-zero elements.

1.3 Algorithms based on decompositional methods

A lot of research has been done concerning the numerical solutions of some linear equations that occur when one studies Markov chains (see for example [11, 31, 61]). 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 [62]). 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. Among others, algorithm, called GTH (Grassmann-Taksar-Heyman [29]) in the sequel, is adopted in Chapter 3 for dealing with small subsystems arising from decomposition of large Markov chain.

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

To avoid immense fill-in iterative methods can be used, as in that case, the only operation in which the matrices are involved are multiplications by one or more vectors. These operations do not alter the form of the matrix. For these reason, iterative methods (such as Gauss-Seidel iteration or Successive Overrelaxation) have traditionally been preferred to direct methods. On the other hand, a major disadvantage of iterative methods is a very long time often required for convergence to the desired solution. In the case of direct methods the upper bound on the time required to obtain the solution may be determined a priori.

In this thesis we consider *nearly uncoupled* or *nearly completely decomposable* Markov chains. Such chains often arise in queueing network analysis, large scale economic modeling and computer systems performance evaluation. The state space of these chains can be naturally divided into groups of states such that transitions between states belonging to different groups are significantly less likely than transitions between states within the same group.

For solving nearly uncoupled Markov chains, a family of new methods has been proposed recently. They are jointly classified as *iterative aggregation/disaggregation* [38] methods, and based on a very attractive decompositional approach. The idea follows well known *divide and conquer* principle — if the model is too large or complex to analyze, it is divided into separate subproblems. Ideally, subproblems can be solved independently and the global solution is obtained by "merging" the subproblem solutions together. Such an ideal situation occurs when the Markov chain is completely decomposable i.e. its matrix has a block structure in which all off-diagonal blocks has only zero elements. If the nonzero elements in off-diagonal blocks are small compared with those of the diagonal blocks we say about nearly completely decomposable (NCD) Markov chain (see [9, 61, 12]); a formal definition is to be found in Section 2.3.

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

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

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

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

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

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

is often referred to as **degree of coupling** and is taken to be a measure of the decomposability of the matrix (see [53] and discussion in Section 4.1). If it is zero then the Markov chain is reducible (i.e. completely decomposable).

Another concept closely related to decomposability is that of *lumpability*. In [12] it was proved that the concept of NCD Markov chains coincides with quasi lumpability property. We define a Markov chain to have the lumpability property if it can be divided into independent subchains and consequently its state space can be significantly reduced. In many cases, performance measures we are interested in depend on the probability of being in certain groups of states, i.e., the probability needs to be computed at a coarser level. In such a case, solving much smaller lumped chain will give us all needed information. Instead of formal definition, consider a simple example of a lumpable Markov chain defined by the probability transition matrix $\mathbf{P}$:

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

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

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

If we are looking for all stationary probabilities then aggregation algorithms should be used. They are designed to exploit the nearly decomposable structure in one of two ways. In the first approach, some states are excluded from state space and the chain based on remaining state space is solved; the other way is to lump groups of closely related states together and solving the lumped Markov chain, treating each lump as a single state. This latter approach forms also a background of a new class of aggregation algorithms proposed in [52] by P. Pokarowski.

In Chapter 2 a mathematical theory behind the algorithms is sketched. The next chapter contains algorithms themselves. The complexity analysis and several case studies can be find in Chapter 4. Our presentation is much more concise and readable than the original one [52]. We provide, moreover, some minor improvements and report results of experiments we have performed — they are very promising and clearly justify the applicability of the algorithms in practical problems. This novel approach, joining successfully research from different areas such as combinatorics, linear algebra and numerical analysis, leads to algorithms which have several advantages over previous aggregation methods:

- The applicability of traditional aggregation methods is restricted to Markov chains with a regular NCD structure; in particular, the existence of asymptotically transient states (i.e., those states with the outgoing probability of a bigger order of magnitude than the ingoing probability, cf. 2.1) are problematic (such states are very common in large Markov chain resulting from practical examples). Algorithms derived from Pokarowski method work correctly when such states are present.
- Most of decomposition methods considered in the literature are designed only to solve the problem of stationary distribution, and other characteristics of Markov chain are neglected. In contrast, one of the algorithms presented in Chapter 3 allows to compute *mean hitting time*; designing the procedures for other characteristics based on Pokarowski method is discussed latter.

1.4 Approximation scheme for counting matroid bases

Any base of a given matroid can be transformed by means of consecutive exchanges of single element into any other base, possibly in many ways. Thus it seems appropriate to can regard each base as a vertex of a graph and each pair of bases differing by a single exchange as adjacent. The graph obtained is called the **bases-exchange graph** of a matroid [17, 45]. Several classes of matroids have been defined by their exchange property: base orderable and strongly base orderable matroid, binary matroid (possessing an odd number of symmetric exchanges) as well as the class of all matroids (symmetric exchanges and symmetric subset exchanges) [23, 49, 67, 68]. Our main motivation was to study the structure of symmetric exchanges in matroid bases and its correlation with the expansion property of the bases-exchange graph.

The Matroid Expansion Conjecture (see [46]) states that for every matroid its bases-exchange graph has cutset expansion at least 1. On the other hand at least inverse polynomially bounded expansion (w.r.t. the size of matroid) of such graph implies the existence of an effective procedure approximating the number of bases. The approximation scheme is based on a simulation of a Markov chain, corresponding to a random walk on bases-exchange graph. In the case when this Markov chain is rapidly mixing, we have an effective procedure to approximate the number of bases called *fully polynomial randomized approximation scheme* [35, 39]. It is the best we can achieve as the exact enumeration of bases is proved to be #P–hard [2].

For certain classes of matroids researchers managed to prove the rapid mixing property of this random walk [5, 19, 44, 46], but the problem still remains open for the class of all matroids. Matroids satisfy very interesting properties for single-step exchanges, which offers possibilities for searching an argument for rapid mixing similar to *canonical path method* proposed by Jerrum and Sinclair [35]. Unfortunately, these properties are not known to hold for sequences of exchanges, thus it was doubted that the analogue of Jerrum-Sinclair type argument for expansion could be applied in the case of the bases-exchange graph. We revise it by showing that this method is applicable to certain classes of matroids.

In Chapter 5 we consider the class of *strongly base orderable* matroids (see [4, 67]), containing all gammoids [49], and, in particular, all transversal matroids. The base orderable bijection [67] between any pair of bases gives rise to a family of paths having common source and destination base. Our original contribution is a new approach in which strongly base orderable matroids are for the first time investigated in this context. We formulate *exchange-preservation property*, satisfied by a subclass of strongly base orderable matroids. Our main result is to show that this property implies expansion at least $\frac{1}{2}$ of the bases-exchange graph of a matroid. In the proof we apply the method of canonical paths, demonstrating in this way that its applicability is not restricted to Markov chains with regular structure, as suggested in [19, 46]. We also show that the class under consideration contains all binary transversal matroids.

Exchange-preservation property, crucial in our approach, is similar to the strong base orderability condition — while the former property guarantees the expansion of the bases exchange graph, for the latter one this remains a challenging open problem.

1.5 Markov chain description of MINs

Multistage interconnection networks (MIN) have been widely used in constructing high speed communication networks like ATM networks [57] as well as for connecting processors in parallel computing systems. Popular examples of commercial parallel computers using multistage interconnection network for communication between processors are CM-5 (Thinking Machines) and CS-2 (Meiko). Other examples of applications of interconnection networks in today's parallel computers can be found in [6, 28, 54].

MINs consist of $(k \times k)$-switching elements (*switches*), grouped into several stages (cf. Fig. 6.1), and are used to route messages (packets) from input switches to output ones. The MINs in which there exists precisely one path from each input to each output form the broad class of *banyan networks* [27]. For an overview of different architectures of banyan networks see [43]. In Chapter 6 we are concerned with MINs having the butterfly interconnection structure containing (2×2)-switches with input buffers of capacity one (see Fig. 6.1).

A number of papers studied the behavior of butterfly MINs, aiming mainly at evaluation of their performance [14, 41, 48, 69]. In most cases it is assumed that destination addresses of packets are random (with uniform distribution), which simplifies the efficiency analysis and motivates the Markov chain approach.

A large number of results have been achieved for the simplified case (**static model**) when every input link holds exactly one message, which is sent to a randomly chosen destination [3, 66, 55]. A model which is more closely related to the practical situation is so called **dynamic (continuous) model**, in which messages are generated continuously during run-time and routed to random addresses. The most frequently evaluated performance measure in this model is *throughput* (expected number of packets reaching their destinations per time step). Pure analytical results can be (up to now) found only for the **unbuffered** MIN [43, 50]. An analysis of the throughput in this case can take benefit from the fact that consecutive time-steps are independent from each other, because the network has no memory. The situation is much more complex in the case of **buffered networks**. The consecutive steps are not independent from each other, and compositional descriptions of the network's behavior are in general not markovian, hence difficult to analyze. This is the reason why known approaches to buffered networks assume some independence between different parts of the network (single buffers [34], switches [65] or small groups of switches), which makes the analysis much easier. Another possible simplification is to introduce **relaxations**: some packets are deleted when they can not be routed immediately [41, 56]. This resolves the bottom-up dependencies between consecutive stages and enables an approximation of throughput by recursive formulas.

In Chapter 6 we present a formal description of the continuous routing process in the butterfly MIN without any simplifying assumptions or relaxations, i.e. in so called *full blocking model*. As far as we know, no such definitions have been proposed by now. We consider here butterfly MINs with (2×2)-switches having buffers of capacity one (for simplicity called MINs in the sequel), but all the results can be easily repeated for $(k \times k)$-switches ($k \geq 3$) and/or buffers of greater capacity. The main motivation of the work is to formalize the descriptions of packet flow appearing in the literature, as they explicitly or implicitly admit several variations [65, 69]. We describe the network by means of Markov chain, since its global behavior is history-independent. Our approach is *compositional*, i.e. we built the description of the whole network from descriptions of its small components. This is in contrast to former approaches, where behavior of MIN was modeled by description of only small parts of it. In the full blocking model it is not sufficient to investigate only small pieces of the network. Instead, we give a systematic way of composing descriptions of single switches into a Markov chain describing the continuous routing process. An important advantage of the compositional method is its simplicity.

We distinguish three different models of packet flow, being formalizations of the generic full blocking model, differing in a degree of parallelism of switches' actions. We found some links between the models, which show that they are closely related. We develop also a calculus providing operations of composition of smaller Markov chains into bigger ones. This calculus is applied in defining the aforementioned models of packet flow. Our algebraic framework is useful in comparing different models, as it supports compositional approach: we can restrict ourselves to analyzing only small parts of the network instead of the whole one. The proposed method is quite general, independent of specific network topology and protocol of packets routing. This means that usefulness of our approach is not bounded only to description of butterfly MINs: we expect it to be applicable also to other issues dealing with probabilistic description of dynamic behavior of parallel systems, for instance to stochastic neuronal networks. As an example, we apply our notation to *relaxed blocking model* [56]. Studying relaxed blocking and full blocking in the common framework is promising in the context of further work on estimating throughput (cf. Section 7.5).

We study properties of defined Markov chains, and show that all of them are ergodic (cf. Section 6.2). This justifies a steady state (equilibrium state) analysis of MIN's performance, by now tacitly assumed to be correct in many papers. Moreover, as corollaries from relating different models we obtain some links between stationary distributions of the considered Markov chains.

An alternative compositional approach to model parallel systems by means of *Stochastic Automata Networks* (SAN) is described in [61]. Because the analytical results concerning the process (such as evaluating throughput) are hard to obtain when the number of dependencies in the network is large, the numerical solution of SAN are discussed.

Chapter 2

Directed forests method

This chapter is devoted to mathematical preliminaries crucial for aggregation algorithms considered in Chapter 3. The structure of different subjects discussed in this chapter may be explained by the diagram presented in Figure 2.1. We discuss several known results enabling to express the solution of systems of linear equations by means of weighted directed forests (in the underlying graph). Having such *forests expansion* of a characteristics under consideration (such as stationary distribution), we are looking for an effective procedure to compute it. To this end the concept of *powerly perturbed Markov chain* is used (see [52]), as in this case there exist recursive formulas enabling to evaluate effectively the forest expansions. This procedure yields an approximation of characteristics considered.

2.1 Preliminaries

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

The classification of states in a graph follows the Markov chain terminology (see [33, 40] for a detailed treatment of Markov chain theory). State j is **reachable** from state i if there exists a path (nonempty sequence of edges) leading from i to j (we use the short notation $i \to j$). A state i is called **recurrent** if, for any state j, $i \to j$ implies $j \to i$; otherwise, i is called **transient**. We say that states i and j **communicate** ($i \leftrightarrow j$) when $i \to j$ and $j \to i$; state i is **absorbing** if there is no $j \neq i$ reachable from i. When all states communicate, the Markov chain and its graph are called **irreducible**.

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

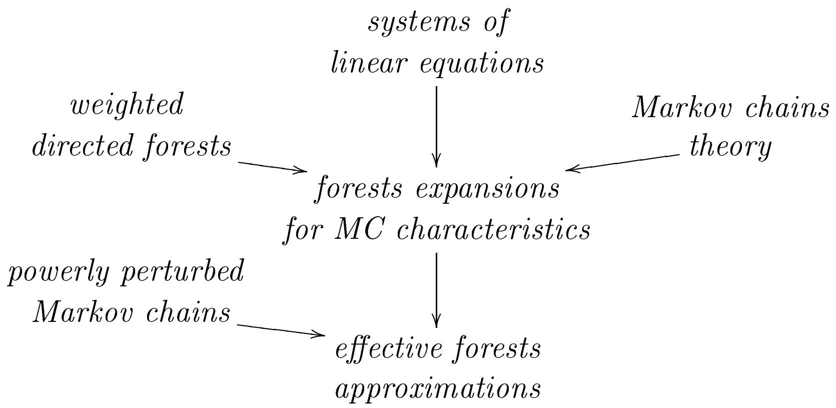

Figure 2.1: The structure of this chapter.

sequel. An underlying graph of each Markov chain has at least one strong absorbing component.

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

Let $\mathcal{F}_G(R)$ denote the set of all forests in G having the root R (the forest is identified with the set of its edges). We will omit the subscript G when it is obvious from the context. For readability, if $R = \{i_1, i_2, \dots i_m\}$ we use the notation $\mathcal{F}(i_1, \dots i_m)$ instead of $\mathcal{F}(R)$. For fixed $i \notin R$ and $j \in R$, $\mathcal{F}_{ij}(R) \subseteq \mathcal{F}(R)$ denotes a set of forests with the root R containing a path from i to j.

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

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

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

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

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

Formally, a Markov chain with the state space $S = \{1, \ldots, s\}$ 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 [21, 33, 40]). Thorough this thesis, we assume that Markov chains are given by a probability transition matrices — the only difference between these two ways of introducing a Markov chain is that given X, we have a distinguished initial distribution X_0, which in not given in a probability transition matrix.

In [52] 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 [52], a **laplacian matrix**, i.e. matrix $\mathbf{L} = (l_{ij})_{i,j=1}^{s}$, $l_{ij} \in \mathbb{C}$ satisfying $l_{ii} = -\sum_{j:\, j\neq i} l_{ij}$ for $i = 1, \ldots, s$.

Denote by $\mathbf{I}$ the identity matrix of size s. Let $\mathbf{P}$ be the transition probability matrix of a Markov chain X. It is easy to verify that matrix $\mathbf{L} = \mathbf{I} - \mathbf{P}$ is a laplacian matrix induced by $\mathbf{P}$. From now on, we assume that the Markov chain is introduced by the **Markov chain laplacian matrix** i.e. laplacian matrix with nonpositive real off-diagonal entries. Such matrices are known in graph theory and combinatorics (see discussion in [52] and the references therein).

For $U, W \subseteq S$ and a square matrix $\mathbf{A}$ of size s, let us denote by $\mathbf{A}(U\big|W)$ 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 $\mathbf{A}(\{i\}\big|\{j\})$. Let $\mathbf{e}_s$ and $\mathbf{0}_s$ denote column vectors with each component equal to 1 and 0 respectively.

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

$$\mathbf{L}(R\big|R)\mathbf{x} = \mathbf{b} \tag{2.1}$$

$$\mathbf{L}^T(R\big|R)\mathbf{x} = \mathbf{b}, \tag{2.2}$$

where R is a subset of states and $\mathbf{b}$ is a nonnegative vector of size $s - |R|$. As an example of (2.2), consider computing the stationary distribution. By (1.1), 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:

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

Assuming for simplicity that the states numbering implies $\pi_s > 0$, one of possible ways of solving (2.3) is to compute the solution $\mathbf{a}_s$ of a system of the form (2.2):

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

and then to normalize the vector $\mathbf{a}^T := (\mathbf{a}_s^T, 1)$. On the other hand solving system (2.2) can be reduced to computing the solution of (2.3) for appropriately defined laplacian $\mathbf{L}$.

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

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

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

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

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

The following equality is easy to verify (cf. [33]):

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

it can be reformulated in the matrix form (2.1) as follows ($\mathbf{m} = (m_i(R))_{i \in S \setminus R}$):

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

2.2 Markov chain tree theorem

We express the solution of a system of linear equations as a rational function of directed forest weights (called the **forest expansion**) [7]. For stationary distribution this is formulated in Markov Chain Tree Theorem (Theorem 2.2.1). We also give analogous expansions for other characteristics. Unfortunately, expressions obtained could contain an exponential number of summands. In Section 2.4 we present recursive formulas yielding the effective procedures for evaluating forest expansions in the special case of powerly perturbed Markov chains.

Without loss of generality assume that states are numbered in such a way that $R = \{s - |R| + 1, \ldots, s\}$; we assume that $R \neq \emptyset$. For a laplacian matrix $\mathbf{L}$ of size $s \times s$, a set $R \subseteq S$ and $i, j \notin R$ consider forests in graph $G(\mathbf{L})$. The following facts have been proved in [20] and [52], respectively:

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

(we omit subscripts and write $\mathcal{F}(R)$ instead of $\mathcal{F}_{G(\mathbf{L})}(R)$, etc.). Now, consider the system of linear equations of the form (2.2):

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

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

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

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

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

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

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

Consider again the problem of computing the stationary distribution $\boldsymbol{\pi} = (\pi_1, \dots, \pi_s)^T$ of a Markov chain defined by the laplacian matrix $\mathbf{L} = (l_{ij})_{i,j=1}^s$. Assume that an underlying graph of the Markov chain has exactly one absorbing strong component and the states are numbered in such a way that $\pi_s > 0$. Now, putting $\mathbf{A} = \mathbf{L}_{ss}$, $\mathbf{x} = (\frac{\pi_1}{\pi_s}, \dots, \frac{\pi_{s-1}}{\pi_s})^T$ and $\mathbf{b} = (-l_{s,1}, \dots, -l_{s,s-1})^T$ in (2.5) leads to the following theorem (cf. equation (2.1)).

Theorem 2.2.1 (Markov chain tree theorem) *If the underlying graph of a Markov chain has exactly one absorbing strong component, then the stationary distribution is given by:*

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

This was proved independently by many authors, among them [22, 58]; Aldous [1] calls it "the most often rediscovered result in probability theory". A proof by Matrix Tree Theorem was proposed in [52]. Analogously, one obtains a forest expansion for the mean hitting time, for a set $R \subseteq S$:

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

Other characteristics widely studied in the literature are:

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

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

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

They can be computed by solving systems of the form (2.1). For a Markov chain determined by laplacian matrix $\mathbf{L}$, such that graph $G(\mathbf{L})$ contains a spanning forest rooted in a fixed subset R of states, there exist analogously to Theorem 2.2.1 the following forest expansions, for $i, j \in S \setminus R$, $k \in R$:

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

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

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

2.3 Powerly perturbed Markov chains

Nearly completely decomposable (NCD) Markov chains (see [9, 12, 61]) are defined by laplacian matrices that can be ordered so that the matrix has a block structure in which the nonzero elements of the off-diagonal blocks are small compared with those of the diagonal blocks. Such matrices often arise in queueing network analysis, large scale economic models and computer systems performance evaluation.

Formally, NCD Markov chain is defined as a family of Markov chains $\{\mathbf{L}(\varepsilon),\ \varepsilon \in (0, \varepsilon_1)\}$, indexed by a small parameter $\varepsilon < \varepsilon_1$, for some constant $\varepsilon_1 > 0$:

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

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

In the literature one can find some generalizations of NCD Markov chains, aiming in expressing several different orders of magnitude of interaction strength (see for example [30]):

1. The **linearly perturbed** Markov chains

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

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

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

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

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

In [52] a wider class of perturbed Markov chains have been defined, subsuming both classes above. For given functions $A, B: \mathbb{R} \to \mathbb{R}$, the notation $A(\varepsilon) \sim B(\varepsilon)$ means that:

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

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

A family $\{\mathbf{L}(\varepsilon) = (l_{ij}(\varepsilon))_{i,j=1}^{s},\ \varepsilon \in (0, \varepsilon_1)\}$ of laplacian matrices of size $s \times s$ is a **powerly perturbed** Markov chain, if there exist matrices $\Delta = (\delta_{ij})_{i,j \in S}$, 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 Δ and $\mathbf{D}$ as follows:

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

We assume that $d_{ij} = \infty$ if and only if $\delta_{ij} = 0$. In the following, we use also the concept of **powerly perturbed** nonnegative vector which is defined as the family $\{\mathbf{b}(\varepsilon),\ \varepsilon \in (0, \varepsilon_1)\}$ of nonnegative vectors of size u, such that for some vectors $\zeta = (\zeta_i)_{i=1}^{u}$ and $\mathbf{z} = (z_i)_{i=1}^{u}$, with $\zeta_i \geq 0$, $z_i \in \mathbb{R} \cup \{\infty\}$, for $i = 1, \ldots, u$, the following holds:

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

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

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

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

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

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

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

Fact 2.3.1 *Consider a powerly perturbed Markov chain defined by* $\mathbf{L}$*, with matrices* Δ *and* $\mathbf{D}$ *such that (2.8) above holds; furthermore let* f *and* $\mathcal{F}$ *be a forest and a set of forests in* $G^*(\mathbf{D})$*. Then:*

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

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

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

Theorem 2.3.2 ([52]) *Let matrices* Δ *and* $\mathbf{D}$ *be such that (2.8) 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 (2.9) holds, for a powerly perturbed vector* $\mathbf{b}$*. Suppose that there exist a forest with the root* R *in* $G^*(\mathbf{D})$*. Then the following hold:*

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

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

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

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

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

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

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

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

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

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

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

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

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

where

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

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

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

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

Unfortunately, all obtained expressions for asymptotic coefficients are computationally non-tractable, at least directly, because of their exponential length. We discuss the aggregation approach, yielding effective and accurate procedures for computing the asymptotic coefficients and approximate values of the interesting characteristics of NCD Markov chain.

2.3.1 Asymptotic coefficients and exact solutions

In Chapter 3 we describe algorithms which compute asymptotic coefficients $\mathbf{h}$ and η from Theorem 2.3.2. Now, we explain how they can be used to obtain the approximation of e. g. stationary distribution vector (case *(1)* from Theorem 2.3.2).

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

1. construct matrices $\mathbf{\Delta}$ and $\mathbf{D}$ such that:
$$-l_{ij} = \delta_{ij}\varepsilon^{d_{ij}};$$
2. run Algorithm 2 to compute vectors η and $\mathbf{h}$;
3. set $\pi_i(\varepsilon) := \eta_i \varepsilon^{h_i}$.

This algorithm is correct in a sense that for sufficiently small ε, the computed value $\boldsymbol{\pi}(\varepsilon)$ can be arbitrarily close to an exact value. Even more, when $\varepsilon < \min_{ij} -l_{ij}$, we have $d_{ij} = 0$ (for all i, j), hence $\mathbf{L} = \Delta$ and Algorithm 2 gives the exact solution (in particular, $h_i = 0$, for all i). In that case, no aggregation can be done and algorithm runs a direct (GTH) method. On the other hand, larger ε's allow to profit from a specific block structure of a laplacian matrix, which improves efficiency. Hence, there exists a tradeoff between the time and space efficiency of the algorithm and a precision of the approximation.

2.4 Fast computation of forest expansions

The algorithms discussed in Chapter 3 use the combinatorial approach to compute the asymptotic coefficients (vectors $\mathbf{h} = (h_i)_{i=1}^s$ and $\boldsymbol{\eta} = (\eta_i)_{i=1}^s$ from Theorem 2.3.2), hence they approximate the solution of systems $\mathbf{L}(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon)$ and $\mathbf{L}^T(R)(\varepsilon)\mathbf{x}(\varepsilon) = \mathbf{b}(\varepsilon)$. The algorithms reduce the size of state-space of a Markov chain by lumping together closely related states. This process is repeated in the consecutive phases of aggregation during which some graphs induced by matrices $\mathbf{D}$ and $\mathbf{\Delta}$ are considered. The algorithms group states of the Markov chain belonging to the same closed class of the graph and solve the system of linear equations restricted to this class. A smaller size, hence tractable, system of equations can be solved by a direct method. The solution of this system is used to upgrade the values of interested characteristic computed for each state of the original Markov chain. Such an iterative upgrade scheme, enabling to calculate effectively the asymptotic coefficients in the case of stationary distribution and mean hitting time, is discussed in detail in Chapter 3.

