# Stabilization of Branching Queueing Networks*

**Tomáš Brázdil**[1] **and Stefan Kiefer**[2]

1 **Faculty of Informatics, Masaryk University, Czech Republic**
brazdil@fi.muni.cz
2 **Department of Computer Science, University of Oxford, United Kingdom**
stefan.kiefer@cs.ox.ac.uk

## Abstract

Queueing networks are gaining attraction for the performance analysis of parallel computer systems. A Jackson network is a set of interconnected servers, where the completion of a job at server $i$ may result in the creation of a new job for server $j$. We propose to extend Jackson