The task of computing exponents h_i is of quite different nature than the task of computing the coefficients η_i. While the former can be performed using purely combinatorial methods (hence precisely), the latter uses a procedure of solving a system of linear equations, exposed to numerical errors. Although calculating coefficients η_i is of crucial importance, in the sequel we concentrate on h_i only (we explain how to calculate η_i in Section 3.2.1). We overview the process of aggregation (it is treated again in more detail in Chapter 3) and state some facts on **shortest forests**, useful in computing exponents h_i.

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

$$m(i) := \min_j d_{ij}. \tag{2.12}$$

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

$$\begin{aligned} d_{IJ} &:= \min_{i \in I, j \in J} (d_{ij} + h(i|I)) \quad \text{where} \\ h(i|I) &:= \max_{k \in I} m(k) - m(i). \end{aligned} \tag{2.13}$$

Values $h(i|I)$ are computed in Algorithm 2 — they correspond to coefficients h_i in a graph induced by a closed class I. Lemma 2.4.1 below justifies such an aggregation scheme in order to calculate h_i — recall from (2.10) that to this aim we need $d(\mathcal{F}(i))$. From this lemma (and from an accompanying fact for η_i) one derives correctness of (3.5) in Section 3.2.1. 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.,

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

Lemma 2.4.1 ([52]) *Let f be a shortest tree in G, rooted in i, and let I be a closed class in G_{min} containing i, i.e. $i \in I$. Let f_I be a shortest tree in the subgraph of G_{min} induced by I, rooted in i. Moreover, let f' be a shortest tree in G', rooted in I. The following holds (for simplicity, we apply here notation $d(\_)$ to graph G' as well):*

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

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

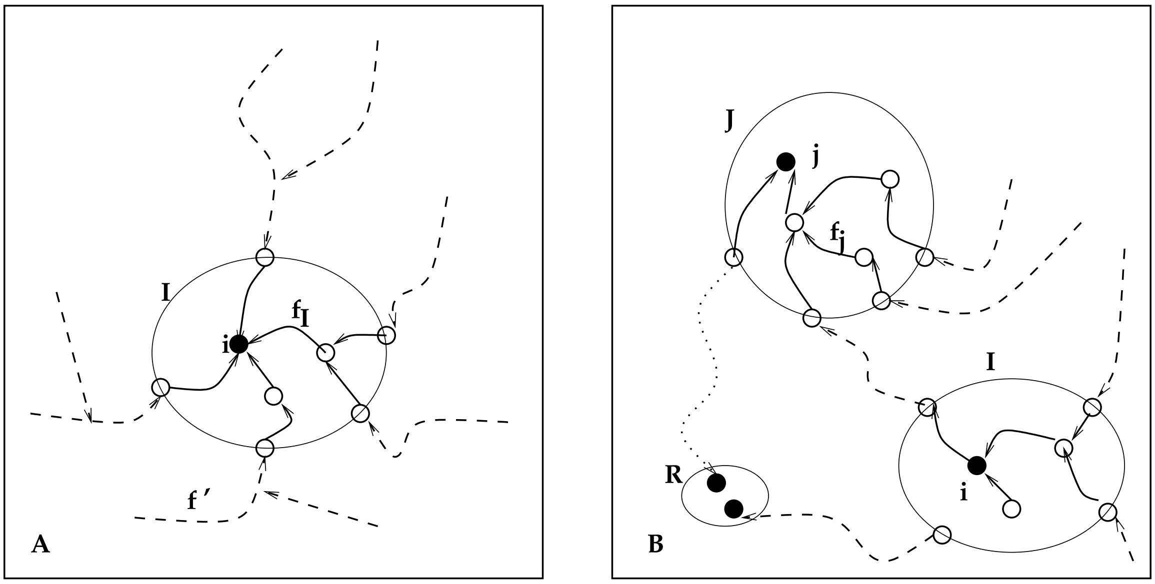

Figure 2.2: A: shortest tree rooted in i from Lemma 2.4.1, B: shortest forest rooted in $R \cup j$ from Lemma 2.4.2

Lemma 2.4.2 ([52]) *Let R be a subset of states of G s.t. $\mathcal{F}(R)$ is nonempty. Let $I \neq J$ be closed classes or transient states in G_{min} s.t. R and $I \cup J$ are disjoint. Fix a state $i \in I$. Let f_j be a shortest tree in the subgraph of G_{min} induced by J, rooted in j. The following holds (similarly as before, we extend here notation $d(\_)$ and $\mathcal{F}_{\_}(\_)$ to graph G'):*

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

Figure 2.4 illustrates concept of shortest forests factorization from Lemmas 2.4.1 and 2.4.2.

Chapter 3

Combinatorial aggregation algorithms

We present two algorithms — one approximates the stationary distribution and the other gives the approximation of the mean hitting time. They represent two classes of methods, designed to calculate approximation of solutions of linear systems build up of transpose or non-transpose matrices (cf. Section 2.1). The main idea is to reduce the size of a Markov chain state space by performing the aggregation of closely related states. During the computation process one needs to solve small Markov chains, corresponding to one aggregate (i.e. to solve the system of linear equations of the form $\mathbf{L}(R\big|R)\mathbf{x} = \mathbf{b}$ or $\mathbf{L}^T(R\big|R)\mathbf{x} = \mathbf{b}$). To this aim the GTH method is adopted below; it can solve both kinds of systems, while keeping the numerical errors on a relatively low level (cf. Section 3.1).

3.1 Direct methods

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

The state-reduction method (called here also GTH algorithm) was introduced in [29] by Grassmann, Taksar and Heyman, where it was applied to the equations yielding the stationary vector; applications to other characteristics of a Markov chain are proposed among others in [42, 52]. This approach has a probabilistic interpretation: each iteration can be viewed as a construction of a different Markov chain with one state less. Moreover, from its solution the solution of the original problem can be obtained. Numerically, this algorithm behaves surprisingly well: its numerical features are discussed in many papers (see [8, 31, 11]; an exhausting analysis can be found in [52]). The rest of this section is devoted to explain the idea of the GTH approach; we use here the notation from Chapter 2 and the presentation is based on [11, 52]. Consider the system of equations:

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3.1}$$

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

$$\mathbf{L}^T(R\big|R)\mathbf{x} = \mathbf{b}, \tag{3.2}$$

$\mathbf{b}$ is nonnegative vector of size $u := s - |R|$, s is the number of states of a Markov chain and R denotes subset of states forming root of the directed forest in the graph induced by the laplacian $\mathbf{L}$. From Section 2.2 (c.f. equation(2.4)) we know that the matrix $\mathbf{L}(R\big|R)$ is invertible. Assume that we are interested in computing the stationary probability vector i.e:

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

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

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

being an instance of (3.2) and then normalize the vector $\mathbf{a}^T := (\mathbf{a}_s^T, 1)$.

GTH algorithm has two variants: for a transposed and nontransposed laplacian $\mathbf{L}$, respectively. Below we describe the GTH variant dealing with transposed system (such as for the calculation of the stationary distribution); analogical approach, proposed for nontransposed variant in [42, 52], is mentioned later.

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

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

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

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

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

where

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

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

$$\begin{aligned} \mathbf{e}^T\mathbf{A} = \mathbf{e}^T(\mathbf{B}_1^{-1}\mathbf{B}_1)\mathbf{A} = 0, \\ \mathbf{e}^T\mathbf{B}_1^{-1} = (1, 1, \dots, 1) \begin{bmatrix} 1 & \mathbf{0} \\ \frac{\mathbf{w}}{a_{11}} & \mathbf{I} \end{bmatrix} = (0, 1, \dots, 1), \\ (0, 1, \dots, 1)(\mathbf{B}_1\mathbf{A}) = 0 \quad \Longrightarrow \quad \mathbf{e}^T\mathbf{A}_{22}^* = 0 \end{aligned}$$

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

$$u_k = \sum_{j=k+1}^{u+1} w_j, \tag{3.3}$$

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

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

As a conclusion from the above discussion we formulate Algorithm 1. It is presented in a more general setting i.e. for solving (3.2); recall that $l_{iR} := \sum_{j\in R} l_{ij}$ for $i = 1, \dots, u$.

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

$$\mathbf{L}(R|R)\mathbf{x} = \mathbf{b}, \tag{3.4}$$

where $\mathbf{b}$ is nonnegative vector of size u. Systems like 3.4 are used to determine for instance the vector of mean hitting time (cf. Section 3.2.2) — from the equations $\mathbf{L}(R|R)\mathbf{m} = \mathbf{e}$, as well as the distribution matrix at the moment reaching the set R — from the equations $\mathbf{L}(R|R)\mathbf{A} = -\mathbf{L}(R|S \setminus R)$. The only difference here is dealing with non-transposed matrix; now the pivots are calculated as a upper-diagonal row sums.

3.2 Approximation algorithms

We present here two combinatorial aggregation algorithms for solving large and sparse Markov chains. The algorithms themselves are already formulated in [52] — the con-

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

for $k = 1$ to u **do**
  $l_{k,u+1} := l_{kR}$
  $l_{u+1,k} := b_k$
end for
for $k = 1$ to u **do**
  $u_k := -\sum_{j=k+1}^{u+1} l_{kj}$
  for $i, j = k+1,\ i \neq j$ to $u+1$ **do**
    $l_{ij} := l_{ij} - l_{ik}l_{kj}/u_k$
  end for
end for
for $k = u$ downto 1 **do**
  $x_k := \left(l_{u+1,k} - \sum_{j=k+1}^{u} l_{jk}x_j\right)/u_k$
end for

tribution here consists of the detailed discussion justifying the correctness of Algorithm 4, a novel approach to the presentation of algorithms (which improve substantially readability) and the complexity analysis of algorithms. Moreover, we found and fixed some minor inaccuracies in the original presentation. Both algorithms have been (effectively) implemented by us, which allow us to compare this new method with existing ones, especially with a direct method and block successive overrelaxation.

3.2.1 Stationary distribution

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

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

As an illustrative example, we consider here the stationary distribution — analogous results for Markov chain characteristics defined by the system with non-transposed $\mathbf{L}$ are discussed in Section 3.2.2.

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

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

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

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

Initially, define the graph G^0 as G_{min}, where $G = G^*(\mathbf{D})$. So, in the first step we start with the subgraph of $G^*(\mathbf{D})$ consisting of all shortest edges outgoing from every vertex. For $k = 1, 2, \ldots$, we define inductively $G'_k = aggr(G_{k-1})$. Recall that the states of G'_k are all closed classes and all transient states in G^{k-1}; the set of edges linking the aggregated states is constructed as in (2.13). Now, as a new graph G_k we take $(G'_k)_{min}$, whose states are the same as in G'_k and whose edges are the shortest edges in G'_k.

Algorithm 2 Calculate asymptotic coefficient η and h, i.e. approximate the stationary distribution $\pi_i = \eta_i \varepsilon^{h_i}$.

construct $G^0 = (S^0, E^0)$
$k := 1$
repeat
  find partition of G^{k-1} into closed classes
  construct S^k
  for each closed class in S^k, say I^k **do**
    construct laplacian L_k
    compute stationary distribution i. e. solve the system $\mathbf{L}_k^T \mathbf{x} = \mathbf{b}$
    compute $m(I^k)$ (cf. (2.12) in Section 2.4)
    for each aggregated state I^{k-1} in I^k **do**
      compute $\eta^k(I^{k-1}|I^k)$ and $h^k(I^{k-1}|I^k)$
      for each state i aggregated into state I^{k-1} **do**
        upgrade $\eta_i = \eta(i|I^k)$ and $h_i = h(i|I^k)$ according to (3.6)
      end for
    end for
    for all neighbors of class I^k **do**
      determine shortest edges
    end for
  end for
  construct new set of aggregated edges E^k
  $G^k := (S^k, E^k)$
  $k := k + 1$
until G^k has only one closed class
set $\pi_i := 0$, for all transient states i in G

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

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

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

(T denotes the subset of transient states) containing i during the consecutive phases of aggregation. We assume that the graph G^n, resulting from the n-th phase, is irreducible. Denote by $\eta(i|I^k)$ and $h(i|I^k)$ coefficients η_i and h_i computed in the subgraph restricted to some closed class I^k. Following this convention, $\eta^k(I^{k-1}|I^k)$ and $h^k(I^{k-1}|I^k)$ correspond to the η and h coefficient for the aggregated state I^{k-1} computed during the k-th phase for the subgraph of G^k restricted to a closed class I^k. The following recursive relation was proved in [52]:

$$\pi_i(\varepsilon) \sim \eta^1(i|I^1)\varepsilon^{h^1(i|I^1)}\eta^2(I^1|I^2)\varepsilon^{h^2(I^1|I^2)} \cdot \ldots \cdot \eta^n(I^{n-1}|I^n)\varepsilon^{h^n(I^{n-1}|I^n)}\pi(I^n) \tag{3.5}$$

where $\pi(I^n) = 1$ is the stationary probability of being inside the class I^n. This relation holds due to the iterative upgrade scheme for asymptotic coefficients (step 12), the correctness of which follows by Lemma 2.4.1:

$$\begin{aligned} h(i|I^k) &= h(i|I^{k-1}) + h^k(I^{k-1}|I^k) \\ \eta(i|I^k) &= \eta(i|I^{k-1})\eta^k(I^{k-1}|I^k) \end{aligned} \tag{3.6}$$

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

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

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

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

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

3.2.2 Mean hitting time

Procedure 3 Aggregation

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

We describe here an approximation algorithm based on Pokarowski method for another important characteristics of the Markov chain — the mean hitting time. According to Section 2.1, this is an interesting example of the linear equations with a non-transposed matrix. It is parametrized by a subset of states $R \subseteq \mathcal{S}$ and the task is to approximately calculate the expected number of steps before reaching this set. Formal definition can be found in Chapter 2; recall that $\mathbf{m}_i(R)$ satisfies the following equality:

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

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

$$\mathbf{L}(R|R)\mathbf{m} = \mathbf{e}, \tag{3.7}$$

where $\mathbf{e} = (11\ldots1)^T$. Hence, computing mean hitting time corresponds to approximating the solution of the system $\mathbf{L}(R|R)\mathbf{x} = \mathbf{e}$. The following forest expansion is given for the vector $\mathbf{m}_i(R)$:

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

In the case of a powerly perturbed Markov chain it was proved that:

$$m_i(R)(\varepsilon) \sim \beta_{iR}\varepsilon^{b_{iR}}, \tag{3.8}$$

where asymptotic coefficient β and b are defined as follows:

$$\begin{aligned} a_i :=& \min_{j \notin R}[d(\mathcal{F}_{ij}(R \cup \{j\}))], &(3.9)\\ b_{iR} :=& a_i - d(\mathcal{F}(R)), &(3.10)\\ \beta_{iR} :=& \frac{\sum_{j \notin R: d(\mathcal{F}_{ij}(R\cup\{j\}))=a_i} \delta(\mathcal{F}_{ij}(R \cup \{j\}))}{\delta(\mathcal{F}(R))}. &(3.11)\end{aligned}$$

The algorithm for the mean hitting time performs the aggregation process starting from a graph $G^0 = (S^0, E^0)$, which differs slightly from that in Algorithm 2. Namely, all states from R are replaced by a some single new state u_R; it has no outgoing edges and for any $i \notin R$, an edge from i to u_R is given by $\min_{j\in R} d_{ij}$ (this means that we are interested in reaching *any* of states of R). Moreover, we add one more state u^*, and set $d_{iu^*} := 0$, $\delta_{iu^*} := 1$ for all $i \notin R \cup \{u_R\}$. Intuitively, this new state corresponds to the right-hand side vector e in equation (3.7), forming a fragment of the matrix $(comp(\mathbf{L}(R|R)^T, e))^T$. Notice that edges leading to u^* are ignored, during the construction of a new set of aggregated states in steps 4 and 5. However they are updated in step 17 according to the same rule as other edges (cf. formula (2.13) in Section 2.4). Finally, if the graph constructed by now we denote by G, as G_0 we take G_{min}, similarly as previously.

Algorithm 4 consists of two phases. The first one runs the aggregation scheme, similarly as before; however, due to the artificial states u_R and u^*, being sinks in the graph, the aggregation can leave several closed classes of the original graph not lumped together. Then, the second one calculates coefficients b_{iR} and β_{iR} for these closed classes. Computed values are then propagated throughout each class.

3.2.3 Improvements and extensions

The construction of laplacian $\mathbf{L}$ in step 4 of Algorithm 4 requires taking into account all edges between aggregated states, not only the shortest ones. Hence, coefficients β_{iR} computed by this algorithms do not coincide precisely with those needed in (3.8); fortunately, the computed values, say β'_{iR}, are asymptotically equivalent:

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

Originally, in step 4 of Algorithm 4 only shortest edges were considered — the necessity of the modification was observed during experiments. A similar modification of Algorithm 2 is also possible. The procedure solving a system of equations in a closed class (while calculating $\boldsymbol{\eta}$) can take into account all edges between vertices inside this

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

Aggregation
for each closed class in S^k, say I^k, s.t. $I^k \neq u^R$ **do**
  compute $b_{I^kR} := d_{I^ku^*} - m(I^k)$
end for
repeat
  remove from S^k classes I^k having minimal b_{I^kR} value, denote the set of removed classes by M
  for each $I^k \in M$ and any state $i \in S^0$ belonging to I^k **do**
    $b_{iR} := b_{I^kR}$
  end for
  for each I^k remaining in S^k **do**
    let $n(I^k) := \min_{J^k \in M}(d_{I^kJ^k} - m(I^k) + b_{J^kR})$
    compute $b_{I^kR} := \min(b_{I^kR}, n(I^k))$
  end for
until the set $S^k = \{u_R\}$
construct laplacian $\mathbf{L}$, taking S^k as the set of states
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
for each $I^k \in S^k$ different than u_R and each state $i \in S^0$ belonging to I^k **do**
  compute $\beta_{iR} := x_{I^k}\varepsilon_0^{-b_{I^kR}}$
end for

class rather then only the shortest ones. After such a modification the algorithm computes no more the asymptotic coefficients — however, it gives a closer approximation of stationary distribution. In certain situations it can significantly improve the accuracy of approximation.

Algorithm 4 can be easily applied to calculation of other characteristics of a Markov chains, such as: $\mathbf{p} = p_{ik}(R)$ — the probability distribution (in R) in the hitting time and and $\boldsymbol{\mu} = \mu_{ij}(R)$ — the mean number of visits before absorption.

Chapter 4

Analysis of algorithms and case studies

We analyze combinatorial aggregation algorithms, in particular we estimate time and space complexity of Algorithms 2 and 4. Then we point out that the efficiency of algorithms depends strongly on the value of parameter ε which determines the number of phases of algorithm and the structure of aggregation process in each phase. A substantial part of this chapter is devoted to a detailed analysis of several case studies investigated in the literature. We compare our combinatorial aggregation method with a direct GTH method in the case of small size examples and with the block successive over relaxation (BSOR) in the case of large models. The latter one was chosen because it is widely accepted as the best known method calculating the approximate solution of problems related to large and sparse Markov chains (see for example [13, 51]).

4.1 Complexity analysis

The upper bound on time complexity of Algorithms 2 and 4 presented in previous chapter is $\mathcal{O}(n^3)$, where n is the number of states. The upper bound on memory needed is $\mathcal{O}(n^2)$. However, the complexity of algorithms depends strongly on the structure of a Markov chain under consideration. In this section we study in detail some important cases. The main conclusion is that if we can profit from a specific structure of a matrix (e.g. if we choose an appropriate ε), time $\mathcal{O}(n^2)$ is sufficient. Moreover, when a matrix is sparse, i.e. the number of edges m is significantly smaller that $\mathcal{O}(n^2)$, the algorithms use only space $\mathcal{O}(n+m)$. This is crucial since matrices appearing in applications are often sparse and it is not rare that $m = \Theta(n)$. These estimates strongly motivate further studies of methods to establish an appropriate value of parameter ε.

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

Theorem 4.1.1 *The time and space costs of a single phase of aggregation (in both algorithms) are as follows:*

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

Proof: The time cost of $n + m$ is due to Tarjan's algorithm for finding strongly connected components, used to determine closed classes. Construction of aggregated edges (i.e. edges in the graph used in the next phase) can be performed in time proportional to m. Finally, $\sum_{i=1}^{k} {n_i}^3$ is the cost of running GTH procedure k times, for each closed class separately.

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

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

1. $p = 1, k = 1$; a pessimistic case — all states are aggregated during the first step; GTH procedure is performed on the whole matrix; $T_{total} = \mathcal{O}(n^3), S_{total} = \mathcal{O}(n^2)$.

2. $p = n - 1$, $k^{(i)} = n - i$; "lazy" aggregation — after step i there are still $n - i - 1$ isolated states; $T_{total} = \mathcal{O}(n^2), S_{total} = \mathcal{O}(n + m)$.

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

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

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

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

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

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

Degree of coupling

There are several approaches aiming at defining a measure of decomposability of a transition matrix (see for example [53]). Such a measure is usually closely related to the parameter ε and could be probably used to find an optimal value for ε. Moreover, a right value of ε can significantly improve precision of approximation as well as time and space needed for the algorithm.

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

4.2 Numerical experiments

In this section we describe the results obtained when combinatorial aggregation algorithms are used to compute stationary distribution and mean hitting time of several Markov models. In Section 4.2.1 we concentrate on small examples which appear frequently in the literature (cf. [9, 61]). Although the sizes of the matrices considered are quite small, it is still instructive to examine the effect of using our algorithms in such cases. Two problems investigated in Sections 4.2.3 and 4.2.4 are real life examples and had been extensively studied by many authors (see e. g. [13, 11, 64]).

For each small example considered in the sequel we compute the relative error of the solution (denoted by $\boldsymbol{\pi}^*$ and $\mathbf{m}^*$) compared with outcome of the GTH procedure (vectors: $\boldsymbol{\pi}$ and $\mathbf{m}$, respectively). Errors are computed w.r.t. GTH algorithm, despite that it is also exposed itself to numerical errors. GTH is suitable for this purpose as it has an *a priori* error estimation [8, 52].

Another two measures of algorithms accuracy correspond to a mean number of correct most significant digits and are given by the following formulas (n is the size of the matrix):

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

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

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

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

[1]for a formal definition see conditions $6.1 - 6.4$ in [61] pp. 335.

4.2.1 Small examples

Courtois matrix

The first problem considered in this section is the 8×8 Courtois matrix studied already in [9] 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.

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

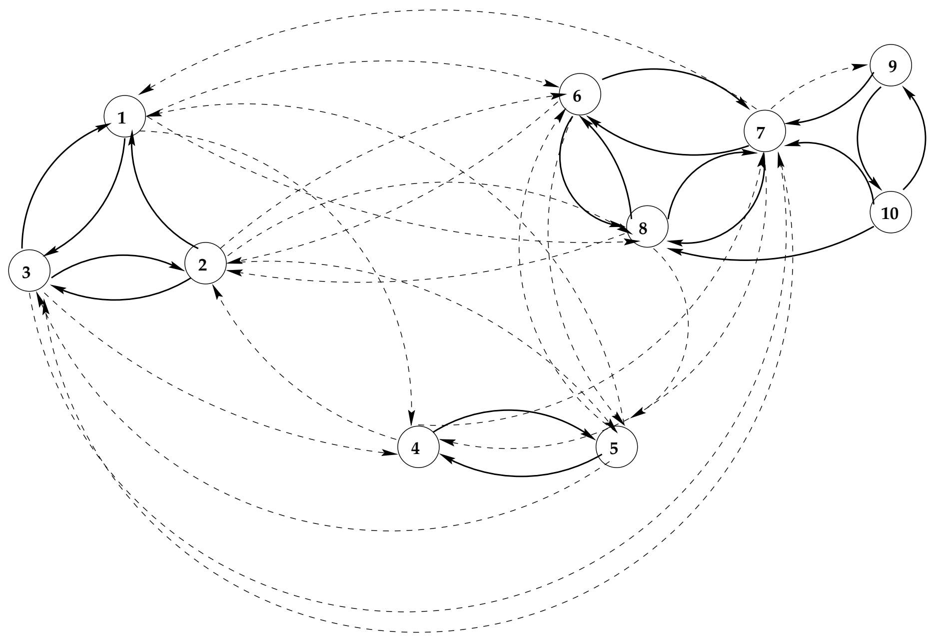

Figure 4.1: Courtois matrix with two asymptotically transient states.

Courtois matrix with two additional states

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

10. Second step of aggregation results in one closed class lumping together all states (cf. Fig.4.1).

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

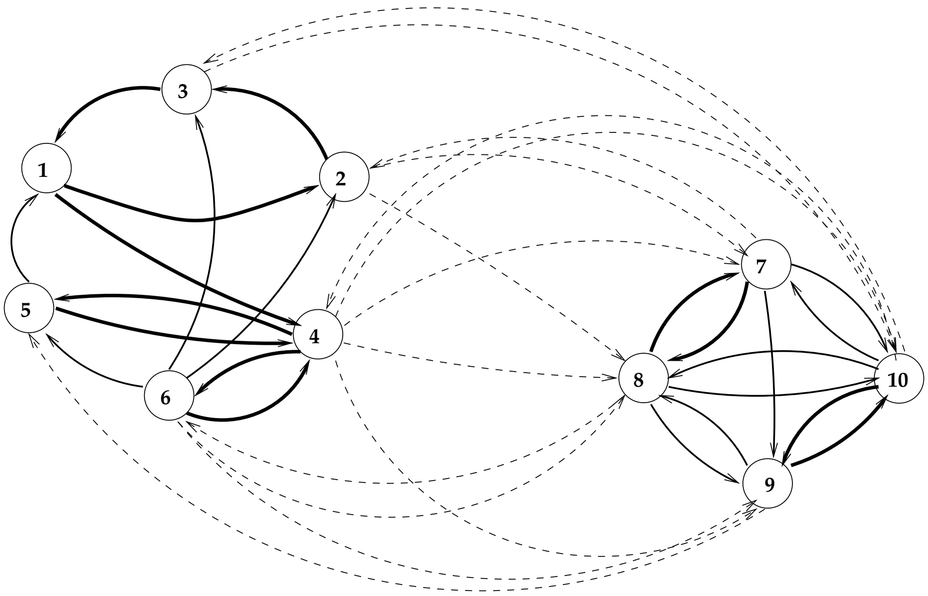

Figure 4.2: Aggregation in Stewart matrix.

Stewart matrix

The last small example is that proposed by Stewart [61]. Note the impact of different ε values on accuracy of the result.

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

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

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

4.2.2 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 [13]:

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

where neither A nor c depend on the iteration step k. In particular, BSOR procedure (see Algorithm 5) is parametrized by:

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

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

SCC We are looking for strongly connected components in the underlying graph of the Markov chain. Edges weighted with probability less than ε (prespecified decomposability parameter) are ignored. This partitioning coincides with one resulting from the near-decomposability test of the Markov Chain Analyzer (MARCA) [63].

Algorithm 5 BSOR method for solving the system $\mathbf{L}^T(R|R)\boldsymbol{\pi} = \mathbf{0}$

repeat

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

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

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

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

end for

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

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

until stop criterion succeeds

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 [13, 10]

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 4.4 for two-dimensional Markov chain model.

4.2.3 Interactive Computer System

In this section we come back to the model described in Figure 1.1 in Chapter 1 (for a detailed treatment see [60]). We recall that the model represents a time-shared multiprogrammed, paged, virtual memory computer system, modeled as a closed queueing network. In order to perform numerical experiments we assign specific values for the parameters of the model such as:

- the number of user in the system $N = 3, 10, 20, 50$,

- *Belady-Kuehner lifetime function* $q(\alpha, M, k)$ where M denotes the size of primary memory and α, k are some constants which depend on program characteristics and memory management strategy [60],
- the mean think time of a user at the terminal $\lambda^{-1} = 10s$,
- the mean service time of a SM $u_1^{-1} = 5ms$,
- the mean service time of a I/O device $u_2^{-1} = 20ms$,
- the mean compute time I/O requests $r = 20ms$,
- the mean compute time of the process $c = 500ms$.

We fix all parameters, except the number of users in the system (N), according to [60]. Recall that the state of the system is coded by a triple $\mathbf{z} = (z_1, z_2, z_3)$ of non-negative numbers, where z_1 denotes the number of users thinking or busy at theirs terminals, z_2 and z_3 denote, respectively, the number of processes in the queue of SM and I/O. Obviously $z_1 + z_2 + z_3 \leq N$. There are at most six transition which can be made from any state, i.e. from (z_1, z_2, z_3) to states:

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

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

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

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

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

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

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

Table 4.1: Numerical results for ICS example

For ICS model with 50 users we perform several experiments with iterative method. The solution computed by combinatorial aggregation is used as a starting vector for BSOR algorithm. The speed of convergence, measured in the number of iterations, is compared with BSOR starting from the uniform distribution. We investigate three block partitionings:

1. **SCC** for $\varepsilon = 0.0002$ which gives the same partition as **Aggr** into 51 strongly connected components.

2. **CC*** for $\varepsilon = 0.1$ yields 1221 blocks (connected components).

3. **CC** for $\varepsilon = 0.003$ results in 1326 blocks.

The stopping criterion we use in BSOR is $||\boldsymbol{\pi}^{(k)} - \boldsymbol{\pi}^{(k-1)}||_\infty \leq$ *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 4.1) that starting from approximate solution computed by combinatorial aggregation allows us to reduce the number of iterations about $4-5$ times. In some cases e.g. for **SCC** partition the time cost of Algorithm 2 is comparable with the cost of the first iteration of BSOR. One can additionally accelerate combinatorial aggregation by using iterative method (e.g. SOR) instead of GTH for solving subproblems inside closed classes.

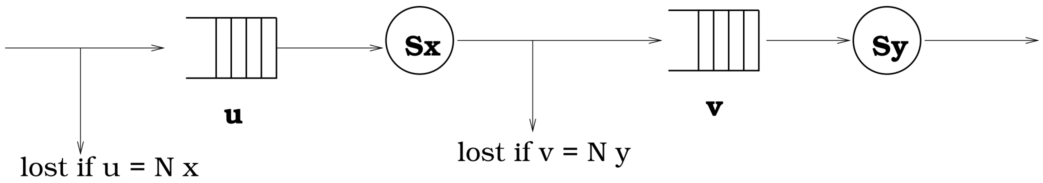

Figure 4.3: Telecommunication model for 2D example

4.2.4 A Two-Dimensional Markov Chain Model

We consider here a two dimensional Markov chain, studied e.g. in [13, 63]. It is a simple telecommunication model illustrated in Figure 4.3. There are two servers; each has a queue of waiting tasks of prespecified maximum size. A task arrives, wait for the first server, then waits for the second and finally leaves the network.

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

Table 4.2: Numerical results for 2D example.

The states are pairs (u, v) where u ranges from 0 through N_x and v ranges from 0 through N_y. States correspond to the size of two queues. In this model we assume transitions to the South, East and North-West. From any non-boundary state (u, v) there are following values assigned to the transitions:

$(u, v-1)$ (South)	with rate	v	task leaves the network
$(u+1, v)$ (East)	with rate	2025.0	task arrives to first queue
$(u-1, v+1)$ (North-West)	with rate	u	task enters second queue

The state space of the Markov chain is of size $(N_x + 1)(N_y + 1)$. In larger experiments the values of N_x and N_y are both set to 128, yielding a matrix of order 16,641 with 66,049 nonzero elements. For this model we perform several experiments:

1. the approximate solution is calculated using combinatorial aggregation algorithm for $\varepsilon = 0.06$;

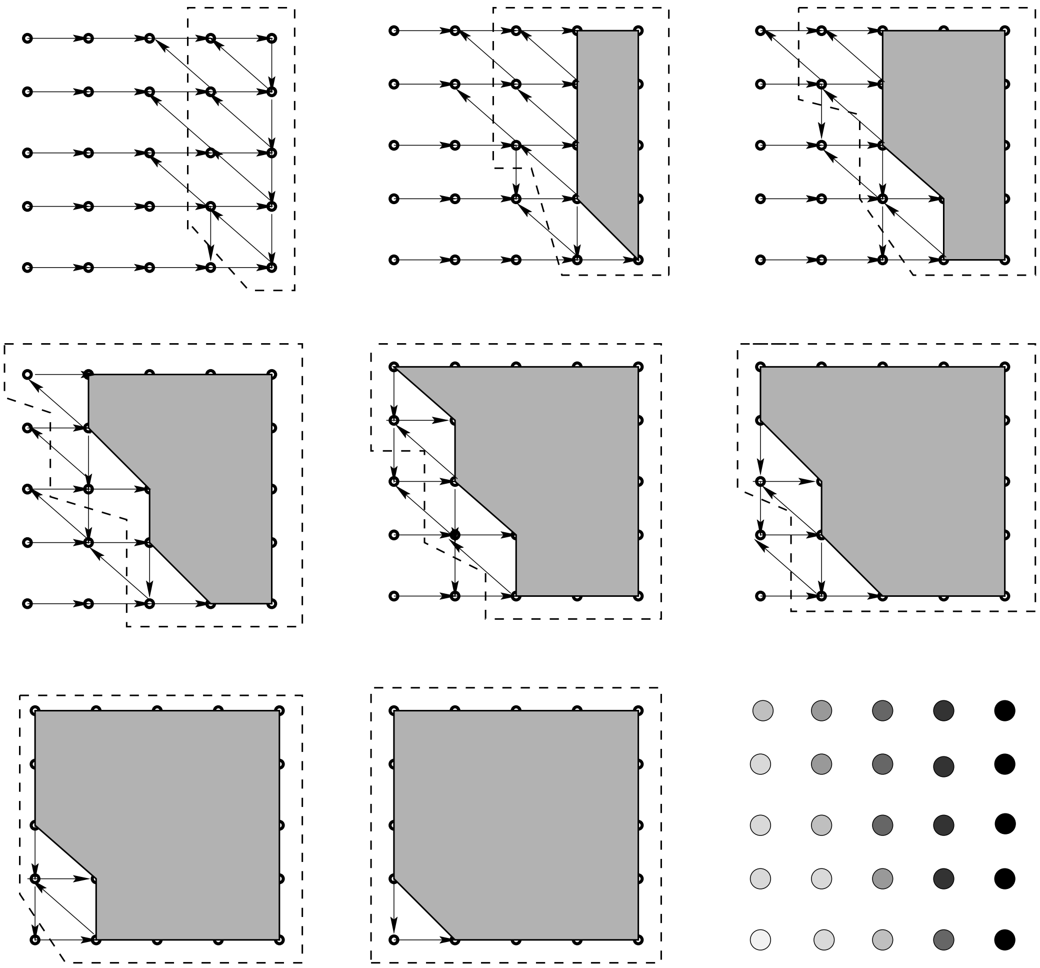

Figure 4.4: Aggregation in 2D example.

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 4.4 illustrates the process of aggregation in combinatorial aggregation algorithms for $N_x = N_y = 4$. There are 8 phases of aggregation each shown in a separate figure. We see a subgraph of shortest edges G_{min} in every phase; dashed line surrounds the only non-singleton closed class appearing in that phase. In the last figure the states are colored according to the values of their asymptotic coefficients yielding **Asymp** partition for BSOR. These correspond to consecutive aggregation phases, i.e. those states which are aggregated earlier have bigger stationary probability. Despite that during every step there is only one non-singleton closed class, i.e. the aggregation does not proceed in parallel, the time needed for the whole computation in only $\mathcal{O}(n^2)$ (cf. "ripple" aggregation from Section 4.1).

The numerical results are summarized in Table 4.2. 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**.

Chapter 5

Rapid mixing for matroid Markov chain

We investigate some problems in matroid theory related to Markov chains. We introduce the notion of exchange-preservation property and show that it implies a cutset expansion. Some subclasses of transversal matroids turn out to have exchange-preservation property. We also an alternative proof of the strong orderability for transversal matroids.

5.1 Preliminaries

5.1.1 Matroids

A matroid is an abstraction which generalizes both graphs and vector spaces. A matroid $\mathcal{M}$ consists of a ground set E of which some subsets are declared to be **independent**. The independent sets must satisfy three properties:

(1) the empty set is independent,

(2) all subsets of an independent set are also independent,

(3) if U and V are independent and $|U| > |V|$ then some element of U can be added to V to yield an independent set.

We will denote the set of independent sets of a matroid by $\mathcal{I}$, hence $\mathcal{M} = (E, \mathcal{I})$. All maximal independent sets, called **bases** of $\mathcal{M}$, have the same cardinality (called the rank of a matroid). The set of all bases will be denoted by $\mathcal{B}$. Equivalently one can also define matroid $\mathcal{M}$ by giving its set of bases; the family of independent sets is then induced as the family of all subsets of bases. For $B_1, B_2 \in \mathcal{B}$ we have following base axiom: for each $x \in B_1 \setminus B_2$ there exists $y \in B_2 \setminus B_1$ such that $(B_1 \cup \{y\}) \setminus \{x\} \in \mathcal{B}$.

For a matroid $\mathcal{M}$ the **bases-exchange graph** $\mathcal{G}_{\mathcal{M}}$ is a graph, whose nodes are all bases of $\mathcal{M}$. There exists an edge between B_1 and B_2 in $\mathcal{G}_{\mathcal{M}}$ iff $|B_1 \triangle B_2| = 2$, where the symbol $\triangle$ denotes a symmetric difference, i.e. B_2 can be obtained from B_1 (and B_1 from B_2) by a exchange operation $B_2 = B_1 \setminus \{x\} \cup \{y\}$ ($B_1 = B_2 \setminus \{y\} \cup \{x\}$). Bases-exchange graph was first introduced by Edmonds [17], for further information the reader is referred to [45, 46, 67].

We will use a family of bipartite graphs

$$\mathcal{IG}_{\mathcal{M}} = \{\mathcal{IG}(B_1, B_2)\}_{B_1, B_2 \in \mathcal{B}},$$

indexed by pairs of bases B_1, B_2 of $\mathcal{M}$, defined as follows. For any two bases B_1, B_2, sets of vertices of $\mathcal{IG}(B_1, B_2)$ are B_1 and B_2; its edges are all the pairs $(x, y) \in B_1 \times B_2$ satisfying $(B_1 \setminus \{x\}) \cup \{y\} \in \mathcal{B}$ and $(B_2 \setminus \{y\}) \cup \{x\} \in \mathcal{B}$. We call graph $\mathcal{IG}(B_1, B_2)$ an **internal exchange graph** of bases B_1, B_2.

For any connected graph $\mathcal{G}$, we define a matroid $\mathcal{M}(\mathcal{G})$ as follows: the independent sets are sets of edges of all acyclic subgraphs (forests) of G; bases are all spanning trees of G. Another example of a matroid is a vectorial matroid, whose ground-set is the set of vectors over a field and whose bases are maximum cardinality linearly independent subsets of the set of vectors. In the uniform matroid of rank k over n-element groundset E (denoted as U_n^k), independent sets are all subsets of E of cardinality $\leq k$. For a detailed treatment of matroid theory the reader is referred to [49, 67].

5.1.2 Matroid Expansion Conjecture

We start with the definition of graph expansion.

Definition 5.1.1 *For a graph $\mathcal{G} = (V, E)$ and for $S \subseteq V$, let $\Gamma(S)$ denote the cut set in G induced by S, i.e. the set of edges in E with one endpoint in S and one endpoint in $V \setminus S$. The cutset-expansion (or simply expansion) of $\mathcal{G}$ is:*

$$\mathcal{CS}(\mathcal{G}) = \min_{S \subseteq V, |S| \leq \frac{|V|}{2}} \frac{|\Gamma(S)|}{|S|}$$

To the best of our knowledge, there is no counter-example to the following conjecture proposed in [47].

Conjecture 5.1.2 *For any matroid $\mathcal{M}(E, \mathcal{I})$*

$$\mathcal{CS}(\mathcal{G}_{\mathcal{M}}) \geq 1.$$

The positive resolution of the above conjecture, even in a slightly weaker form and/or for special classes of matroids (vectorial, transversal, etc.) would be of major combinatorial interest. Furthermore, it would also be of fundamental algorithmic significance, which is argumented below.

For a given matroid $\mathcal{M}$ consider the following Markov chain being the **natural random walk** on $\mathcal{G}_{\mathcal{M}}$ $((X_t)_{t=0,1,\ldots})$. If X_t is the state (base) at time t, then with probability $\frac{1}{2}$ take $X_{t+1} = X_t$, and with probability $\frac{1}{2}$, let X_{t+1} be determined as follows: Choose e from X_t and f from E uniformly at random and if $X' = X \setminus \{e\} \cup \{f\} \in \mathcal{B}$ then $X_{t+1} = X'$, otherwise $X_{t+1} = X_t$. We see at once that the Markov chain X_t can be simulated efficiently (given an independence oracle for $\mathcal{M}$), and that it converges to the uniform distribution over $\mathcal{B}$ (it is symmetric). Moreover, the expansion of $\mathcal{G}_{\mathcal{M}}$ implies that X_t has large *conductance*, and hence possesses the *rapid mixing property*, which amounts, roughly, to X_t approaching its stationary distribution arbitrarily close for t = poly-log($|\mathcal{B}|$). Therefore, the natural random walk on $\mathcal{G}_{\mathcal{M}}$ can be used as an efficient almost uniform generation scheme for the set of bases, and in view of the fact of *self-reducibility* [36, 39] of matroids we obtain an efficient randomized algorithm to approximately count the number of bases.

The results of this chapter show that the expansion conjecture holds for the class of all matroids satisfying *exchange-preservation property*, which is properly contained in the class of *strongly base orderable* matroids. We define a matroid $\mathcal{M}$ to be **strongly base orderable** if for any two bases B_1 and B_2 there exists a bijection $\pi : B_1 \to B_2$ such that for all subsets $A \subseteq B_1$, $(B_1 \setminus A) \cup \pi(A)$ is a base of $\mathcal{M}$. It is easy to show that for any such π also $(B_2 \setminus \pi(A)) \cup A$ is a base of $\mathcal{M}$. In subsequent sections bijection π will be called **strongly orderable bijection**. In [67] it was proved that all *transversal matroids* (see Section 5.3) and all *gammoids* (for definition see [49, 67]) are strongly base orderable.

5.2 Cutset expansion

We consider the problem of bounding the cutset expansion of the bases-exchange graph of a matroid. Our investigations are restricted to the strongly base orderable matroids which satisfy the **exchange-preservation property** (Definition 5.2.1). Theorem 5.2.2 claims that cutset expansion for this class is at least $\frac{1}{2}$. In order to prove it we apply the method of bounding path congestion using the state-space path encoding [35]. In the next section we prove that the class of exchange-preserving matroids contains, in particular, all binary transversal matroids.

Throughout this section we fix a matroid $\mathcal{M} = (E, \mathcal{I})$; the set of its bases will be denoted by $\mathcal{B}$. For any bases B_1, B_2 of $\mathcal{M}$, their internal exchange graph gives raise to transformations of B_1 and B_2 into another pair of bases B_1', B_2', differing only by one element from original bases. These transformations are induced by exchanging elements connected by edges of $\mathcal{IG}(B_1, B_2)$. Formally, for every edge $(x, y) \in B_1 \times B_2$ of $\mathcal{IG}(B_1, B_2)$, we define $\Theta_{x,y}(B_1, B_2) = (B_1', B_2')$ by $B_1' = B_1 \setminus \{x\} \cup \{y\}$ and $B_2' =$

$B_2 \setminus \{y\} \cup \{x\}$. Observe that for every strongly orderable bijection $\Phi : B_1 \to B_2$, if $\Phi(x) = y$ then $\Theta_{x,y}(\Phi)$ is also a strongly orderable bijection $B_1' \to B_2'$, where $\Theta_{x,y}(\Phi)$ denotes Φ with $x \mapsto y$ replaced by $y \mapsto x$:

$$\Theta_{x,y}(\Phi) = \Phi \setminus \{(x,y)\} \cup \{(y,x)\}. \tag{5.1}$$

Roughly speaking, strongly orderable bijections are preserved by transformations $\Theta_{x,y}$. Two pairs of bases, (B_1, B_2) and (B_1', B_2') are called **compatible** ([68]) if they are over the same multi-set of elements of matroid, that is if

$$B_1 \cup B_2 = B_1' \cup B_2' \quad \text{and} \quad B_1 \cap B_2 = B_1' \cap B_2'.$$

In Theorem 5.2.2 we consider matroids, in which each set of compatible pairs of bases can be divided into disjoint subsets having a common strongly orderable bijection, up to transformation (5.1). If we consider a **bases pairs graph**, with the set of vertices consisting of ordered pairs of different bases and edges linking pairs (B_1, B_2) and $\Theta_{x,y}(B_1, B_2)$, the condition above is equivalent to the existence of family of disjoint hypercubes which covers the bases pairs graph. Each hypercube corresponds to a single strongly orderable bijection. In the sequel we will identify a strongly orderable bijection between a pair of bases with a matching in its internal exchange graph.

Definition 5.2.1 *We say that $\mathcal{IG}_{\mathcal{M}}$ has* **exchange-preservation property** *if there exists a family of strongly orderable bijections Φ_{B_1,B_2} in graphs $\mathcal{IG}(B_1, B_2)$, indexed by pairs of bases of $\mathcal{M}$, satisfying for each such pair B_1, B_2,*

$$\Theta_{x,y}(\Phi_{B_1,B_2}) = \Phi_{\Theta_{x,y}(B_1,B_2)}, \quad \textit{for every edge } (x,y) \textit{ in } \Phi_{B_1,B_2}. \tag{5.2}$$

A matroid $\mathcal{M}$ is called **exchange-preserving** if $\mathcal{IG}_{\mathcal{M}}$ has exchange-preservation property. In particular, exchange-preserving matroid is necessarily strongly base orderable. The main result of this section gives some sufficient condition on a matroid to induce satisfactory expansion of its bases-exchange graph.

Theorem 5.2.2 *If $\mathcal{IG}_{\mathcal{M}}$ has exchange-preservation property then $\mathcal{G}_{\mathcal{M}}$ has cutset expansion at least $\frac{1}{2}$.*

Proof: Following a method introduced in [35], we construct a family of canonical paths $\Gamma = \{\gamma_{X,Y}\}_{X,Y \in \mathcal{B}}$ in $\mathcal{G}_{\mathcal{M}}$, where $\mathcal{B}$ denotes the set of bases of $\mathcal{M} = (E, \mathcal{I})$. Let $\{\Phi_{X,Y}\}_{X,Y \in \mathcal{B}}$ be any fixed family of strongly orderable bijections satisfying (5.2) in Definition 5.2.1. We also fix some linear order $\leq_E$ in E. For arbitrary bases X and Y this order induces a linear ordering $\leq_{X,Y}$ of edges connected by bijection $\Phi_{X,Y}$ in the following way: $(x,y) \leq_{X,Y} (x',y')$ if $\max(x,y) \leq_E \max(x',y')$, where $x, x' \in X \setminus Y$ and $y, y' \in Y \setminus X$. Note that $\leq_{X,Y}$ is preserved by transformations $\Theta_{x,y}$, that is up to transformation (5.1), $\leq_{X,Y} = \leq_{\Theta_{x,y}(X,Y)}$, for each (x,y) connected by $\Phi_{X,Y}$.

For every pair of bases $X, Y \in \mathcal{B}$, we define a path $\gamma_{X,Y}$ in $\mathcal{G}_{\mathcal{M}}$ as a sequence $X_0, X_1, \ldots, X_m$, where $m = |X \setminus Y| = |Y \setminus X|$. Observe that each strongly orderable bijection $X \to Y$ can be equivalently defined as a function $(X \setminus Y) \to (Y \setminus X)$, since it is necessarily an identity on $X \cap Y$. A path $\gamma_{X,Y}$ is determined, together with an auxiliary sequence $Y_0, \ldots, Y_m$, by the following equations, where $(x_1, y_1) \leq_{X,Y} \ldots \leq_{X,Y} (x_m, y_m)$ denote the edges of $\Phi_{X,Y}$ ordered by $\leq_{X,Y}$:

$$\begin{array}{rcl} (X_0, Y_0) & = & (X, Y), \\ (X_i, Y_i) & = & \Theta_{x_i, y_i}(X_{i-1}, Y_{i-1}) \quad \text{for } i = 1 \ldots m. \end{array} \tag{5.3}$$

Obviously $X_m = Y, Y_m = X$, moreover $\gamma_{X,Y}$ contains precisely $m \leq |E|$ edges. To prove that Γ is a family of canonical paths for $\mathcal{G}_{\mathcal{M}}$, which implies cutset expansion at least $\frac{1}{2}$, it is sufficient (cf. [59]) to bound the path congestion. For our purposes it is sufficient to show the following PATH CONGESTION BOUND:

> for every edge (X', X'') in $\mathcal{G}_{\mathcal{M}}$, the number of paths $\gamma_{X,Y}$ containing this edge is not greater than $|\mathcal{B}|$.

For an edge (X', X'') in $\mathcal{G}_{\mathcal{M}}$, let ***path***$(X', X'')$ be the set of pairs (X, Y) such that the path $\gamma_{X,Y}$ contains (X', X''). In order to show the PATH CONGESTION BOUND, we define a 1-1 mapping σ : ***path***$(X', X'') \to \mathcal{B}$ as follows:

$$\sigma(X, Y) = X \triangle Y \triangle X' = (X \cap Y) \cup ((X \triangle Y) \setminus X'),$$

where $\triangle$ denotes a symmetric difference. It can be verified that if (X, Y) belongs to ***path***(X', X''), it must satisfy:

$$X' \subseteq X \cup Y \quad \text{and} \quad X \cap Y \subseteq X'.$$

Hence, it follows easily that $\sigma(X, Y)$ is a base. Moreover, if $X \neq X'$ (or equivalently $\sigma(X, Y) \neq Y$), then there exists $1 \leq i \leq m$ such that

$$(X', \sigma(X, Y)) = \Theta_{x_i, y_i}(\ldots \Theta_{x_1, y_1}(X, Y) \ldots). \tag{5.4}$$

For the PATH CONGESTION BOUND, it is enough to show that for every base Y', there exists at most one pair (X, Y) for which $\sigma(X, Y) = Y'$. Assuming that $\sigma(X, Y) = Y'$, we will show that X and Y are determined uniquely by Y'; to this end we will "reconstruct" X and Y from X', Y' and X''. Put $\{\bar{x}\} := X' \setminus X''$, $\{\bar{y}\} := X'' \setminus X'$. Let

$$(\bar{y}_1, \bar{x}_1) \leq_{X', Y'} (\bar{y}_2, \bar{x}_2) \leq_{X', Y'} \ldots \leq_{X', Y'} (\bar{y}_j, \bar{x}_j)$$

denote all the edges of $\Phi_{X', Y'}$ less then $(\bar{x}, \bar{y})$ in ordering $\leq_{X', Y'}$. The edge $(\bar{x}, \bar{y})$ belongs to $\Phi_{X', Y'}$ by (5.4) because it belongs to $\Phi_{X,Y}$. As (X', Y') is on the canonical path starting at X, there were made exactly j exchanges $(\bar{x}_1, \bar{y}_1) \ldots (\bar{x}_j, \bar{y}_j)$ on the way from X to X'. It means that $X' = X \setminus \{\bar{x}_1, \ldots, \bar{x}_j\} \cup \{\bar{y}_1, \ldots, \bar{y}_j\}$, which uniquely determines X by

$$X := X' \setminus \{\bar{y}_1, \ldots, \bar{y}_j\} \cup \{\bar{x}_1, \ldots, \bar{x}_j\}.$$

Similarly, (5.4) allows us to "reconstruct" Y from Y' by

$$Y := Y' \setminus \{\bar{x}_1, \ldots, \bar{x}_j\} \cup \{\bar{y}_1, \ldots, \bar{y}_j\}.$$

□

We conclude this section by noting that requirements in Theorem 5.2.2 could be slightly relaxed. For the efficient counting (and generation) of the bases of a matroid it is sufficient to have cutset expansion at least inverse polynomial with respect to the size of a matroid (cf. [59]). As usual, we assume that the size of (a representation of) a matroid is of the same order as the size of its groundset. To be able to formulate appropriate weaker assumptions, we generalize the exchange-preservation property introduced in Definition 5.2.1:

Definition 5.2.3 *We say that $\mathcal{IG}_{\mathcal{M}}$ has k-***local exchange-preservation property** *if for each pair B_1, B_2 of bases of $\mathcal{M}$, there exists a family of strongly orderable bijections Φ_{B_1,B_2} in graph $\mathcal{IG}(B_1, B_2)$ of size $|\Phi_{B_1,B_2}| \leq k$, such that for each such pair B_1, B_2,*

$$\Theta_{x,y}(\pi) \in \Phi_{\Theta_{x,y}(B_1,B_2)}, \quad \textit{for each } \pi \in \Phi_{B_1,B_2} \textit{ and each edge } (x,y) \textit{ in } \pi.$$

Observe that Definition 5.2.1 corresponds to 1-local exchange-preservation property. Note that we deliberately overload notation Φ_{B_1,B_2} to denote single strongly orderable bijections (in Definition 5.2.1) as well as sets of strongly orderable bijections (in Definition 5.2.3), as this should not lead to any confusion. Result from Theorem 5.2.2 can be easily extended to matroids satisfying k-local exchange-preservation property for any bound k.

Theorem 5.2.4 *If $\mathcal{IG}_{\mathcal{M}}$ has k-local exchange-preservation property then*

$$\mathcal{CS}(\mathcal{G}_{\mathcal{M}}) \geq \frac{1}{2k}.$$

Proof: Similar to the proof of Theorem 5.2.2, but now we have more freedom in choosing paths $\gamma_{X,Y}$. We take for $\gamma_{X,Y}$ a path determined like in (5.3) by *any* strongly orderable bijection $\pi \in \Phi_{X,Y}$. The only modification amounts to replacing injectivity of the coding function σ by the requirement that inverse image $\sigma^{-1}(B)$ of any base is of size at most k. The assumption $|\Phi_{X',Y'}| \leq k$ implies that we can "reconstruct" a pair (X, Y) in at most k ways. This gives a modified path congestion bound:

for every edge (X', X'') in $\mathcal{G}_{\mathcal{M}}$, the number of paths containing this edge is not greater than $k|\mathcal{B}|$,

from which we conclude that cutset expansion is at least $\frac{1}{2k}$. □

As an easy conclusion from this theorem we obtain:

Corollary 5.2.5 *For matroids with polynomially bounded (w.r.t. the size of a matroid) local exchange-preservation property there exists an efficient algorithm to approximate $|\mathcal{B}|$.*

5.3 Transversal matroids

In this section we show that the class of *exchange preserving* matroids contains all binary transversal matroids and strict matching matroids. We can associate a matroid with an arbitrary family

$$\mathcal{A} = (A_1, \dots, A_n)$$

of subsets of a finite set E (groundset). A subset $X = \{x_1, \dots, x_k\}$ of distinct elements of E is a **partial transversal** of $\mathcal{A}$ if there exists a subfamily $(A_{i_1}, \dots, A_{i_k})$ of $\mathcal{A}$ such that $x_j \in A_{i_j}$ $(1 \le j \le k)$. (x_j is said to **represent** A_{i_j} in X.) The maximal partial transversals are then the bases of a matroid $\mathcal{T}(E, \mathcal{A})$ on E, called a **transversal matroid** [18, 67]. The collection $\mathcal{A}$ is a **presentation** of $\mathcal{T}(E, \mathcal{A})$ (it is not necessarily unique), for a given transversal matroid.

Each presentation of a transversal matroid can be associated in a natural way with a bipartite graph representation. One set of vertices in this bipartite graph consists of the underlying set of elements of the matroid, the other set is the family of sets of elements in the matroid presentation; an 'element' vertex is joined to a 'set' vertex if the element is contained in the set. Denote this graph as $\mathcal{G}(E, \mathcal{A})$.

5.3.1 Strong orderability

Transversal matroids are strongly base orderable. We present here a proof of this fact alternative to that in [67]. We construct a strongly orderable bijection $\pi : (X \setminus Y) \to (Y \setminus X)$ for arbitrary bases X and Y of a transversal matroid $\mathcal{M}$, starting from the bipartite graph representation associated to an arbitrary presentation $\mathcal{A}$ of $\mathcal{M}$. We will apply this construction in Lemma 5.3.2, which states exchange-preservation property for binary subclass of transversal matroids. Consider two matchings in $\mathcal{G}(E, \mathcal{A})$: m_X between X and $\mathcal{A}$, and m_Y between Y and $\mathcal{A}$, representing X and Y, respectively. Denote by $X_{\neg Y}$ the subset of elements in X matched (via m_X) to some set $A \in \mathcal{A}$, which is not matched (via m_Y) to any element of Y. Symmetrically, let $Y_{\neg X}$ denote the subset of elements in Y matched (via m_Y) to some set $A \in \mathcal{A}$, which is not matched (via m_X) to any element of X. Notice that $|X_{\neg Y}| = |Y_{\neg X}|$. Assume that m_X and m_Y are chosen to satisfy:

$$X_{\neg Y} \cap Y = \emptyset \quad \text{and} \quad Y_{\neg X} \cap X = \emptyset, \tag{5.5}$$

i.e. there is no element from $X \cap Y$ in $X_{\neg Y}$ nor in $Y_{\neg X}$. If m_X and m_Y do not satisfy (5.5), then they can be modified appropriately according to the following procedure:

> Assume $e \in X_{\neg Y} \cap Y$. Modify m_Y by matching e via m_Y to the same set from $\mathcal{A}$, which is matched to e via m_X. This will result in excluding e from $X_{\neg Y}$, while not enlarging $Y_{\neg X}$, but possibly some other element $e_1 \in X$ will be added to $X_{\neg Y}$ (namely the one matched to the same set, to which e was matched previously by m_Y). Continuing in the same way with e_1, we obtain a sequence $e_0 = e, e_1, \ldots e_l$. The crucial observations is that once e_i was excluded from $X_{\neg Y}$, it can never be added back, since it is matched via both m_X and m_Y to the same set.

When the procedure terminates, we have two possibilities: either after excluding e_l no other element was added to $X_{\neg Y}$ (hence globally e was excluded from $X_{\neg Y}$ and by the way e_l was excluded from $Y_{\neg X}$), or e_{l-1} was replaced by $e_l \notin Y$ (in this case the global modification of $X_{\neg Y}$ amounts to replacing e_1 by e_l and $Y_{\neg X}$ remains unchanged). In both cases, number of elements of $X_{\neg Y} \cap Y$ was decreased by excluding e_1. Symmetrically we treat each $e \in Y_{\neg X} \cap Y$.

Observe that (5.5) implies immediately that

$$X_{\neg Y} = \emptyset \quad \text{and} \quad Y_{\neg X} = \emptyset. \tag{5.6}$$

To show this, notice that whenever some $e \in X_{\neg Y}$ then by (5.5) $e \notin Y$, hence $Y \cup \{e\}$ is a transversal, which is obviously false.

Now we are ready to define π: for each $x \in (X \setminus Y) \setminus X_{\neg Y}$, define $\pi(x) \in (Y \setminus X) \setminus Y_{\neg X}$ as the outcome of the following procedure:

> Let $A_1 \in \mathcal{A}$ be a set matched to x and let y_1 be an element of Y matched to A_1. (A_1 is matched via m_Y to some element of Y, since $x \notin X_{\neg Y}$.) If $y_1 \notin X \cap Y$, put $\pi(x) := y_1$. Otherwise, let $A_2 \in \mathcal{A}$ be a set matched to y_1 via m_X and let y_2 be an element of Y matched to A_2. Existence of such y is guaranteed by (5.5). Continuing in this manner we eventually reach $y_i \notin X \cap Y$ (since $y_1, y_2, \ldots$ are distinct elements of the finite set $X \cap Y$) and put $\pi(x) := y_i$.

To show that π is strongly orderable, take any $x \in (X \setminus Y) \setminus Y_{\neg X}$, and let $y_1, \dots, y_i = \pi(x)$ denote the sequence resulting from the procedure above. After exchanging x and $\pi(x) = y_i$, matchings m_X and m_Y (on the left) are transformed to m'_X and m'_Y (on the right):

$$
\begin{array}{ccccc}
x & — & A_1 & — & y_1 \\
y_1 & — & A_2 & — & y_2 \\
\vdots & & \vdots & & \vdots \\
y_{i-1} & — & A_i & — & y_i
\end{array}
\qquad\qquad
\begin{array}{ccccc}
y_1 & — & A_1 & — & x \\
y_2 & — & A_2 & — & y_1 \\
\vdots & & \vdots & & \vdots \\
y_i & — & A_i & — & y_{i-1}
\end{array}
$$

Obviously, m'_X and m'_Y demonstrate that sets $X \setminus \{x\} \cup \{\pi(x)\}$ and $Y \setminus \{\pi(x)\} \cup \{x\}$ are transversals. Moreover, exchanging x and $\pi(x)$ doesn't eliminate any other possible exchanges induced by π (since sets of elements from $X \cap Y$ belonging to sequences $y_1, \dots, y_i$ for different elements $x \in X$ are disjoint) which implies that π is strongly orderable. $\square$

When transversals X and Y are total, each pair of matchings m_X and m_Y satisfies (5.6), hence induces some strongly orderable bijection. On the other hand, not each such bijection is induced (in all presentations) by some pair of matchings, even if X and Y are total. The counterexample is shown in Figure 5.1.

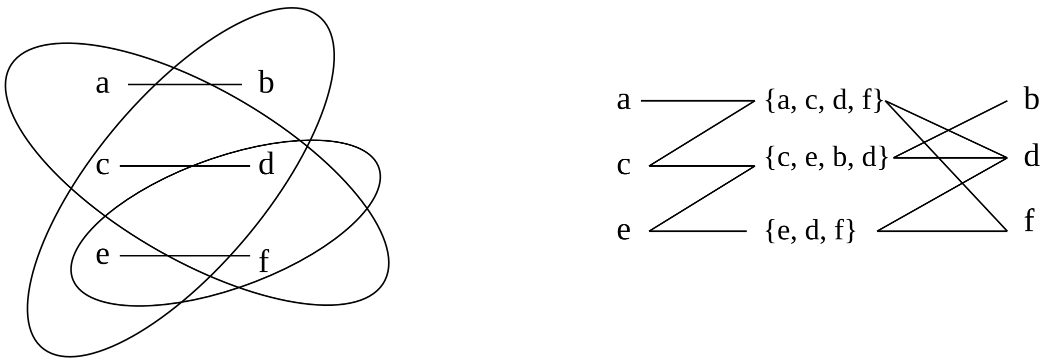

Figure 5.1: The strongly orderable bijection $a \mapsto b$, $c \mapsto d$, $e \mapsto f$ between bases $\{a, c, e\}$ and $\{b, d, f\}$ is not induced by any pair of matchings.

5.3.2 Exchange-preservation property

We will show that transversal binary matroids satisfy exchange-preservation property. The following fact was proved by Edmonds (cf. [4]).

Fact 5.3.1 *A transversal matroid is binary if and only if it has a presentation whose associated bipartite graph is acyclic.*

Lemma 5.3.2 *Transversal binary matroids satisfy exchange-preservation property.*

Proof: The family of strongly orderable bijections $\Phi_{X,Y}$ is defined as follows: for each pair of bases (X, Y) we take the family of bijections induced by matchings satisfying (5.6) in the bipartite graph $\mathcal{G}(E, \mathcal{A})$. We can assume that this graph is acyclic, according to Fact 5.3.1. Assume that for some bases X, Y there are two different strong orderable bijections in $\Phi_{X,Y}$ ($x_i \to y_i$ and $x_i \to y_{\sigma(i)}$), both induced by a pair of matchings in $\mathcal{G}(E, \mathcal{A})$; $\looparrowright$ represents the alternating path in $\mathcal{G}(E, \mathcal{A})$.

$$\begin{array}{ccccc} x_1 & \text{—} \looparrowright \text{—} & y_1 & \qquad\qquad x_1 & \text{—} \looparrowright \text{—} & y_{\sigma(1)} \\ x_2 & \text{—} \looparrowright \text{—} & y_2 & \qquad\qquad x_2 & \text{—} \looparrowright \text{—} & y_{\sigma(2)} \\ \vdots & \vdots & \vdots & \qquad\qquad \vdots & \vdots & \vdots \\ x_k & \text{—} \looparrowright \text{—} & y_k & \qquad\qquad x_k & \text{—} \looparrowright \text{—} & y_{\sigma(k)} \end{array} \tag{5.7}$$

We argue, that this situation implies the existence of a cycle in $\mathcal{G}(E, \mathcal{A})$. Consider the following sequence of vertices:

$$x_1 \looparrowright y_1 = y_{\sigma(\sigma^{-1}(1))} \looparrowright x_{\sigma^{-1}(1)} \looparrowright y_{\sigma^{-1}(1)} \ldots$$

where $\looparrowright$ represents a path induced by one of pairs of matchings in (5.7). After a finite number of $\looparrowright$ steps we finally repeat one of the elements:

$$x_1, y_1, x_{\sigma^{-1}(1)}, y_{\sigma^{-1}(1)}, \ldots$$

twice. This gives a cycle in $\mathcal{G}(E, \mathcal{A})$, which contradicts Fact 5.3.1. $\square$

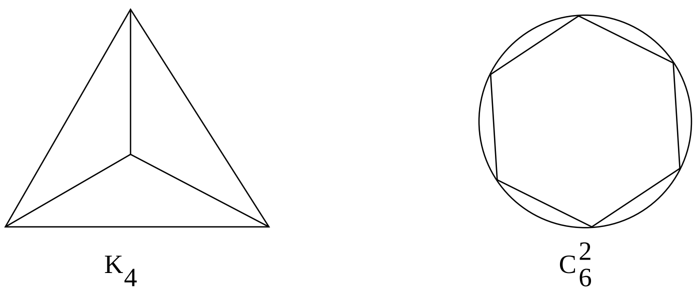

Figure 5.2: Forbidden subgraphs for binary transversals.

It was proved [67] that transversal binary matroids are graphic. Moreover, we have the following characterization:

Theorem 5.3.3 ([4]) *Let G be a finite graph. Then the cycle matroid of G is transversal if and only if G contains no subgraph homeomorfic to K_4 or C_k^2 $(k > 2)$.*

5.3.3 Strict matching matroids

In this section we investigate another subclass of transversal matroids which satisfies exchange-preservation property. **A strict matching matroid** is induced by matchings in an arbitrary graph. A standard result of Edmonds and Fulkerson [67] states the following:

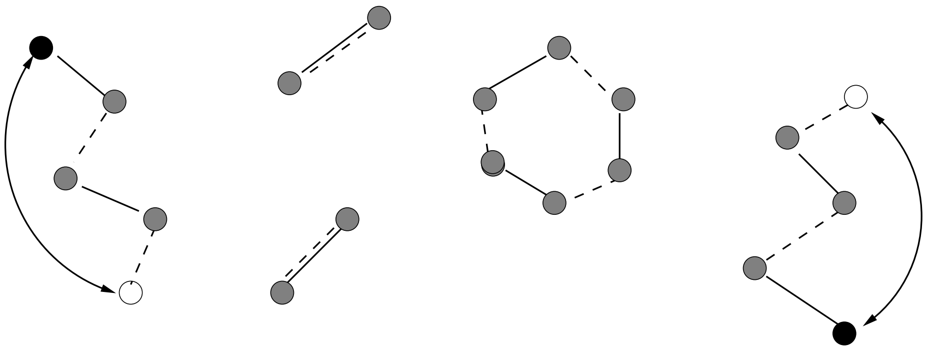

Figure 5.3: Strong orderable bijection in a strict matching matroid

Fact 5.3.4 *Let $G = (V, E)$ be a graph and let* $\mathbf{match}(G)$ *be the collection of subsets of vertices defined by* $X \in \mathbf{match}(G)$ *if* X *is the sets of endpoints of edges of some matching of* G*. Then* $\mathbf{match}(G)$ *is the collection of independent sets of a matroid on* V*.*

Fix a graph G and the matroid $\mathbf{match}(G)$. Consider any two bases of this matroid. Each of them is induced by a matching in G. This matching may not be unique, however it is maximal as it defines a base. The union of these two matching is a subgraph of G which consist only of cycles and paths (cf. Fig 5.3: one matching is marked by dashed lines, another by straight lines). Cycles are always of even length, paths are single edges or have also even length. This property follows from the maximality of both matchings. The vertices one the cycles, inside the paths and endpoints of single edges meet both matchings hence belong to the intersection of bases. While constructing a strong base orderable bijection we must take into account only the endpoints of paths. The endpoints of a path belong to different bases (black or white), hence we can match them to obtain a bijection. Observe that the matching in internal exchange graph defined in such a way is preserved by exchanges (cf. Definition 5.2.1) as we can switch each pair of endpoints of a given path independently from all others.

Fact 5.3.5 *If* $\mathcal{M}$ *is a strict matching matroid then* $\mathcal{IG}_{\mathcal{M}}$ *has exchange-preservation property.*

5.4 Summary of Chapter 5

The inclusions between different classes of matroids considered in this chapter are presented in Figure 5.4. We have introduced the class of *exchange preserving* matroids and shown that they have a cutset expansion $\frac{1}{2}$. This immediately implies the existence of effective sampling and randomized approximation schemes for bases of matroids satisfying exchange-preservation property. We also proved that the class

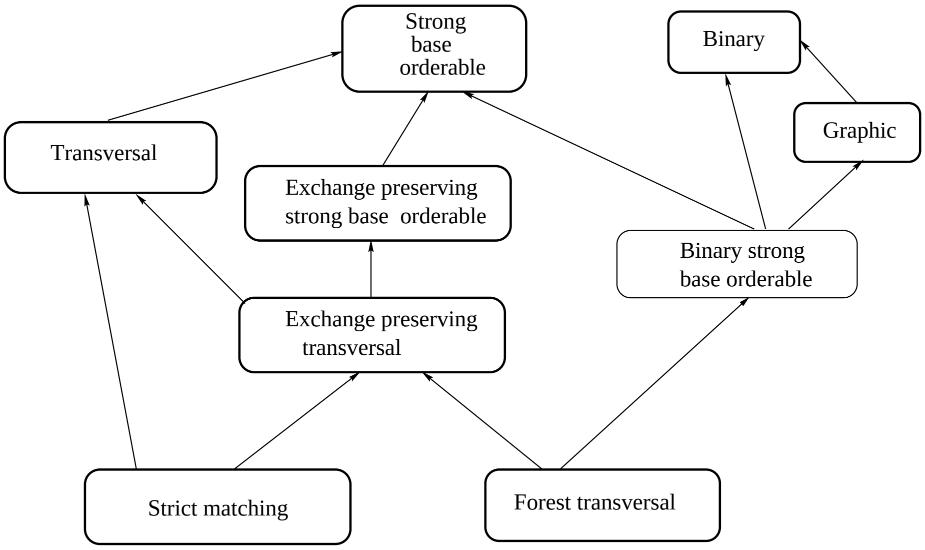

Figure 5.4: Classes of matroids, arrows denote inclusions.

under consideration contains all binary transversal matroids and strict matching matroids.

The challenging open question and a natural continuation of this work is to try to enlarge this class to all transversal or strong base orderable matroids.

We have also presented some preliminary results yielding the expansion for all strong base orderable matroids having polynomially bounded families of strongly orderable bijections preserved by single exchanges.

Further study is needed to find a characterization of the class of matroids satisfying polynomially bounded local exchange-preservation property, for which the expansion follows from Theorem 5.2.4.

The comparison of transversal matroids satisfying exchange-preservation property and strongly base orderable matroids satisfying this property with the class of balanced matroids, for which inverse polynomially bounded expansion was proved in [19], remains an interesting open question.

Chapter 6

Markov models of communication networks

We define three Markov chains corresponding to three models of packet flow. We start by describing the local behavior of one switch, then we define the global-state Markov chains. Similar approach is presented in the case of relaxed blocking model. Last part of this chapter is devoted to proving ergodicity of the Markov chains. In the next chapter we relate different models for full blocking as well as for relaxed blocking.

6.1 Modeling dynamic behavior of MIN

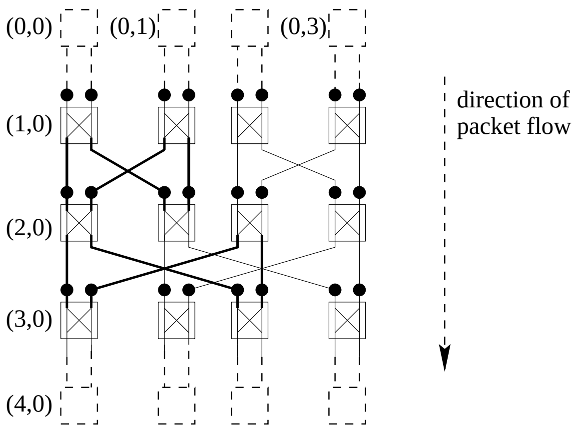

Figure 6.1: Butterfly network with three stages ($H = 3$); buffers are denoted by black circles. The border stages, depicted with dashed line, are introduced in Section 6.1 to represent an interaction of the network with the environment, i. e. input and flow-out of packets.

Multistage Interconnection Networks are used to route packets appearing in the switches at the first stage (input switches or input links) to appropriate output switches (addresses) at the last stage (see Fig. 6.1). There is exactly one path from every input switch to every output switch, and each packet visits all the stages along its way through the network. We assume ([43, 65]), that an address of a packet is random (it is usually done to make a performance analysis possible) — this means that a packet after entering into each switch chooses its direction randomly. The computation of the network proceeds in synchronized steps more or less simultaneously by all the switches. The degree of parallelism varies in different models.

One step consists of the flow of packets between every pair of neighbor stages. Locally, in one switch, such a step consists of two half-steps, (how they are related in time depends on a chosen model of packet flow):

1) *Flow-out phase*: some packets leave the switch (after resolving conflicts), but only if there is a place in the corresponding buffer at the next stage.

2) *Input phase*: packets from the previous stage enter to the empty buffers of the switch.

The *ripple model* is the most frequently used in the literature [48, 65, 69]. It has the property that the flow-out phase at the stage happens after the flow-out phase at the next stage has been completed. We can effectively simulate such a model (switches need not to wait) by means of a closely related *odd-even model* (which can be thought of as a pipelining implementation of the ripple model): the flow-out phases occur simultaneously at the odd stages (while even stages are idle), followed by input phases at the even stages, then in every consecutive step the odd and even stages change their roles. Another natural possibility is offered by the *simultaneous model*, in which the flow-out phase occurs simultaneously at all the stages (similarly input phase which follows flow-out one). In this model switches are never idle.

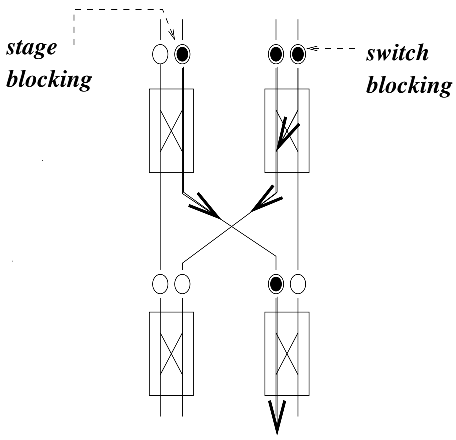

Figure 6.2: Switch blocking and stage blocking.

Full blocking versus relaxed blocking. We distinguish between two cases of message blocking each other (cf. Fig. 6.2). **Switch blocking** occurs when both messages in the input buffers of a switch have to be routed to the same output link. In this case only one of these two messages can be routed in a current time step, the other has to stay in its buffer. The choice (conflict resolution) is made randomly during the flow-out phase. **Stage blocking** occurs if a buffer at the consecutive stage is occupied, disallowing another packet from the previous stage to be routed. This introduces bottom-up dependencies between the switches of the network. To simplify the analysis, so called **relaxed blocking model** (called here **relaxed model** in short) was proposed [56], where stage blocking is ignored: a message is deleted, if the receiving buffer is occupied.

In the previous paragraph we described three different models of packet flow, being variations of the full blocking model. In the sequel we also consider similar variations of relaxed model. Throughout the rest of the chapter let H denote the number of stages of the butterfly network under consideration. The structure of the network can be represented by the graph $G = (I, E)$, where the sets of nodes and edges are defined as follows:

$$I = \{1, 2, \dots, H\} \times \{0, 1, \dots, W\}, \quad W = 2^{H-1} - 1,$$

$$E = \{((r, c), (r+1, c')) : 1 \le r \le H-1, \quad 0 \le c \le W, \quad c' = c \text{ or } c' = c^{(r)}\}$$

where $c^{(r)}$ is the unique number in $[0, W]$ that differs from c only in the rth bit, $0 \le c \le W$, $1 \le r \le H-1$. Nodes (r, c) and $(r, c^{(r)})$ have common neighbors $(r+1, c)$ and $(r+1, c^{(r)})$ at the $(r+1)$st stage.

Definition 6.1.1 *For finite sets A,B, a* **c-map** $P : A \rightsquigarrow B$ *is a function* $P : A \to [0,1]^B$ *such that for every* $a \in A$, $P(a)$ *is a probability distribution, i.e.* $\sum_{b \in B} P(a)(b) = 1$. *Usually sets A and B are called* **domain** *and* **co-domain** *of P. We say that P is* **deterministic** *if for each a, there exists some b with* $P(a)(b) = 1$.

Note A concept of c-map coincides with the notion of the conditional probability distribution; a c-map $\{*\} \rightsquigarrow A$ from a singleton set can be seen as an unconditional probability distribution on A. When $A = B$, $P : A \rightsquigarrow A$ corresponds to the transition probability matrix of a finite Markov chain.

6.1.1 Local description of one switch

A single switching element has two input ports (each equipped with one-element buffers) and two output ones, with each input port connected to each output one (see Fig. 6.1). Let the set $B = \{\circ, \downarrow, \searrow\}$ represent possible states of one buffer in the switch. Element "$\circ$" denotes an empty buffer, and the two other elements of B represent the two possible directions of a packet residing in the buffer ("$\downarrow$" denotes the straight direction, "$\searrow$" denotes the crossing direction, see Fig. 6.1). To improve readability, we introduce a redundant notation: "$\swarrow$"="$\searrow$". Let $S = B \times B$ denote the set of all possible states of both buffers of a switch.

As a single switching element cannot be described by a Markov chain $S \rightsquigarrow S$ (because its behavior depends on states of its neighbors), we encode the behavior of it by means of two c-maps, describing first and second half of the step, respectively:

$$\sigma : S \times \{0,1\}^2 \rightsquigarrow S \times \{0,1\}^2, \qquad \tau : S \times \{0,1\}^2 \rightsquigarrow S.$$

We define these c-maps in Tables 6.1 and 6.2.

initial S	IN:$(1,1)$		IN:$(0,1)$		IN:$(1,0)$		IN:$(0,0)$	
	new S	out	new S	out	new S	out	new S	out
$\circ\ \circ$	$\circ\ \circ$	$(0,0)$	$\circ\ \circ$	$(0,0)$	$\circ\ \circ$	$(0,0)$	$\circ\ \circ$	$(0,0)$
$\downarrow\ \circ$	$\downarrow\ \circ$	$(0,0)$	$\circ\ \circ$	$(1,0)$	$\downarrow\ \circ$	$(0,0)$	$\circ\ \circ$	$(1,0)$
$\searrow\circ$	$\searrow\circ$	$(0,0)$	$\searrow\circ$	$(0,0)$	$\circ\ \circ$	$(0,1)$	$\circ\ \circ$	$(0,1)$
$\circ\ \downarrow$	$\circ\ \downarrow$	$(0,0)$	$\circ\ \downarrow$	$(0,0)$	$\circ\ \circ$	$(0,1)$	$\circ\ \circ$	$(0,1)$
$\circ\swarrow$	$\circ\swarrow$	$(0,0)$	$\circ\ \circ$	$(1,0)$	$\circ\swarrow$	$(0,0)$	$\circ\ \circ$	$(1,0)$
$\downarrow\swarrow$	$\downarrow\swarrow$	$(0,0)$	$\frac{1}{2}\ \downarrow\ \circ$ $\frac{1}{2}\ \circ\swarrow$	$(1,0)$	$\downarrow\swarrow$	$(0,0)$	$\frac{1}{2}\ \downarrow\ \circ$ $\frac{1}{2}\ \circ\swarrow$	$(1,0)$
$\searrow\downarrow$	$\searrow\downarrow$	$(0,0)$	$\searrow\downarrow$	$(0,0)$	$\frac{1}{2}\ \circ\ \downarrow$ $\frac{1}{2}\ \searrow\circ$	$(0,1)$	$\frac{1}{2}\ \circ\ \downarrow$ $\frac{1}{2}\ \searrow\circ$	$(0,1)$
$\searrow\!\!\!\!\swarrow$	$\searrow\!\!\!\!\swarrow$	$(0,0)$	$\searrow\circ$	$(1,0)$	$\circ\swarrow$	$(0,1)$	$\circ\ \circ$	$(1,1)$
$\downarrow\ \downarrow$	$\downarrow\ \downarrow$	$(0,0)$	$\circ\ \downarrow$	$(1,0)$	$\downarrow\ \circ$	$(0,1)$	$\circ\ \circ$	$(1,1)$

Table 6.1: Definition of σ: the first half of the step (flow out of packets from a switch). For each state (in the left-most column) and content of buffers of consecutive neighbour nodes (in the top-most row) there is a new state (together with a probability) and output of packets. Note that output of packets turns out to be determined (i.e. with probability 1).

The domain of σ consists of pairs $(s,(b_1,b_2))$, where s denotes the state of the switch before a step and (b_1,b_2) describes the contents of the two buffers of the neighbor switches at the next stage: $b_1 = 0$ ($b_2 = 0$) means that the buffer is empty, otherwise the buffer is occupied by a packet. The co-domain of σ consists of pairs $(s,(o_1,o_2))$, where s denotes the state of the switch after the flow out of packets and a pair (o_1,o_2) describes how many packets flow out. Similarly, $o_1 = 1$ ($o_2 = 1$) means that a packet flows out. Hence $\sigma(s,(b_1,b_2))$ "returns" an output of packets from a switch together with a new state of that switch, given an initial state s and (b_1,b_2) describing the contents of the two buffers of neighbors at the next stage.

Similarly, the domain of τ consists of pairs $(s,(i_1,i_2))$, where s is a state before an input of new packets (i.e. after σ), and a pair (i_1,i_2) describes how many packets

initial S	in:$(0,0)$ new S	in:$(1,0)$ new S	in:$(0,1)$ new S	in:$(1,1)$ new S
$\circ\ \circ$	$\circ\ \circ$	$\frac{1}{2}\downarrow\circ$, $\frac{1}{2}\searrow\circ$	$\frac{1}{2}\circ\downarrow$, $\frac{1}{2}\circ\swarrow$	$\frac{1}{4}\downarrow\swarrow$, $\frac{1}{4}\searrow\downarrow$, $\frac{1}{4}\searrow\!\!\!\!\swarrow$, $\frac{1}{4}\downarrow\downarrow$
$\downarrow\circ$	$\downarrow\circ$		$\frac{1}{2}\downarrow\swarrow$, $\frac{1}{2}\downarrow\downarrow$	
$\searrow\circ$	$\searrow\circ$		$\frac{1}{2}\searrow\downarrow$, $\frac{1}{2}\searrow\!\!\!\!\swarrow$	
$\circ\downarrow$	$\circ\downarrow$	$\frac{1}{2}\searrow\downarrow$, $\frac{1}{2}\downarrow\downarrow$		
$\circ\swarrow$	$\circ\swarrow$	$\frac{1}{2}\downarrow\swarrow$, $\frac{1}{2}\searrow\!\!\!\!\swarrow$		
$\downarrow\swarrow$	$\downarrow\swarrow$			
$\searrow\downarrow$	$\searrow\downarrow$			
$\searrow\!\!\!\!\swarrow$	$\searrow\!\!\!\!\swarrow$			
$\downarrow\downarrow$	$\downarrow\downarrow$			

Table 6.2: Definition of τ: the second half of the step: (input of packets). Empty fields could be here arbitrarily filled, as they correspond to situations which can never occur, although we will need to fill them in Section 6.1.3 to express relaxation.

enter the switch. $\tau(s,(i_1,i_2))$ describes a new state after packets have entered the switch in state s. Our description of one switch is in the spirit of the probabilistic automaton from [56].

For convenience let us introduce sets $I_r = \{(r,c) : 0 \le c \le W\}$, for $0 \le r \le H+1$. The set I_r represents all nodes at the rth stage. Stages 0 and $H+1$ are auxiliary and are introduced only for technical convenience: the stage 0 intuitively represents input of packets into the network and symmetrically stage $(H+1)$ receives packets which flow out from the network.

In the next section we will define Markov chains $V \rightsquigarrow V$, describing one step of the computation of the network, where $V = S^{I \cup I_{H+1}}$ is the set of states, $I = \bigcup_{1 \le r \le H} I_r$. Elements of V represent all configurations (states) of the network, i.e. all possible states of all the switches. We will also need a set V' of states in a half-step, defined as $V' = (S \times \{0,1\}^2)^{I_0 \cup I \cup I_{H+1}}$. Elements of V' describe the contents of all switches together with the packets which flowed out in the first half-step (and are placed on the links between stages).

For further definitions we need two families of auxiliary functions:

$$IN^{r,c} : V \to \{0,1\}^2, \quad \text{for} \quad (r,c) \in I \cup I_0,$$

$$in^{r,c} : V' \to \{0,1\}^2, \quad \text{for} \quad (r,c) \in I \cup I_{H+1}.$$

$IN^{r,c}(v)$ returns information about the contents of buffers of neighbors at the next stage for a switch at node (r,c) (the pair (b_1,b_2) above); similarly $in^{r,c}(v')$ returns information about packets coming into the switch ((i_1,i_2) above). In the definition of these functions we are "looking through" the edges of graph G describing the structure of the network (cf. Fig 6.1):

$$IN^{r,c}(v) = \begin{cases} (occ(s\downarrow_1), occ(\bar{s}\downarrow_1)) & \text{if } c < c^{(r)}, \\ (occ(\bar{s}\downarrow_2), occ(s\downarrow_2)) & \text{otherwise,} \end{cases} \qquad \text{for } 0 < r < H;$$

where $s = v(r+1,c) \in B^2$, $\bar{s} = v(r+1, c^{(r)}) \in B^2$ and $(a,b)\downarrow_1 = a$, $(a,b)\downarrow_2 = b$ denote the usual projections. The auxiliary function $occ : B \to \{0,1\}$ tells us whether the buffer is occupied; $occ(\circ) = 0,\ occ(\downarrow) = occ(\searrow) = 1$.

$$in^{r,c}(v') = \begin{cases} (o\downarrow_1, \bar{o}\downarrow_1) & \text{if } c < c^{((r-1))}, \\ (\bar{o}\downarrow_2, o\downarrow_2) & \text{otherwise,} \end{cases} \qquad \text{for } 1 < r < H+1;$$

where $o = v'(r-1,c)\downarrow_2 \in \{0,1\}^2$, $\bar{o} = v'(r-1, c^{((r-1))})\downarrow_2 \in \{0,1\}^2$. The definitions of $IN^{0,c}$,$in^{1,c}$, $IN^{H,c}$ and $in^{H+1,c}$ are slightly different, as they describe the "border" of the network:

$$IN^{r,c}(v) = (occ(v(r+1,c)\downarrow_1), occ(v(r+1,c)\downarrow_2)), \quad \text{for } r = 0, H;$$

$$in^{r+1,c}(v') = v'(r,c)\downarrow_2 \in \{0,1\}^2, \quad \text{for } r = 0, H.$$

We introduce four families of c-maps $\sigma^{r,c}$, $\tau^{r,c}$, $\pi^{r,c}$, $\omega^{r,c}$, indexed by $(r,c) \in I$, which will be the atomic elements of the full Markov chains in the next section. The c-maps

$$\sigma^{r,c} : V \rightsquigarrow S \times \{0,1\}^2, \qquad \tau^{r,c} : V' \rightsquigarrow S$$

apply σ,τ to the switch at node (r,c):

$$\sigma^{r,c}(v)(s,o) = \sigma(v(r,c), IN^{r,c}(v))(s,o), \qquad \tau^{r,c}(v')(s) = \tau(v'(r,c)\downarrow_1, in^{r,c}(v'))(s)$$

On the other hand $\pi^{r,c} : V \rightsquigarrow S \times \{0,1\}^2$, $\omega^{r,c} : V' \rightsquigarrow S$ describe an idle phase of a switch at node (r,c) and are defined by:

$$\pi^{r,c}(v)(v(r,c),(0,0)) = 1, \qquad \omega^{r,c}(v')(v'(r,c)\downarrow_1) = 1.$$

Note It is surprising that the domains of $\sigma^{r,c}, \tau^{r,c}, \pi^{r,c}$ and $\omega^{r,c}$ are so "large", but observe that these c-maps examine only a little part of their domains. For example $\sigma^{r,c}$ needs only its state (i.e. the state of the switch at (r,c)) and the state of some neighbors, and is "independent"[1] from the rest of the network. Such a blowup of domains is unavoidable in our setting, as a parallel composition of c-maps as defined in Section 6.1.2 works only for c-maps with equal domains.

To unify our notation we will also need some "border" c-maps describing input and output of packets into/from the network:

$$\sigma^{0,c}, \pi^{0,c}, \sigma^{H+1,c}, \pi^{H+1,c} : V \rightsquigarrow S \times \{0,1\}^2,$$

[1] The formalization of the notion of independence is given in Section 7.1.

$$\tau^{H+1,c}, \omega^{H+1,c} : V' \rightsquigarrow S$$

defined as follows:

$$\sigma^{0,c}(v)(s,o) = \sigma((\downarrow,\downarrow), IN^{0,c}(v))(s,o),$$

$$\pi^{0,c}(v)((\downarrow,\downarrow),(0,0)) = 1,$$

$$\sigma^{H+1,c}(v)((\circ,\circ),(0,0)) = 1,$$

$$\pi^{H+1,c}(v)(v(H+1,c),(0,0)) = 1.$$

The c-maps $\tau^{H+1,c}$, $\omega^{H+1,c}$ are defined analogously to the corresponding c-maps for other stages:

$$\tau^{H+1,c}(v')(s) = \tau(v'(H+1,c)\downarrow_1, in^{H+1,c}(v'))(s),$$

$$\omega^{H+1,c}(v')(v'(H+1,c)\downarrow_1) = 1.$$

Remark In the above definitions of $\sigma^{r,c}, \pi^{r,c}, \tau^{r,c}, \omega^{r,c}$ we intentionally omitted some detail. The co-domains of $\sigma^{r,c}, \pi^{r,c}$ should be actually $(S \times \{0,1\}^2)^{\{(r,c)\}}$, i.e. a set of functions from a singleton set $\{(r,c)\}$; moreover, co-domains of $\tau^{r,c}, \omega^{r,c}$ should be $S^{\{(r,c)\}}$. In the next sections we assume that it actually is the case, as such co-domains are appropriate while composing these c-maps into more complex ones.

6.1.2 Global behavior of the network

In this section we define three Markov chains Γ_{ripple}, Γ_{sim}, $\Gamma_{\text{odd-even}} : V \rightsquigarrow V$, which correspond to the described models of packet flow. The Markov chains will be composed of c-maps $\sigma^{r,c}$, $\tau^{r,c}$, $\pi^{r,c}$, $\omega^{r,c}$ as defined in the previous section by means of the following two operations on c-maps:

Definition 6.1.2 *For $P : A \rightsquigarrow B$ and $Q : B \rightsquigarrow C$, we define the* **sequential composition of** P **and** Q, *denoted by $P \bullet Q$, as a c-map $A \rightsquigarrow C$ determined by the following formula: $(P \bullet Q)(a,c) = \sum_{b \in B} P(a,b) \cdot Q(b,c)$.*
For $P : A \rightsquigarrow B$ and $Q : A \rightsquigarrow C$, the **parallel composition of** P **and** Q, *$P \parallel Q : A \rightsquigarrow B \times C$, is defined by $(P \parallel Q)(a)(b,c) = P(a)(b) \cdot Q(a)(c)$.*

Essentially "$\bullet$" corresponds to matrix multiplication, while "$\|$" gives a family of product distributions. Both operations are associative and parallel composition is also commutative. While defining them we were inspired by the operators of *Communicating Sequential Processes* (CSP) [32] – for example "$\|$" enforces component c-maps to "proceed" simultaneously like the operator of synchronized parallel composition in CSP. In the following "$\|$" has higher priority than "$\bullet$". We sometimes write $\binom{P}{Q}$ instead of $(P \bullet Q)$ for the sake of simplicity (this allows us to better recognize c-maps concerned with certain stages of the network). Throughout the rest of the chapter we deliberately ignore the difference between sets $A^{B\cup C}$ and $A^B \times A^C$; moreover we identify the sets A and $(A \times \{*\})$, where $\{*\}$ is an arbitrary singleton set. Due to such an identification "up to obvious bijection" we avoid some inessential technicalities.

Both operations of composition have identities; for any set A, we define c-maps $Id^{\bullet}_A : A \rightsquigarrow A$ and $Id^{\|}_A : A \rightsquigarrow \{*\}$ as follows:

$$Id^{\bullet}_A(a)(a) = 1, \quad Id^{\|}_A(a)(*) = 1, \quad \text{for any} \quad a \in A.$$

Proposition 6.1.3 *The families of c-maps $Id^{\bullet}_A$ and $Id^{\|}_A$, are identities for $\bullet$ and $\|$, respectively, i.e. for any c-map $P : B \rightsquigarrow C$:*

$$Id^{\bullet}_B \bullet P = P = P \bullet Id^{\bullet}_C,$$

$$Id^{\|}_B \parallel P = P.$$

The domain V can be split into subparts corresponding to the stages as follows:

$$V = V_1 \times V_2 \times \cdots \times V_{H+1},$$

where $V_r = S^{I_r}$. Analogously $V' = V'_0 \times V'_1 \times \cdots \times V'_{H+1}$, where $V'_r = (S \times \{0,1\}^2)^{I_r}$. Before defining the Markov chains, let us group together the c-maps $\sigma^{r,c}$, $\pi^{r,c}$ belonging to the rth stage into $\sigma^r, \pi^r : V \rightsquigarrow V'_r$:

$$\sigma^r = \|_{0\le c\le W}\, \sigma^{r,c}, \quad \pi^r = \|_{0\le c\le W}\, \pi^{r,c}, \quad \text{for } r = 0, 1, \ldots, H+1.$$

Analogously we define τ^r,ω^r : $V' \rightsquigarrow V_r$ as a parallel composition of appropriate components:

$$\tau^r = \|_{0\le c\le W}\, \tau^{r,c}, \quad \omega^r = \|_{0\le c\le W}\, \omega^{r,c}, \quad \text{for } r = 1, 2, \ldots H+1.$$

We define the aforementioned three Markov chains as follows (bold letters correspond to non-idle actions):

$$\begin{aligned}\Gamma_{\text{sim}} = \quad & (\boldsymbol{\sigma^0} \| \, \boldsymbol{\sigma^1} \| \, \boldsymbol{\sigma^2} \| \cdots \quad \| \, \boldsymbol{\sigma^{H-1}} \| \, \boldsymbol{\sigma^H} \| \, \boldsymbol{\sigma^{H+1}}) \bullet \\ & \quad (\boldsymbol{\tau^1} \; \| \, \boldsymbol{\tau^2} \| \cdots \quad \| \, \boldsymbol{\tau^{H-1}} \| \, \boldsymbol{\tau^H} \| \, \boldsymbol{\tau^{H+1}}).\end{aligned}$$

$$
\begin{aligned}
\Gamma_{\text{ripple}} = \ & (\pi^0 \parallel \pi^1 \parallel \pi^2 \parallel \cdots \parallel \pi^{H-1} \parallel \pi^H \parallel \boldsymbol{\sigma^{H+1}}) \bullet \\
& (\omega^1 \parallel \omega^2 \parallel \cdots \parallel \omega^{H-1} \parallel \omega^H \parallel \omega^{H+1}) \bullet \\
& (\pi^0 \parallel \pi^1 \parallel \pi^2 \parallel \cdots \parallel \pi^{H-1} \parallel \boldsymbol{\sigma^H} \parallel \pi^{H+1}) \bullet \\
& (\omega^1 \parallel \omega^2 \parallel \cdots \parallel \omega^{H-1} \parallel \omega^H \parallel \boldsymbol{\tau^{H+1}}) \bullet \\
& (\pi^0 \parallel \pi^1 \parallel \pi^2 \parallel \cdots \parallel \boldsymbol{\sigma^{H-1}} \parallel \pi^H \parallel \pi^{H+1}) \bullet \\
& (\omega^1 \parallel \omega^2 \parallel \cdots \parallel \omega^{H-1} \parallel \boldsymbol{\tau^H} \parallel \omega^{H+1}) \bullet \\
& \cdots \\
& (\boldsymbol{\sigma^0} \parallel \pi^1 \parallel \pi^2 \parallel \cdots \parallel \pi^{H-1} \parallel \pi^H \parallel \pi^{H+1}) \bullet \\
& (\boldsymbol{\tau^1} \parallel \omega^2 \parallel \cdots \parallel \omega^{H-1} \parallel \omega^H \parallel \omega^{H+1}).
\end{aligned}
$$

$$
\begin{aligned}
\Gamma_{\text{odd-even}} = \ & (\boldsymbol{\sigma^0} \parallel \pi^1 \parallel \boldsymbol{\sigma^2} \parallel \pi^3 \parallel \cdots \parallel \boldsymbol{\sigma^{H-2}} \parallel \pi^{H-1} \parallel \boldsymbol{\sigma^H} \parallel \pi^{H+1}) \bullet \\
& (\boldsymbol{\tau^1} \parallel \omega^2 \parallel \boldsymbol{\tau^3} \parallel \cdots \parallel \omega^{H-2} \parallel \boldsymbol{\tau^{H-1}} \parallel \omega^H \parallel \boldsymbol{\tau^{H+1}}) \bullet \\
& (\pi^0 \parallel \boldsymbol{\sigma^1} \parallel \pi^2 \parallel \boldsymbol{\sigma^3} \parallel \cdots \parallel \pi^{H-2} \parallel \boldsymbol{\sigma^{H-1}} \parallel \pi^H \parallel \boldsymbol{\sigma^{H+1}}) \bullet \\
& (\omega^1 \parallel \boldsymbol{\tau^2} \parallel \omega^3 \parallel \cdots \parallel \boldsymbol{\tau^{H-2}} \parallel \omega^{H-1} \parallel \boldsymbol{\tau^H} \parallel \omega^{H+1}).
\end{aligned}
$$

Note that border stages proceed in accordance with other stages in each model. Let $\Gamma_1 : V \rightsquigarrow V$ denote the c-map consisting of two first lines of the definition of $\Gamma_{\text{odd-even}}$, $\Gamma_2^1 : V \rightsquigarrow V'$ denote the third line, $\Gamma_2^2 : V' \rightsquigarrow V$ denote the fourth, and let $\Gamma_2 = \Gamma_2^1 \bullet \Gamma_2^2$.

For the definition of $\Gamma_{\text{odd-even}}$ we assumed H to be even. When H is odd one should slightly modify the definition above as follows:

$$
\begin{aligned}
\Gamma_{\text{odd-even}} = & (\pi^0 \parallel \boldsymbol{\sigma^1} \parallel \pi^2 \parallel \boldsymbol{\sigma^3} \parallel \cdots \parallel \boldsymbol{\sigma^{H-2}} \parallel \pi^{H-1} \parallel \boldsymbol{\sigma^H} \parallel \pi^{H+1}) \bullet \\
& (\omega^1 \parallel \boldsymbol{\tau^2} \parallel \omega^3 \parallel \cdots \parallel \omega^{H-2} \parallel \boldsymbol{\tau^{H-1}} \parallel \omega^H \parallel \boldsymbol{\tau^{H+1}}) \bullet \\
& (\boldsymbol{\sigma^0} \parallel \pi^1 \parallel \boldsymbol{\sigma^2} \parallel \pi^3 \parallel \cdots \parallel \pi^{H-2} \parallel \boldsymbol{\sigma^{H-1}} \parallel \pi^H \parallel \boldsymbol{\sigma^{H+1}}) \bullet \\
& (\boldsymbol{\tau^1} \parallel \omega^2 \parallel \boldsymbol{\tau^3} \parallel \cdots \parallel \boldsymbol{\tau^{H-2}} \parallel \omega^{H-1} \parallel \boldsymbol{\tau^H} \parallel \omega^{H+1}).
\end{aligned}
$$

6.1.3 Relaxed blocking

In the recent work by R. Diekmann et al. [56], in order to analyze the throughput of MINs, so called relaxed blocking model was proposed, allowing packets to disappear when stage blocking occurs (for detailed description of relaxation see Section 6.1). We formally describe this phenomenon by using slight modification of already defined Markov chains Γ_{ripple}, $\Gamma_{\text{odd-even}}$ and Γ_{sim}. We start with definition of "atomic components"

$$\sigma_* : S \times \rightsquigarrow S \times \{0,1\}^2, \quad \tau_* : S \times \{0,1\}^2 \rightsquigarrow S$$

from which we build the description of the whole network. C-map σ_* is obtained from σ by applying the latter always to input $(0,0)$:

$$\sigma_*(s) = \sigma(s,(0,0)), \quad \text{for any} \quad s \in S.$$

This means that σ_* "ignores" the information about switches in the next stage, and always sends as many packets as possible.

The c-map τ_* remains almost the same as τ from Section 6.1.1; we only fill empty fields of Table 6.2 (corresponding there to impossible inputs for τ) to express the relaxation of packets entering into the switch. The full definition of τ_* can be found in Table 6.3.

Now, having already defined σ_* and τ_* we can define new families of c-maps $\sigma_*^{r,c}$, $\tau_*^{r,c}$ for $(r,c) \in I_0 \cup I \cup I_{H_1}$, as it was done in Section 6.1.1, with the only difference that functions $IN^{r,c}$ need no more to be used. We similarly group switches in the same stage to obtain:

$$\sigma_*^r = \|_{0 \le c \le W}\ \sigma_*^{r,c}, \qquad \tau_*^r = \|_{0 \le c \le W}\ \tau_*^{r,c}.$$

Three Markov chains for the relaxed model, Γ^*_{ripple}, $\Gamma^*_{\text{odd-even}}$ and Γ^*_{sim} are defined precisely as corresponding Markov chains in Section 6.1.2, with σ^r replaced by σ^r_* and τ^r replaced by τ^r_*, for any $r = 0, 1, \ldots, (H+1)$. Note that the same c-maps for idle phases are used here.

	in:(0, 0)	in:(1, 0)	in:(0, 1)	in:(1, 1)
initial S	new S	new S	new S	new S
○ ○	○ ○	$\frac{1}{2}$ ↓ ○, $\frac{1}{2}$ ↘○	$\frac{1}{2}$ ○ ↓, $\frac{1}{2}$ ○↙	$\frac{1}{4}$ ↓↙, $\frac{1}{4}$ ↘↓ $\frac{1}{4}$ ⤩, $\frac{1}{4}$ ↓ ↓
↓ ○	↓ ○	↓ ○	$\frac{1}{2}$ ↓↙, $\frac{1}{2}$ ↓ ↓	$\frac{1}{2}$ ↓↙, $\frac{\mathbf{1}}{\mathbf{2}}$ ↓ ↓
↘○	↘○	↘○	$\frac{1}{2}$ ↘↓, $\frac{1}{2}$ ⤩	$\frac{1}{2}$ ↘↓, $\frac{\mathbf{1}}{\mathbf{2}}$ ⤩
○ ↓	○ ↓	$\frac{1}{2}$ ↘↓, $\frac{1}{2}$ ↓ ↓	○ ↓	$\frac{1}{2}$ ↘↓, $\frac{\mathbf{1}}{\mathbf{2}}$ ↓ ↓
○↙	○↙	$\frac{1}{2}$ ↓↙, $\frac{1}{2}$ ⤩	○↙	$\frac{1}{2}$ ↓↙, $\frac{\mathbf{1}}{\mathbf{2}}$ ⤩
↓↙	↓↙	↓↙	↓↙	↓↙
↘↓	↘↓	↘↓	↘↓	↘↓
⤩	⤩	⤩	⤩	⤩
↓ ↓	↓ ↓	↓ ↓	↓ ↓	↓ ↓

Table 6.3: Definition of τ_*.

6.2 Ergodicity of Markov chains

In this section we prove that Γ_{ripple}, $\Gamma_{\text{odd-even}}$ and Γ_{sim} have stationary distributions. Moreover the Markov chain for Γ_{sim} is periodic with period 2.

Observe that a probability distribution p on U can be seen as a c-map from a singleton set $p : \{*\} \rightsquigarrow U$. p is **stationary distribution** for $\mathcal{M}$ if $p \bullet \mathcal{M} = p$.

In the sequel we need one more definition:

Definition 6.2.1 *A* **normal state** *of a finite Markov chain $\mathcal{M}$ is any state $u \in U$ reachable from all other states in the underlying graph of $\mathcal{M}$.*

The following facts are classical results in Markov chain theory (cf. [21, 33, 40]).

Theorem 6.2.2 *Let $\mathcal{M}$ be a finite Markov chain and $\mathcal{G}$ its underlying graph.*

(1) If u is a normal state then $\mathcal{G}$ has exactly one absorbing strong component, which consists of all states that are reachable from u.

(2) Let $\mathcal{G}'$ be a strong component of $\mathcal{G}$. Then all states in $\mathcal{G}'$ have the same period (or are aperiodic).

(3) Finite Markov chain $\mathcal{M}$ has a unique stationary distribution if $\mathcal{G}$ has a unique absorbing strong component.

Turning to our Markov chains Γ_{sim}, $\Gamma_{\text{odd-even}}$ and Γ_{ripple} (also the relaxed versions), we define three states, v_{full}, v_{even}, $v_{\text{odd}} \in V$, which will be proved to be normal states. In the state v_{full} all the switches are filled with two packets, each of them heading in the straight direction. v_{even} is identical to v_{full} on **even** stages, i.e. on stage H, stage $(H-2)$, and so on, and has empty switches on the other stages. State v_{odd} is similar to v_{even}, but now **odd** stages are full, while even ones are empty. The formal definitions:

$$v_{\text{full}}(i) = (\downarrow, \downarrow), \quad \text{for} \quad i \in I \cup I_{H+1}$$

$$v_{\text{even}}((r,c)) = \begin{cases} (\downarrow,\downarrow) & \text{if } (H-r) \text{ is even,} \\ (\circ,\circ) & \text{otherwise,} \end{cases} \qquad \text{for } (r,c) \in I \cup I_{H+1};$$

$$v_{\text{odd}}((r,c)) = \begin{cases} (\downarrow,\downarrow) & \text{if } (H-r) \text{ is odd,} \\ (\circ,\circ) & \text{otherwise,} \end{cases} \qquad \text{for } (r,c) \in I \cup I_{H+1}.$$

Proposition 6.2.3 *v_{full} is an aperiodic normal state of Γ_{ripple}.*

Proof: We give a scenario for reaching, from any state, the state v_{full} (with a non-zero probability). First we force all packets in the network to choose only the straight direction in future steps (i.e. the crossing direction is not allowed from now on). In this way we avoid new collisions inside switches. There may still be some old collisions, but after few steps they disappear, if we give priority to packets with crossing direction (a collision is always between a straight directed packet and a crossing directed one). Now the packets are arranged in independent (non-colliding) paths, represented by the straight direction, and every packet follows its path (there is no blocking). Let us call any such a state **collision-free**.

If we continue entering packets into the network, which processes them continuously, we see that after H steps we obtain the state v_{full}. This state is aperiodic as with a non-zero probability we stay in the same state. □

Proposition 6.2.4 *v_{even} is an aperiodic normal state of $\Gamma_{\mathrm{odd-even}}$.*

Proof: As in the proof of Proposition 6.2.3, we give a scenario for reaching v_{even}. In exactly the same way we obtain a collision-free state. From now on all the paths proceed independently. We call a path **even** if it is filled by packets precisely as in the state v_{even}. We must argue that in H steps every path eventually becomes even. If a packet resides at an even stage and all packets below it as well, then all these packets will move down 2 stages in the next odd-even step. If a packet resides at an odd stage and all packets below it reside at even stages, then after one step all these packets will have reached even stages. By induction, one sees that from bottom to top, after at most H steps, all packets reach even stages and act simultaneously (see Fig. 6.3). The argument for aperiodicity is the same as in Proposition 6.2.3. □

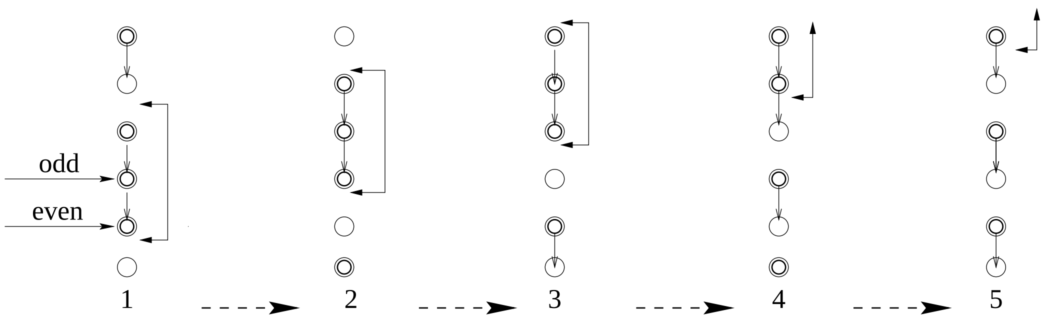

Figure 6.3: Initially blocked packets start moving continuously in one non-colliding path. The figure shows two steps of $\Gamma_{\mathrm{odd-even}}$ (i.e. four half-steps).

Proposition 6.2.5 *Both v_{even}, v_{odd} are normal states of Γ_{sim}. They are periodic with period 2.*

Proof: We proceed as in the previous propositions to obtain a state consisting of independent paths (a collision-free state). We call a path **odd** (**even**) if it is filled by packets precisely as in the state v_{odd} (v_{even}). Similarly Proposition 6.2.4, we can reach a state in which all the paths are either odd or even. Let us consider one non-colliding path and the packet which have all packets below residing at even (odd) stages. If our packet resides also at an even (odd) stage then in the next simultaneous step all these packets move down and reach odd (even) stages. If the packet resides at an odd (even) stage – that means it is blocked by the packet below (via stage blocking) then after one simultaneous step it is not blocked anymore and in the next step it will move simultaneously with packets below it. By induction, one sees that after at most H step every path is either odd or even. The situation is a bit more complicated than in Proposition 6.2.4, as we have to deal with paths of different parities. All the paths can be divided into two disjoint groups such that each two paths from the same group have at every moment the same parity.

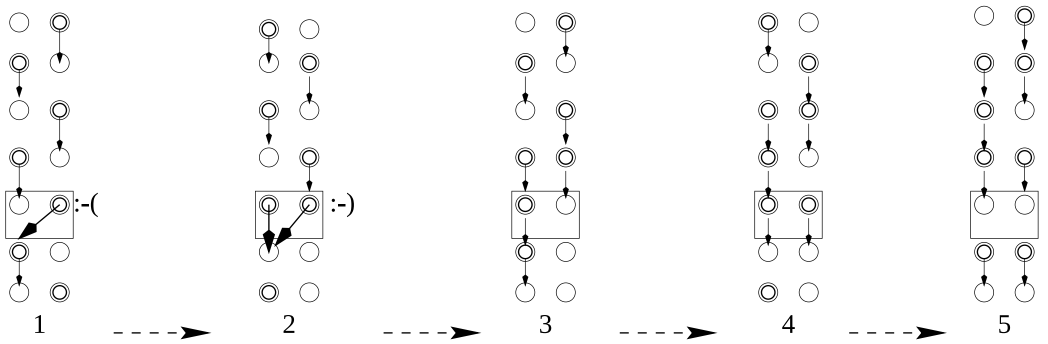

Figure 6.4: Changing the parity of one path. The packet depicted as :-(is blocked in the first step and it wins the collision in a switch during the second step :-).The left path preserves its parity, while the parity of the right one changes.

Now we want to force all paths to have the same parity (as in the states v_{even} and v_{odd}). We can change the parity of one path, assuming the existence of another path with different parity that shares one of its switches with our path. Figure 6.4 shows two paths which have a common switch. We want to change the parity of the right one. We force a packet from the right path to choose the crossing direction, which causes stage blocking. In the next step we have a collision in this switch. We give priority to the packet with the crossing direction, and after one more step obtain the situation marked in Fig. 6.4 as number 4. It is easy to observe that the right path changed its group, while the left one remained in the same group. There are still some packets on both paths which are clustered together (three packets on the left path and two on the right one), so precisely speaking the paths are not even nor odd at the moment, but as was argued in previous proof, clusters of packets will move up and finally disappear at the first stage.

Obviously not every pair of paths has a common switch, but we deal with this in the following way. Let us consider a graph with the set of all paths as its set of vertices and the set of switches as edges. There is an edge between two paths when they cross the same switch. It is easy to see that this graph is a connected hypercube. We conclude that if not all paths have the same parity, then there are two paths of different parity which cross the same switch. We can change the parity of one of these two paths, increasing or decreasing the number of even paths. Iterating this we can obtain any of the two states v_{even} and v_{odd}.

Now we prove the periodicity of these two states. We consider two sets of states of Γ_{sim}. The first one contains all states in which all even stages are stage blocked (it means there is no packet at these stages which can move). We call this set “after Γ_1”, as the described situation appears after performing Γ_1. Analogously we have the set of states “after Γ_2”. It contains all states with odd stages blocked. These sets are disjoint, because there exists always at least one switch which is not blocked. For example in state v_{full} $(H+1)$st stage is not blocked, dually in empty state the stage 0 is not blocked. In the underlying graph of Γ_{sim} these sets form a bipartite subgraph by Lemmas 7.2.2 and 7.2.3 (which are proved in Section 7.2). One of the components of this bipartite graph contains v_{odd}, the other contains v_{even}, which implies that these states are periodic with period 2.

□

As a consequence of Theorem 6.2.2 and Propositions 6.2.3, 6.2.4 and 6.2.5 we obtain:

Corollary 6.2.6 *The Markov chains Γ_{ripple}, $\Gamma_{\text{odd}-\text{even}}$ and Γ_{sim} have unique stationary distributions.*

Now we look at the Markov chains for the relaxed model.

Proposition 6.2.7 *v_{full} is an aperiodic normal state of Γ^*_{ripple} (and also of Γ^*_{sim}, which follows by Theorem 7.4.1 from Section 7.4).*

Proposition 6.2.8 *v_{even} is an aperiodic normal state of $\Gamma^*_{\text{odd}-\text{even}}$.*

Proof: The proof proceeds as for Proposition 6.2.4. The only difference is that after obtaining collision-free state, after only one step stage blocking disappears (due to relaxation of packets). Now if we continue entering packets into the network we see that after at most H steps we reach v_{even} state. □

The following corollary is a conclusion from Theorem 6.2.2 and Proposition 6.2.7 and 6.2.8.

Corollary 6.2.9 *The Markov chains Γ^*_{ripple}, Γ^*_{sim} and $\Gamma^*_{\text{odd}-\text{even}}$ have unique stationary distributions.*

Chapter 7

Comparisons between models

7.1 Algebraic properties of decomposition

We develop here mathematical tools consisting of some notation, definitions and technical facts, which will enable to prove the results linking different models.

Throughout this section, let $P : A \rightsquigarrow B_1$, $Q, Q' : B_1 \times B_2 \rightsquigarrow C$ be arbitrary c-maps, for fixed sets A, B_1, B_2. Omitted proofs are easy and follow from definitions.

Sometimes we need two c-maps to be equal only on some part of their domain, for example when they are sequentially composed after another c-map.

Definition 7.1.1
Q **is equal to** Q' **after** P, *denoted by* $Q = Q'(\lhd P)$ *if* $\forall a, (b_1, b_2) : P(a, b_1) > 0$ *implies* $Q(b_1, b_2) = Q'(b_1, b_2)$.

The equality after a composed c-map $(P_1 \parallel P_2)$ is weaker than equality after each of the components P_i $(i = 1, 2)$:

Lemma 7.1.2 *Let* $P_i : A \rightsquigarrow B_i$ *for* $i = 1, 2$ *and* $R, R' : B_1 \times B_2 \times B_3 \rightsquigarrow C$*. Then*

$$R = R'(\lhd P_1) \quad \textit{implies} \quad R = R'(\lhd(P_1 \parallel P_2)).$$

In Section 6.1, when we defined the Markov chains for different models of packet flow, we used several atomic c-maps σ^r, τ^r, sharing the property that their domain was "larger" than their co-domain. Their domains seemed to be larger than necessary. For example, the c-map σ^r "operates" only on the rth and $(r+1)$st stage, and is "independent" from what happens on other stages. Such an independence from a part of a domain is formalized as follows:

Definition 7.1.3 Q **is independent of** B_2 *(denoted by* $Q \dashv B_2$*) if*

$$\forall b_1 \in B_1, \forall b_2, b_2' \in B_2 \;\;:\;\; Q((b_1, b_2)) = Q((b_1, b_2')).$$

When $Q \dashv B_2$, we could define the c-map Q on the restricted domain B_1 without any loss of information. The following lemma is similar to Lemma 7.1.2, although the implication goes in the opposite direction:

Lemma 7.1.4 *Let $R : A \times B_1 \times B_2 \rightsquigarrow C$. Then*

$$R \dashv (B_1 \times B_2) \quad \textit{implies} \quad R \dashv B_1.$$

To make our notation more usable, we define a generalized sequential composition of c-maps $P \bullet Q$ (denoted also by $\bullet$), which allows the co-domain of P to be "smaller" than the domain of Q, when Q is independent from the rest of the domain.

Definition 7.1.5 *For $Q \dashv B_2$, the* **generalized sequential composition**

$$(P \bullet Q) : A \rightsquigarrow C$$

is defined as follows: $(P \bullet Q)(a, c) = \sum_{b_1 \in B_1} P(a)(b_1) \cdot Q((b_1, b_2))(c)$, where $b_2 \in B_2$ is arbitrary.

Lemma 7.1.6 below compares both types of sequential composition. It enables to remove some sub-components of a c-map.

Lemma 7.1.6 *For each $R : A \rightsquigarrow B_2$, $Q \dashv B_2$ implies $P \bullet Q = (P \parallel R) \bullet Q$.*

The following lemma reduces the equality of c-maps to the equality of their small sub-components after a common predecessor.

Lemma 7.1.7 *For any $R : B_1 \times B_2 \rightsquigarrow D$, $Q = Q'(\lhd P)$ implies*

$$\begin{pmatrix} P \\ Q \parallel R \end{pmatrix} = \begin{pmatrix} P \\ Q' \parallel R \end{pmatrix}.$$

For further use let us slightly generalize the notion of the independence.

Definition 7.1.8 *We say that* Q **is independent of** P *(denoted $Q \dashv P$) if*

$$\forall a \in A, \forall b_1, b_1', b_2 \in B_2 : \quad P(a, b_1), P(a, b_1') > 0 \Longrightarrow Q((b_1, b_2)) = Q((b_1', b_2))$$

Independence from a domain is stronger than independence from a c-map.

Lemma 7.1.9 *$Q \dashv B_1$ implies $Q \dashv P$.*

The following proposition essentially allows us to decompose a c-map into a parallel composition of independent components. It will be very useful for proofs of some results in the following sections.

Proposition 7.1.10 *Let $P_i : A \rightsquigarrow B_i$, $Q_i : B_1 \times B_2 \rightsquigarrow C_i$, for $i = 1, 2$, such that $Q_1 \dashv P_2$, $Q_2 \dashv P_1$. Then*

$$\begin{pmatrix} P_1 \parallel P_2 \\ Q_1 \parallel Q_2 \end{pmatrix} = \begin{pmatrix} P_1 \parallel P_2 \\ Q_1 \end{pmatrix} \parallel \begin{pmatrix} P_1 \parallel P_2 \\ Q_2 \end{pmatrix}.$$

Proof: Let $\mathcal{L}, \mathcal{R}$ denote the left and the right hand side of the equation, respectively. By definitions it follows (let B denotes $\{(b_1, b_2) | P_1(a, b_1) > 0 \text{ and } P_2(a, b_2) > 0\}$):

$$\mathcal{R}(a, (c_1, c_2)) = \left(\sum_{(b_1, b'_2) \in B} P_1(a, b_1) \cdot P_2(a, b'_2) \cdot Q_1((b_1, b'_2), c_1) \right) \cdot$$

$$\cdot \left(\sum_{(b'_1, b_2) \in B} P_1(a, b'_1) \cdot P_2(a, b_2) \cdot Q_2((b'_1, b_2), c_2) \right) =$$

$$= \sum_{(b_1, b'_2, b'_1, b_2) \in B \times B} P_1(a, b_1) \cdot P_2(a, b_2) \cdot Q_1((b_1, b'_2), c_1) \cdot Q_2((b'_1, b_2), c_2) \cdot P_1(a, b'_1) \cdot P_2(a, b'_2).$$

Now by the independences $Q_1 \dashv P_2$ and $Q_2 \dashv P_1$ we have

$$Q_1((b_1, b'_2), c_1) = Q_1((b_1, b_2), c_1), \quad \text{for any } \; b'_2 \text{ such that } P_2(a, b'_2) > 0,$$

and analogously

$$Q_2((b'_1, b_2), c_2) = Q_2((b_1, b_2), c_2), \quad \text{for any } \; b'_1 \text{ such that } P_1(a, b'_1) > 0,$$

thus, we obtain:

$$\mathcal{R}(a, (c_1, c_2)) = \sum_{(b_1, b_2) \in B} [P_1(a, b_1) \cdot P_2(a, b_2)] \cdot [Q_1((b_1, b_2), c_1) \cdot Q_2((b_1, b_2), c_2)] \cdot \quad (7.1)$$

$$\cdot \sum_{(b'_1, b'_2) \in B} P_1(a, b'_1) \cdot P_2(a, b'_2).$$

Now, as the following holds:

$$\sum_{(b'_1, b'_2) \in B} P_1(a, b'_1) \cdot P_2(a, b'_2) = \sum_{b \in B} (P_1 \parallel P_2)(a, b) = 1,$$

we conclude that the sum in (7.1) is equal to $\mathcal{L}(a, (c_1, c_2))$, which completes the proof. $\square$

When we replace the assumption on independence from a c-map in Proposition 7.1.10 by the stronger notion of independence from a domain, the c-map $(P_1 \parallel P_2)$ can be replaced by any other, needing not to be a parallel composition:

Proposition 7.1.11 *Let $R : A \rightsquigarrow B_1 \times B_2$, $Q_i : B_1 \times B_2 \rightsquigarrow C_i$, for $i = 1, 2$, such that $Q_1 \dashv B_2$, $Q_2 \dashv B_1$. Then*

$$\binom{R}{Q_1 \parallel Q_2} = \binom{R}{Q_1} \parallel \binom{R}{Q_2}.$$

Proof: The proof is analogous to the proof of Proposition 7.1.10. The only difference is that the set $\{b_1 | P_1(a, b_1) > 0\}$ (resp. $\{b_2 | P_2(a, b_2) > 0\}$) should be replaced by the whole co-domain B_1 (resp. B_2) of P_1 (resp. P_2). $\square$

The following lemma describes some nice preservation properties of parallel composition:

Lemma 7.1.12 *Let $Q_i, Q_i' : B_1 \times B_2 \rightsquigarrow C_i$, for $i = 1, 2$ Then*

(i) $Q_1 = Q_1'(\lhd P)$ *and* $Q_2 = Q_2'(\lhd P)$ *implies* $(Q_1 \parallel Q_2) = (Q_1' \parallel Q_2')(\lhd P)$;

(ii) $Q_1 \dashv B_1$ *and* $Q_2 \dashv B_1$ *implies* $(Q_1 \parallel Q_2) \dashv B_1$;

(iii) $Q_1 \dashv P$ *and* $Q_2 \dashv P$ *implies* $(Q_1 \parallel Q_2) \dashv P$.

Lemma 7.1.12 allows us to analyze (parallelly) composed c-map by looking only at smaller components.

7.2 Odd-even versus simultaneous

We start with some observations, which are put in the following lemma as they are used in other sections as well:

Lemma 7.2.1 *For $1 \le r \le H + 1$ and $0 \le c \le W$ we have the following:*

(1) $\tau^{r,c} \dashv (S \times \{0,1\}^2)^{D'_{r,c}}, \quad \omega^{r,c} \dashv (S \times \{0,1\}^2)^{(I \cup I_0 \cup I_{H+1}) \setminus \{(r,c)\}},$
where $D'_{r,c} = (I \cup I_0 \cup I_{H+1}) \setminus \{(r,c), (r-1,c), (r-1, c^{((r-1))})\}$.

(2) $\tau^{r} \dashv (S \times \{0,1\}^2)^{(I \cup I_0 \cup I_{H+1}) \setminus (I_r \cup I_{r-1})}, \quad \omega^{r} \dashv (S \times \{0,1\}^2)^{(I \cup I_0 \cup I_{H+1}) \setminus I_r}.$

(3) $\tau^{r,c} \dashv \sigma^{r-1,c}, \quad \tau^{r,c} \dashv \sigma^{r-1,c^{((r-1))}}.$

(4) $\tau^{r} \dashv \sigma^{q}, \quad \omega^{r} \dashv \sigma^{q},$ *where* $q \neq r$.

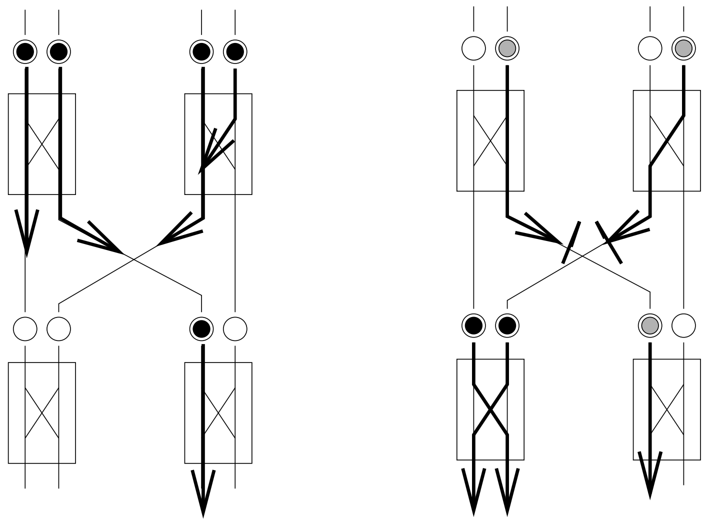

Figure 7.1: One even group of switches before and after Γ_1; on the right side the black packet moved during Γ_1, while the grey ones did not move.

(5) $\tau^r \dashv \pi^q, \quad \omega^r \dashv \pi^q,$ *for any* q.

Moreover we have:

(1') $\sigma^{r,c} \dashv S^{D_{r,c}}$, *where* $D_{r,c} = (I \cup I_{H+1}) \setminus \{(r,c),(r+1,c),(r+1,c^{(r)})\}$,

$\pi^{r,c} \dashv S^{(I \cup I_{H+1}) \setminus \{(r,c)\}}$.

(2') $\sigma^r \dashv S^{(I \cup I_{H+1}) \setminus (I_r \cup I_{r+1})}$ $(r \leq H)$, $\quad \sigma^{H+1} \dashv S^I$,

$\pi^r \dashv S^{(I \cup I_{H+1}) \setminus I_r}$.

(5') $\sigma^r \dashv \omega^q, \quad \pi^r \dashv \omega^q,$ *for any* q.

Proof: (of Lemma 7.2.1) Points (1) and (3) follow from the definitions. Intuitive justification for (3) is as follows: output of packets given to $\tau^{r,c}$ by $\sigma^{r-1,c}$ (or $\sigma^{r-1,c^{(r-1)}}$) is deterministic (in any initial state). (2) is implied by (1) and Lemma 7.1.12. (4) is a consequence of (2) and Lemma 7.1.9, while (5) holds as π^q is deterministic, for any q. Arguments for (1'), (2') and (5') are similar to (1), (2) and (5), respectively. □

Recall the meaning of the components Γ_1 and Γ_2 of $\Gamma_{\text{odd-even}}$ introduced in Section 6.1. Analogously let Γ^1_{sim} and Γ^2_{sim} denote components of Γ_{sim}, i.e. $\Gamma_{\text{sim}} = \Gamma^1_{\text{sim}} \bullet \Gamma^2_{\text{sim}}$. In the following lemma we show that after Γ_1, the full power of a simultaneous step has the same effect as Γ_2.

Lemma 7.2.2 $\Gamma_1 \bullet \Gamma_{\text{sim}} = \Gamma_1 \bullet \Gamma_2$.

Proof: We start with some intuitive arguments, the formal proof follows. All the switches can be divided into groups of four (nodes (r, c), $(r, c^{(r)})$, $(r+1, c)$, $(r+1, c^{(r)})$). We call a group *even* or *odd*, depending on the parity of $H - r$. Each switch belongs to precisely two groups (Fig. 7.1 shows an even group). During Γ_1 even groups are active, while odd ones stay idle. During Γ_2 the roles change: only odd groups perform their actions. During Γ_{sim} odd groups do precisely the same as during Γ_2. We only should observe that after Γ_1 even groups, during Γ_{sim} stay idle. This can be established by an analysis of possible cases. As in Γ_1 groups operate independently, we will consider one group. We should only think about non-empty buffers at a switch at stage r. Let us consider one such a buffer. The packet staying in it could not have come during Γ_1 as odd groups were idle. There are two possible causes why it did not move forward during Γ_1. If the reason was stage blocking, i.e. another packet staying in the appropriate buffer at the next stage, then this packet still remains in its buffer (as it is still blocked). If the reason was a collision in the switch, then our packet is now blocked by the winner. In both cases our packet can not move forward.

The formal proof is divided in two steps. The assumption is made here that H is even, although for odd H the proof runs analogously.

1) First we prove that $\Gamma_1 \bullet \Gamma^1_{\text{sim}} = \Gamma_1 \bullet \Gamma^1_2$. From Lemma 7.1.7 we conclude that it is enough to prove that $\sigma^r = \pi^r(\lhd\Gamma_1)$, for any even r. The principal idea now is to carefully separate the fragments of both c-maps of which σ^r, τ^r, π^r and ω^r are independent.

Now we are prepared to apply Proposition 7.1.10 together with Lemmas 7.1.9 and 7.1.12 in order to divide Γ_1 into independent parts. We obtain:

$$\Gamma_1 = \begin{pmatrix} \sigma^0 & \| \, \pi^1 \\ & \tau^1 \end{pmatrix} \| \begin{pmatrix} \sigma^2 & \| \, \pi^3 \\ \omega^2 & \| \, \tau^3 \end{pmatrix} \| \cdots \| \begin{pmatrix} \sigma^r & \| \, \pi^{r+1} \\ \omega^r & \| \, \tau^{r+1} \end{pmatrix} \| \cdots$$

It is enough to prove:

$$\sigma^r = \pi^r \left(\lhd \begin{pmatrix} \sigma^r & \| \, \pi^{r+1} \\ \omega^r & \| \, \tau^{r+1} \end{pmatrix} \right).$$

We want to consider as small pieces of the network as possible. Once more we use Proposition 7.1.10 together with the lemmas to see that it remains to show the following:

$$\sigma^{r,c} = \pi^{r,c} \left(\lhd \begin{pmatrix} \sigma^{r,c} & \| \, \sigma^{r,c^{(r)}} & \| \, \pi^{r+1,c} & \| \, \pi^{r+1,c^{(r)}} \\ \omega^{r,c} & \| \, \omega^{r,c^{(r)}} & \| \, \tau^{r+1,c} & \| \, \tau^{r+1,c^{(r)}} \end{pmatrix} \right).$$

This can easily be observed by carefully looking at the definitions of σ and τ.

A slight difference arises for $r = 0$ or $r = H$, but the argument is essentially the same.

2) Second, we prove that $\Gamma_2^1 \bullet \Gamma_{\text{sim}}^2 = \Gamma_2$. By the decomposition analogous to the above it is sufficient to show the following:

$$\begin{pmatrix} \pi^r & \| \sigma^{r+1} \\ & \tau^{r+1} \end{pmatrix} = \begin{pmatrix} \pi^r & \| \sigma^{r+1} \\ & \omega^{r+1} \end{pmatrix}, \quad \text{for even } r,$$

and after further decomposition:

$$\begin{pmatrix} \pi^{r,c} & \| \pi^{r,c()} & \| \sigma^{r+1,c} \\ & & \tau^{r+1,c} \end{pmatrix} = \begin{pmatrix} \pi^{r,c} & \| \pi^{r,c()} & \| \sigma^{r+1,c} \\ & & \omega^{r+1,c} \end{pmatrix}, \quad \text{for even } r \text{ and any } c,$$

which can be verified from the definitions.

From 1) and 2) we have $\Gamma_1 \bullet \Gamma_{\text{sim}} = \Gamma_1 \bullet \Gamma_2^1 \bullet \Gamma_{\text{sim}}^2 = \Gamma_1 \bullet \Gamma_2$.

□

The following lemma can be proved in a similar way.

Lemma 7.2.3 $\Gamma_2 \bullet \Gamma_{\text{sim}} = \Gamma_2 \bullet \Gamma_1$.

By $(P)^k$, for any Markov chain $P : A \rightsquigarrow A$, we mean the composition $P \bullet P \bullet \cdots \bullet P$ (k times). The following theorem describes the relation between the simultaneous model and the odd-even model:

Theorem 7.2.4 $(\Gamma_{\text{odd-even}})^k = \Gamma_1 \bullet (\Gamma_{\text{sim}})^{2k-1}$ *for any* $k \geq 1$.

Proof: We apply Lemmas 7.2.2 and 7.2.3 alternatingly to obtain the desired equality.

$$(\Gamma_{\text{odd-even}})^k = (\Gamma_1 \bullet \Gamma_2)^k \overset{\text{Lemma 7.2.2}}{=} (\Gamma_1 \bullet \Gamma_2)^{k-1} \bullet \Gamma_1 \bullet \Gamma_{\text{sim}} \overset{\text{Lemma 7.2.3}}{=} \cdots$$

$$\cdots \overset{\text{Lemma 7.2.3}}{=} \Gamma_1 \bullet (\Gamma_{\text{sim}})^{2k-1}.$$

□

We combine our results about the equivalences between models with the existence of stationary distributions of appropriate Markov chains.

Corollary 7.2.5 *Let* p *be the stationary distribution of* $\Gamma_{\text{odd-even}}$. *Then* p *and* $(p \bullet \Gamma_1)$ *are both stationary distributions of* $(\Gamma_{\text{sim}})^2$ *(we treat* p *as a c-map from a singleton set).*

Proof:

$$p = p \bullet (\Gamma_{\text{odd-even}})^2 \overset{\text{Theorem 7.2.4}}{=} p \bullet \Gamma_1 \bullet (\Gamma_{\text{sim}})^3 =$$

$$(p \bullet \Gamma_1 \bullet \Gamma_{\text{sim}}) \bullet (\Gamma_{\text{sim}})^2 \overset{\text{Lemma 7.2.2}}{=} (p \bullet \Gamma_{\text{odd-even}}) \bullet (\Gamma_{\text{sim}})^2 = p \bullet (\Gamma_{\text{sim}})^2,$$

and

$$p \bullet \Gamma_1 = p \bullet \Gamma_{\text{odd-even}} \bullet \Gamma_1 = p \bullet \Gamma_1 \bullet \Gamma_2 \bullet \Gamma_1 \overset{\text{Lemmas 7.2.3,7.2.2}}{=} (p \bullet \Gamma_1) \bullet (\Gamma_{\text{sim}})^2.$$

□

7.3 Odd-even versus ripple

In this section we compare the odd-even model and the ripple model, both with stage blocking. In the following we will need projection c-maps $Id_{W_1\times W_2\to W_1} : W_1 \times W_2 \rightsquigarrow W_1$, which are defined analogously to the sequential identities $Id^\bullet$ from Section 6.1.2:

$$Id_{W_1\times W_2\to W_1}((w_1,w_2),w_1) = 1.$$

The following lemma will be used, together with results from Section 7.1, to prove the main theorem of this section comparing the two models.

Lemma 7.3.1 *Let $P, S : W_1 \times W_2 \rightsquigarrow W_1$, $Q : W_1 \times W_2 \rightsquigarrow W_2$, be such that $P \dashv W_2$, $S \dashv W_2$, $Q \dashv W_1$. Then*

$$\begin{pmatrix} P & \| \, Id_{W_1\times W_2\to W_2} \\ S & \| \, Q \end{pmatrix} = \begin{pmatrix} P & \| \, Q \\ S & \| \, Id_{W_1\times W_2\to W_2} \end{pmatrix}$$

Proof: Let us denote the left side of the equality by $\mathcal{L}$ and the right one by $\mathcal{R}$. We have:

$$\mathcal{L} \overset{(1)}{=} \begin{pmatrix} P & \| \, Id_{W_1\times W_2\to W_2} \\ S & \end{pmatrix} \| \begin{pmatrix} P & \| \, Id_{W_1\times W_2\to W_2} \\ & Q \end{pmatrix} \overset{(2)}{=}$$

$$\overset{(2)}{=} \begin{pmatrix} P \\ S \end{pmatrix} \| \begin{pmatrix} Id_{W_1\times W_2\to W_2} \\ Q \end{pmatrix} \overset{(3)}{=} \begin{pmatrix} P \\ S \end{pmatrix} \| \begin{pmatrix} Q \\ Id_{W_1\times W_2\to W_2} \end{pmatrix}$$

$$\overset{(4)}{=} \begin{pmatrix} P & \| \, Q \\ S & \end{pmatrix} \| \begin{pmatrix} P & \| \, Q \\ & Id_{W_1\times W_2\to W_2} \end{pmatrix} \overset{(5)}{=} \mathcal{R}$$

Equality (1) follows from Proposition 7.1.10, since we have $S \dashv W_2$ and $Q \dashv W_1$. In (2) simplify the components by removing P and $Id_{W_1\times W_2\to W_2}$, as justified by Lemma 7.1.6. It can be verified from the definitions that Q and $Id_{W_1\times W_2\to W_2}$ can be exchanged (3). Steps (4) and (5) are similar to (1) and (2), but the direction is reversed. $\square$

In our simulation we use the c-map $\triangle : V \rightsquigarrow V$ defined as:

$$\begin{array}{rl}
\triangle = & (\pi^0 \| \, \pi^1 \| \, \pi^2 \| \, \pi^3 \| \cdots \; \| \, \pi^{H-2} \; \| \, \pi^{H-1} \; \| \, \pi^H \; \| \, \boldsymbol{\sigma^{H+1}}) \bullet \\
& (\omega^1 \; \| \, \omega^2 \| \, \omega^3 \| \cdots \; \| \, \omega^{H-2} \; \| \, \omega^{H-1} \; \| \, \omega^H \; \| \, \omega^{H+1} \;) \bullet \\
& (\pi^0 \| \, \pi^1 \| \, \pi^2 \| \, \pi^3 \| \cdots \; \| \, \pi^{H-2} \; \| \, \pi^{H-1} \; \| \, \boldsymbol{\sigma^H} \| \, \pi^{H+1} \;) \bullet \\
& (\omega^1 \; \| \, \omega^2 \| \, \omega^3 \| \cdots \; \| \, \omega^{H-2} \; \| \, \omega^{H-1} \; \| \, \omega^H \; \| \, \boldsymbol{\tau^{H+1}}) \bullet \\
& (\pi^0 \| \, \pi^1 \| \, \pi^2 \| \, \pi^3 \| \cdots \; \| \, \pi^{H-2} \; \| \, \boldsymbol{\sigma^{H-1}} \| \, \pi^H \; \| \, \boldsymbol{\sigma^{H+1}}) \bullet \\
& (\omega^1 \; \| \, \omega^2 \| \, \omega^3 \| \cdots \; \| \, \omega^{H-2} \; \| \, \omega^{H-1} \; \| \, \boldsymbol{\tau^H} \| \, \omega^{H+1} \;) \bullet \\
& (\pi^0 \| \, \pi^1 \| \, \pi^2 \| \, \pi^3 \| \cdots \; \| \, \boldsymbol{\sigma^{H-2}} \| \, \pi^{H-1} \; \| \, \boldsymbol{\sigma^H} \| \, \pi^{H+1} \;) \bullet \\
& (\omega^1 \; \| \, \omega^2 \| \, \omega^3 \| \cdots \; \| \, \omega^{H-2} \; \| \, \boldsymbol{\tau^{H-1}} \| \, \omega^H \; \| \, \boldsymbol{\tau^{H+1}}) \bullet \\
& \qquad \vdots \qquad\qquad\qquad\qquad\qquad \vdots
\end{array}$$

$$\begin{array}{lccccccccr}
(\pi^0 & \| \pi^1 & \| \boldsymbol{\sigma^2} & \| \pi^3 & \| \cdots & \| \boldsymbol{\sigma^{H-2}} & \| \pi^{H-1} & \| \boldsymbol{\sigma^H} & \| \pi^{H+1} &)\bullet \\
 & (\omega^1 & \| \omega^2 & \| \boldsymbol{\tau^3} & \| \cdots & \| \omega^{H-2} & \| \boldsymbol{\tau^{H-1}} & \| \omega^H & \| \boldsymbol{\tau^{H+1}} &)\bullet \\
(\pi^0 & \| \boldsymbol{\sigma^1} & \| \pi^2 & \| \boldsymbol{\sigma^3} & \| \cdots & \| \pi^{H-2} & \| \boldsymbol{\sigma^{H-1}} & \| \pi^H & \| \boldsymbol{\sigma^{H+1}} &)\bullet \\
 & (\omega^1 & \| \boldsymbol{\tau^2} & \| \omega^3 & \| \cdots & \| \boldsymbol{\tau^{H-2}} & \| \omega^{H-1} & \| \boldsymbol{\tau^H} & \| \omega^{H+1} &)\bullet
\end{array}$$

It consists, intuitively speaking, of consecutive odd-even steps (Γ_2 followed by Γ_1, this followed by Γ_2 and so on), which are partially inactive. First Γ_2 acts only on the $(H+1)$st stage (idling on other stages), next Γ_1 acts on the Hth and the $(H+1)$st stage. This it is continued until the last step, which is simply full Γ_2, assumed that H is even. When H is odd, we should remove the last step (equal to Γ_2) from $\triangle$.

The following theorem gives a link between the odd-even model and the ripple model.

Theorem 7.3.2 $\triangle \bullet \Gamma_{\text{odd-even}} = \Gamma_{\text{ripple}} \bullet \triangle$.

Proof: The proof is given for even H; for odd H only a slight modification is needed. If we look carefully at the definition of $A = \triangle \bullet \Gamma_{\text{odd-even}}$ and $B = \Gamma_{\text{ripple}} \bullet \triangle$, we notice that the number of c-maps σ, τ (at any stage of the network) is the same in both of them. Moreover, in $(\triangle \bullet \Gamma_{\text{odd-even}})$ we can distinguish two fragments, one (consisting of the earliest σ's and τ's from all stages) being exactly Γ_{ripple} and another, which is equal to $\triangle$. Below we have c-map A (the part in bold should correspond to the "triangle" component of B). For clarity, the idle actions ω and π are suppressed.

$$\triangle \bullet \Gamma_{\text{odd-even}} = \begin{array}{lccccccccr}
(& \| & \| & \| & \| \cdots & \| & \| & \| & \| \sigma^{H+1} &)\bullet \\
 & (& \| & \| & \| \cdots & \| & \| & \| & \| &)\bullet \\
(& \| & \| & \| & \| \cdots & \| & \| & \| \sigma^H & \| &)\bullet \\
 & (& \| & \| & \| \cdots & \| & \| & \| & \| \tau^{H+1} &)\bullet \\
(& \| & \| & \| & \| \cdots & \| & \| \sigma^{H-1} & \| & \| \boldsymbol{\sigma^{H+1}} &)\bullet \\
 & (& \| & \| & \| \cdots & \| & \| & \| \tau^H & \| &)\bullet \\
(& \| & \| & \| & \| \cdots & \| \sigma^{H-2} & \| & \| \boldsymbol{\sigma^H} & \| &)\bullet \\
 & (& \| & \| & \| \cdots & \| & \| \tau^{H-1} & \| & \| \boldsymbol{\tau^{H+1}} &)\bullet \\
 & \vdots & & & & & & \vdots & &
\end{array}$$

$$\begin{array}{lccccccccr}
(& \| & \| \sigma^2 & \| & \| \cdots & \| \boldsymbol{\sigma^{H-2}} & \| & \| \boldsymbol{\sigma^H} & \| &)\bullet \\
 & (& \| & \| \tau^3 & \| \cdots & \| & \| \boldsymbol{\tau^{H-1}} & \| & \| \boldsymbol{\tau^{H+1}} &)\bullet \\
(& \| \sigma^1 & \| & \| \boldsymbol{\sigma^3} & \| \cdots & \| & \| \boldsymbol{\sigma^{H-1}} & \| & \| \boldsymbol{\sigma^{H+1}} &)\bullet \\
 & (& \| \tau^2 & \| & \| \cdots & \| \boldsymbol{\tau^{H-2}} & \| & \| \boldsymbol{\tau^H} & \| &)\bullet \\
(\sigma^0 & \| & \| \boldsymbol{\sigma^2} & \| & \| \cdots & \| \boldsymbol{\sigma^{H-2}} & \| & \| \boldsymbol{\sigma^H} & \| &)\bullet \\
 & (\tau^1 & \| & \| \boldsymbol{\tau^3} & \| \cdots & \| & \| \boldsymbol{\tau^{H-1}} & \| & \| \boldsymbol{\tau^{H+1}} &)\bullet \\
(& \| \boldsymbol{\sigma^1} & \| & \| \boldsymbol{\sigma^3} & \| \cdots & \| & \| \boldsymbol{\sigma^{H-1}} & \| & \| \boldsymbol{\sigma^{H+1}} &)\bullet \\
 & (& \| \boldsymbol{\tau^2} & \| & \| \cdots & \| \boldsymbol{\tau^{H-2}} & \| & \| \boldsymbol{\tau^H} & \| &)
\end{array}$$

The only problem is that in $(\triangle \bullet \Gamma_{\text{odd}-\text{even}})$ the ripple and "triangle" parts proceed "almost simultaneously", while in $(\Gamma_{\text{ripple}} \bullet \triangle)$ they are composed sequentially. We will prove that A can be obtained from B by moving the triangle $\triangle$ up, in such a way that the the c-map is unchanged. This will be done in several steps, each of them consisting of an atomic exchange allowed by Lemma 7.3.1.

We can view the Markov chain Γ_{ripple} as a sequential composition of c-maps working on consecutive stages:

$$\Gamma_{\text{ripple}} = \Gamma^{H+2}_{\text{ripple}} \bullet \Gamma^{H+1}_{\text{ripple}} \bullet \cdots \bullet \Gamma^{1}_{\text{ripple}}.$$

$\Gamma^{i}_{\text{ripple}}$ denotes the flow out of packets from the $(i-1)$st stage. Analogously we decompose $\triangle$ into several sequential phases:

$$\triangle = \triangle^{H+1} \bullet \triangle^{H} \bullet \cdots \bullet \triangle^{1}.$$

We number atomic steps (exchanges) (i, j) where $i = (H+1), \ldots, 2$, $j = 1, \ldots, (i-1)$. During step (i, j) we move $\triangle^i$ up and compose it in parallel with $\Gamma^{j}_{\text{ripple}}$.

Formally we apply Lemma 7.3.1 to the following c-maps:

$$P = \begin{pmatrix} \pi^0 & \| \, \pi^1 & \| \cdots & \| \, \pi^{j-2} & \| \, \sigma^{j-1} & \| \, \pi^j \\ & \omega^1 & \| \cdots & \| \, \omega^{j-2} & \| \, \omega^{j-1} & \| \, \tau^j \end{pmatrix},$$

$$S = \begin{pmatrix} \pi^0 & \| \, \pi^1 & \| \cdots & \| \, \sigma^{j-2} & \| \, \pi^{j-1} & \| \, \pi^j \\ & \omega^1 & \| \cdots & \| \, \omega^{j-2} & \| \, \tau^{j-1} & \| \, \omega^j \end{pmatrix}, \quad \text{and}$$

$$Q = \begin{pmatrix} \pi^{j+1} & \| \, \pi^{j+2} & \| \cdots & \| \, \sigma^i & \| \, \pi^{i+1} & \| \, \sigma^{i+2} & \| \, \pi^{i+3} & \| \, \sigma^{i+4} & \| \cdots \\ \omega^{j+1} & \| \, \omega^{j+2} & \| \cdots & \| \, \omega^i & \| \, \tau^{i+1} & \| \, \omega^{i+2} & \| \, \tau^{i+3} & \| \, \omega^{i+4} & \| \cdots \end{pmatrix}.$$

An idling c-map $J = Id_{V \to V_{j+1} \times \ldots \times V_{H+1}}$ is a projection defined as

$$J = \begin{pmatrix} \pi^{j+1} & \| \, \pi^{j+2} & \| \cdots & \| \, \pi^{H+1} \\ \omega^{j+1} & \| \, \omega^{j+2} & \| \cdots & \| \, \omega^{H+1} \end{pmatrix},$$

where $V_r = S^{I_r}$ represents rth stage of the network. Notice that P and S are simply fragments of two neighboring ripple stages ($\Gamma^{j}_{\text{ripple}}$ and $\Gamma^{j-1}_{\text{ripple}}$, respectively) and Q is a fragment of $\triangle^i$. The c-map $\Gamma^{j}_{\text{ripple}}$ can be decomposed into $P \parallel J$ (by Lemma 7.2.1 from Section 7.2), and similarly we obtain $S \parallel Q$. For $j = 1$, the c-map S should be slightly modified: σ^{j-2} and τ^{j-1} should be replaced by π^{j-2} and ω^{j-1}, respectively. Moreover, in this case, after applying Lemma 7.3.1, c-map $S \parallel J$ consists only of π's and ω's, thus $S \parallel J = Id^{\bullet}_{V}$, so we remove it (by Proposition 6.1.3). □

Remark In the proof above we need only very general properties of c-maps $\sigma^{r,c}$ and $\tau^{r,c}$, mainly the independence from fragments of their domains (as in Lemma 7.2.1). This means that the proof applies to any other model fulfilling this requirement. Especially it will be adapted in Section 7.4 to the relaxed model of packet flow.

The following corollary from Theorem 7.3.2 can be easily proved by induction:

Corollary 7.3.3 $\triangle \bullet (\Gamma_{\mathrm{odd-even}})^k = (\Gamma_{\mathrm{ripple}})^k \bullet \triangle$, *for any* $k \geq 1$.

Theorem 7.3.2 gives us a kind of simulation of the ripple model by the odd-even model:

Corollary 7.3.4 *Let* p, q *be probability distributions on* V. *Then*

$$p \bullet (\Gamma_{\mathrm{ripple}})^k = q \qquad \textit{implies } (p \bullet \triangle) \bullet (\Gamma_{\mathrm{odd-even}})^k = q \bullet \triangle.$$

Having clarified the relation between Γ_{ripple} and $\Gamma_{\mathrm{odd-even}}$, we conclude with the comparison of stationary distributions.

Corollary 7.3.5 *Let* p_{ripple} *and* $p_{\mathrm{odd-even}}$ *denote stationary distributions of Markov chains* Γ_{ripple} *and* $\Gamma_{\mathrm{odd-even}}$, *respectively. Then we have:*

$$p_{\mathrm{ripple}} \bullet \triangle = p_{\mathrm{odd-even}}$$

(p_{ripple} *and* $p_{\mathrm{odd-even}}$ *are treated here as c-maps from a singleton set).*

7.4 Variant with relaxation

The equivalences in the relaxed case are not the same as in full blocking. Surprisingly, it turns out that the ripple model and the simultaneous model are identical, which we show in Theorem 7.4.1. On the other hand Theorem 7.3.2 can be adapted to the relaxed model, which is done in Theorem 7.4.2.

Theorem 7.4.1 *Ripple and simultaneous models are equal in the relaxed case, i.e.*

$$\Gamma^*_{\mathrm{ripple}} = \Gamma^*_{\mathrm{sim}}.$$

Proof: We show by induction that all rows of $\Gamma^*_{\mathrm{ripple}}$ can be "collapsed" into 2 rows of Γ^*_{sim}. In the proof we suppress the indices of c-maps π and ω, to improve readability. We start with the first four rows of $\Gamma^*_{\mathrm{ripple}}$ (base of induction) and show the following:

$$\begin{pmatrix} \pi^H \| \sigma_*^{H+1} \\ \omega \| \omega \\ \sigma_*^H \| \pi \\ \omega \| \tau_*^{H+1} \end{pmatrix} = \begin{pmatrix} \sigma_*^H \| \sigma_*^{H+1} \\ \omega \| \tau_*^{H+1} \end{pmatrix} \tag{7.2}$$

The c-maps in (7.2) are only fragments of the appropriate rows of Γ^*_{ripple} and Γ^*_{sim}, but it is easy to observe that they can be "separated" (by Proposition 7.1.10) from the rest, consisting in both cases of only idling c-maps, thus equal to a projection $Id_{V\to V_1\times\cdots\times V_{H-1}}$ cf. Section 7.3.

Notice that the last rows in (7.2) are the same on both sides. Consider the first three rows of the left side. We have, by Propositions 7.1.10 and 7.1.6:

$$\begin{pmatrix} \pi\| \ \sigma_*^{H+1} \\ \omega\| \ \omega \\ \sigma_*^{H}\| \ \pi \end{pmatrix} = \begin{pmatrix} \pi\| \ \sigma_*^{H+1} \\ \omega\| \ \omega \\ \sigma_*^{H} \end{pmatrix} \| \begin{pmatrix} \pi \ \|\sigma_*^{H+1} \\ \omega \ \| \ \ \omega \\ \pi \end{pmatrix} =$$

$$= \begin{pmatrix} \pi \\ \omega \\ \sigma_*^{H} \end{pmatrix} \| \begin{pmatrix} \sigma_*^{H+1} \\ \omega \\ \pi \end{pmatrix} = \begin{pmatrix} \sigma_*^{H}\| \ \sigma_*^{H+1} \end{pmatrix},$$

which proves (7.2). We could apply Proposition 7.1.10 and Lemma 7.1.6, since σ_*^r is independent from domains just as σ^r and moreover we have: $\sigma_*^r \dashv V_{r+1}$, for $r \leq H$. As the induction step we should show the following:

$$\begin{pmatrix} \pi & \| \ \sigma_*^i\| \ \mathcal{S}^{i+1} \\ \omega & \| \ \omega \ \| \ \mathcal{T}^{i+1} \\ \sigma_*^{i-1} & \| \ \pi \ \| \ \pi \\ \omega & \| \ \tau_*^i\| \ \omega \end{pmatrix} = \begin{pmatrix} \sigma_*^{i-1}\| \ \sigma_*^i\| \ \mathcal{S}^{i+1} \\ \omega \quad \| \ \tau_*^i\| \ \mathcal{T}^{i+1} \end{pmatrix}, \tag{7.3}$$

where $\mathcal{S}^{i+1} =\|_{j=i+1\cdots,H+1} \ \sigma_*^j$ and $\mathcal{T}^{i+1} =\|_{j=i+1,\cdots,H+1} \ \tau_*^j$. Similarly as in (7.2), we omit here some part of Γ^*_{ripple} and Γ^*_{sim} consisting of idling c-maps, which is equal to a projection $Id_{V\to V_1\times\cdots\times V_{i-2}}$. In the following let $\mathcal{L}$ and $\mathcal{R}$ denote the left and right hand side of (7.3).

Let us begin by transforming the left hand side of (7.3). The first two rows can be decomposed as follows:

$$\begin{pmatrix} \pi\| \ \sigma_*^i\| \ \mathcal{S}^{i+1} \\ \omega\| \ \omega \ \| \ \mathcal{T}^{i+1} \end{pmatrix} = \begin{pmatrix} \pi\| \ \sigma_*^i\| \ \mathcal{S}^{i+1} \\ \omega\| \ \omega \end{pmatrix} \| \begin{pmatrix} \pi\| \ \sigma_*^i\| \ \mathcal{S}^{i+1} \\ \mathcal{T}^{i+1} \end{pmatrix} \tag{7.4}$$

$$= \begin{pmatrix} \pi\| \ \sigma_*^i \\ \omega\| \ \omega \end{pmatrix} \| \begin{pmatrix} \sigma_*^i \ \|\mathcal{S}^{i+1} \\ \mathcal{T}^{i+1} \end{pmatrix}.$$

Now we take the third and the fourth row:

$$\begin{pmatrix} \sigma_*^{i-1}\| \ \pi \ \| \ \pi \\ \omega \quad \| \ \tau_*^i\| \ \omega \end{pmatrix} = \begin{pmatrix} \sigma_*^{i-1}\| \ \pi \ \| \ \pi \\ \omega \quad \| \ \tau_*^i \end{pmatrix} \| \begin{pmatrix} \sigma_*^{i-1}\| \ \pi\| \ \pi \\ \omega \end{pmatrix} \tag{7.5}$$

$$= \begin{pmatrix} \sigma_*^{i-1}\| \ \pi \\ \omega \quad \| \ \tau_*^i \end{pmatrix} \| \begin{pmatrix} \pi \\ \omega \end{pmatrix}.$$

We combine (7.4) and (7.5) to obtain the following, after applying Proposition 7.1.10 and Lemma 7.1.6 once more:

$$\mathcal{L} = \begin{pmatrix} \pi & \| \, \sigma_*^i \\ \omega & \| \, \omega \\ \sigma_*^{i-1} & \| \, \pi \\ \omega & \| \, \tau_*^i \end{pmatrix} \| \begin{pmatrix} \sigma_*^i \, \| \mathcal{S}^{i+1} \\ \mathcal{T}^{i+1} \\ \pi \\ \omega \end{pmatrix}. \tag{7.6}$$

Look at the right side of (7.3):

$$\mathcal{R} = \begin{pmatrix} \sigma_*^{i-1} \| \, \sigma_*^i \| \, \mathcal{S}^{i+1} \\ \omega \quad \| \, \tau_*^i \end{pmatrix} \| \begin{pmatrix} \sigma_*^{i-1} \| \, \sigma_*^i \| \, \mathcal{S}^{i+1} \\ \| \, \mathcal{T}^{i+1} \end{pmatrix} \tag{7.7}$$

$$= \begin{pmatrix} \sigma_*^{i-1} \| \, \sigma_*^i \\ \omega \quad \| \, \tau_*^i \end{pmatrix} \| \begin{pmatrix} \sigma_*^i \| \, \mathcal{S}^{i+1} \\ \| \, \mathcal{T}^{i+1} \end{pmatrix}$$

We see that the right parallel components in (7.6) and (7.7) are identical (as $\binom{\pi^{i+1}}{\omega^{i+1}}$ is a projection c-map $Id_{V \to V_{i+1}}$). To complete the proof we should prove the equality of the left components:

$$\begin{pmatrix} \pi \| \, \sigma_*^i \\ \omega \| \, \omega \\ \sigma_*^{i-1} \| \, \pi \\ \omega \| \, \tau_*^i \end{pmatrix} = \begin{pmatrix} \sigma_*^{i-1} \| \, \sigma_*^i \\ \omega \| \, \tau_*^i \end{pmatrix},$$

but this is exactly the equality (7.2) we have already proved at the beginning (with σ_*^i replaced by σ_*^{H+1} and τ_*^i by τ_*^{H+1}, but the argument remains the same). $\square$

Note In the proof above we used only a very general property of relaxation, mainly that σ_*^r is independent from every stage different than r. This means that the proof applies to any other relaxed model fulfilling this requirement. The analogue to Theorem 7.3.2 holds:

Theorem 7.4.2 *Let $\triangle^*$ be defined analogously to $\triangle$ (we replace σ,τ, by σ_*,τ_*, resp.), then:*

$$\triangle^* \bullet \Gamma^*_{\text{odd-even}} = \Gamma^*_{\text{ripple}} \bullet \triangle^* = \Gamma^*_{\text{sim}} \bullet \triangle^*.$$

Proof: We can repeat here the proof of Theorem 7.3.2, replacing σ by σ_*,in the proof it is only assumed that σ is independent from certain parts of its domain, and σ_* also satisfies this requirement. $\square$

And finally for the relaxed case by Theorem 7.4.1 and 7.4.2 we obtain, analogously as in Corollary 7.3.5 in Section 7.3:

Corollary 7.4.3 *The Markov chains Γ^*_{ripple} and Γ^*_{sim} have the same stationary distribution (say p) and the stationary distribution of $\Gamma^*_{\text{odd-even}}$ is equal to $(p \bullet \triangle^*)$ (p is treated here as a c-map from a singleton set).*

7.5 Summary of Chapters 6 and 7

We have presented the formal description of several models of packet flow in the butterfly MINs by means of ergodic Markov chains. By varying the definitions of c-maps σ and τ one can obtain the number of different models, without introducing any changes in the main definitions. For example the relaxed blocking model of [56] was easily obtained in this way. It is interesting to observe how the correspondence between ripple, odd-even and simultaneous changes when we modify σ and τ.

We considered butterfly MINs with (2×2)-switches having buffers of capacity one, but all the results can be easily repeated for $(k \times k)$-switches, $k \geq 3$. All the considerations can be repeated for buffers of greater capacity as well, assumed that bandwidth of connecting links is equal to this capacity (otherwise Theorem 7.2.4 and associated lemmas do not hold). Another possible continuation of the work is to examine the usefulness of the presented notation in the analysis of the performance of full blocking MIN. Since many years this is a challenging problem, and by now there are no results on throughput in the full blocking MIN model with finite buffers. As there are asymptotic results on throughput in relaxed blocking model [56], the promising direction is to study relationships between relaxed and non-relaxed models, aiming at relating throughputs.

The only available way of estimating performance in full blocking model is by time consuming computer simulations. This motivates the question whether the Markov chains defined by us are *rapidly mixing* [37, 39], i.e. whether they converge quickly to their stationary distributions, which would imply the existence of efficient algorithms for approximating throughput and other efficiency measures.

Final remarks

In this thesis we have discussed several methods efficiently solving various problems arising in Markov modeling of complex systems, e.g. communication networks, using probabilistic and combinatorial techniques. Although the probabilistic Markov chain models we considered are just simplified mathematical abstractions of problems encountered in practice, they allow to compute many interesting parameters of a system.

The rapid development of modern communication networks and the need of algorithm solving hard computational problems resulted in the wide family of Markov chain models. The most important common feature of these chains is the big size of their state space. This often eliminates the possibility of finding the exact solution and the approximation algorithms remain as the only way to solve the problem.

In this thesis we studied a new class of approximation algorithms based on the combinatorial approach proposed by Pokarowski [52]. We analyzed the complexity of algorithms and studied their applicability on several Markov models of communication systems. Both analytic and experimental results obtained by us classify this new method as a potentially very useful tool in practice.

In the next part of the thesis we focused on the famous conjecture about the expansion property of matroids. The positive resolution of this conjecture is of major combinatorial and algorithmic significance as it would yield an to approximation algorithm for counting the number of matroid bases. In Chapter 5 we defined sufficient conditions on matroid to induce the expansion, i.e. we solved the conjecture in the special case of exchange preserving matroids. Our new approach explores the structure of exchanges between bases and focuses on strong orderable bijections in the internal exchange graphs.

The last part of the thesis is devoted to studying the dynamic behavior of large communication networks. In Chapter 6 we propose new method of describing the complex systems of this kind. The main advantage of this approach is its decomposability. A model of large system is built up from smaller segments being well defined mathematical objects. The calculus proposed by us allowed to reduce the complexity of description. We demonstrate how this simple formal description can be useful in proving different properties of Markov chains related to these models.

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

Open problems:

(1) The combinatorial aggregation approach is a new very efficient algorithmic method for solving problems related to Markov chains, e.g. computing the stationary distribution. It would be useful to design procedures for other Markov chain characteristics based on this theory.

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

(3) The largest class of matroids, for which the expansion was proved is the class of *balanced matroids* [19]. The natural question is to relate balanced matroids and *exchange-preservation* matroids, for which the expansion follows from Theorem 5.2.2.

(4) The challenging open question is to try to enlarge the class of matroids satisfying exchange-preservation property by applying other techniques for estimating graph expansion. An upper bound for applicability of such method along the line of Chapter 5 is the class of all strong base orderable matroids.

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

(6) When the analytical results are impossible to obtain, and the simulation are extremely time consuming, the attractive way to obtain solution of the problem related to a Markov chain is to compute it numerically. The natural continuation of our work it to investigate the possibility of performing numerical computations, such as vector multiplication, on the compressed description of the chain.

Bibliography

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

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

[3] Batcher, K.: Sorting networks and their applications, *Proc. of the AFIPS Spring Joint Computing Conference*, **vol. 32**, 1968, pp. 307-314.

[4] Bondy, J.: Transversal Matroids, Base-Orderable Matroids and Graphs. *Quart. J. Math. Oxford*, **vol. 23**, 1972, pp. 81-89.

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

[6] BBN, *Butterfly^{TM} Parallel Processor Overview*, BBN Report No. 6148, Version 1, Cambridge Mass. 1986.

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

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

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.*

[14] Diekmann, R., Lüling, R. and Monien, B., Communication Throughput of Interconnection Networks, *Proc. MFCS 1994, Springer LNCS 841*, pp. 72-86.

[15] Dietzfelbinger M., Gambin, A., Lasota, S.: On Different Models for Packet Flow in Multistage Interconnection Networks, *raport techniczny UW* 97-11(248) oraz *Proc. Workshop Concurrency Specification and Programming*, Berlin 1998.

[16] Dyer, M.E., Frieze, A. and Kannan, R.: A random polynomial time algorithm for approximating the volume of convex bodies. *Journal of the ACM,* **vol. 38**, 1991, pp. 1-17.

[17] Edmonds, J.: Submodular functions, Matroids and certain Polyhedra. *Proceedings of Calgary International Conference on Combinatorial structures and their applications*, 1969.

[18] Farber, M.: Basis pair graphs of transversal matroids are connected. *Discrete Mathematics* **vol. 73**, 1989, pp. 245-248.

[19] Feder, T., Mihail, M.: Balanced Matroid. *Proceedings of the 24th ACM Symposium on Theory of Computing*, 1992, pp. 26-38.

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

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

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

[23] Gabow, H.: Decomposing symmetric exchanges in matroid bases. *Mathematical Programming*, **vol. 10**, 1976, pp. 271-276.

[24] Gambin, A. and Lasota, S.: On the Semantics of Multistage Interconnection Networks, *Proc. XXIII-rd Seminar on Current Trends in Theory and Practice of Informatics SOFSEM*, 1996, LNCS 1175, pp. 359-368.

[25] Gambin, A. On Approximation of the Number of Bases for Exchange Preserving Matroids, *Proc. Mathematical Foundations of Computer Science, Springer LNCS 1672*, 1999.

[26] Gambin, A. and Pokarowski, P: On new class of aggregation algorithms, *manuscript.*

[27] Goke, G. F. and Lipovski, G. J.: Banyan Network for Partitioning Multiprocessor Systems, *Proc. 1st Ann. Symp. on Computer Architecture*, 1973, pp. 21-28.

[28] Gottlieb, A.: An overview of the NYU Ultracomputer Project, in J.J. Dongarra, *Experimental Parallel Computing Architectures* Elsevier Amsterdam, 1987, pp. 25-95.

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

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

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

[32] Hoare, C.A.R.: *Communicating Sequential Processes*, Prentice – Hall, 1985.

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

[34] Jenq, Y.C.: Performance analysis of a packet switch based on a single buffered banyan network, *IEEE Journal on Selected Areas of Comm.*, **vol. SAC-3**, 1983, pp. 1014-1021.

[35] Jerrum, M.R., Sinclair, A.J.: Conductance and rapid mixing property for Markov chains: the approximation of the permanent resolved. *Proceedings of the 20th ACM Symposium on Theory of Computing*, 1988, pp. 235-244.

[36] Jerrum, M.R., Valiant, L.G. and Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, **vol. 43**, 1986, pp. 169-188.

[37] Jerrum, M. R. and Sinclair, A. J.: Approximate counting, uniform generation and rapidly mixing Markov chains, *Information and Computation*, **vol. 82**, 1989, pp. 93-133.

[38] 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.

[39] Kannan, R., Markov chains and polynomial time algorithms, *35th IEEE Symposium on Foundations of Computer Science*, 1994, pp. 656-671.

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

[41] Koch, R. R.: Increasing the Size of a Network by a Constant Factor Can Increase Performance by More Than a Constant Factor, *Proc. 29th Ann. Symposium on Foundations of Computer Science*, 1988, pp. 221-231.

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

[43] Leighton, T. F.: *Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*, Morgan Kaufmann, 1992.

[44] Lomeli, L.C. and Welsh, D.: Randomized Approximation of the Number of Bases. *Contemporary Mathematics*, **vol. 197**, 1996, pp. 371-376.

[45] Maurer, S.B.: Matroid Basis Graphs I and II. *Journal of Combinatorial Theory (B)*, **vol. 14**, pp. 216-240 and **vol. 15**, pp. 121-145, 1993.

[46] Mihail, M. and Sudan, M.: Connectivity properties of matroids, *Technical Report, U.C. Berkeley*, 1989.

[47] Mihail, M. and Vazirani, U.: On the Expansion of 0-1 Polytopes, *Technical Report 3, Aiken Laboratory, Harvard University*, 1989.

[48] Mun, Y. and Yong, H.: Performance Analysis of Finite Buffered Multistage Interconnection Networks, *IEEE Trans. on Computers*, **vol. C-43**, 1994, pp. 152-162.

[49] Oxley, J.G.: *Matroid Theory*, Oxford University Press, New York, 1992.

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

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

[52] Pokarowski, P. Directed forests and algorithms related to Markov chains, *PhD thesis* Institute of Mathematics, Polish Academy of Sciences, 1998.

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

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

[55] Ranade, A.G: How to emulate shared memory, *Proc. 28th Ann. Symposium on Foundations of Computer Science*, 1987, pp. 185-194.

[56] Rehrmann, R., Monien, B., Lüling, R., Diekmann, R.: On the Communication Throughput of Buffered Multistage Interconnection Networks, *Proc. 8th ACM Symposium on Parallel Algorithms and Architectures*, 1996.

[57] Rooholaminiight, R.: Finding the Right ATM Switch for the Market, *IEEE Journal Comp.*, 1994, pp. 16-28.

[58] 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.

[59] Sinclair, A.J.: *Algorithms for Random Generation and Counting.* Progress in theoretical computer science, Birkhäuser, Boston, 1993.

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

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

[62] 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.

[63] 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.

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

[65] Theimer, T.H., Rathgeb, E.P., Huber, M.N.: Performance Analysis of Buffered Banyan Networks *IEEE Trans. on Communication*, **vol. C-39**, 1991, pp. 269-277.

[66] Valiant, L.G.: A scheme for fast parallel communication, *SIAM Journal on Computing*, **vol. 11, no. 2**, 1982, pp. 350-361.

[67] Welsh, D.: *Matroid Theory*, Academic Press, 1976.

[68] White, N.: A Unique Exchange Property for Bases. *Linear Algebra and its Application*, **vol. 31**, 1980, pp. 81-91.

[69] Yoon, H.Y., Lee, K.Y., Liu, M.T.: Performance Analysis of Multibuffered Packet-switching Networks in Multiprocessor Systems, *IEEE Trans. on Computers*, **vol. C-39**, 1990, pp. 319-327.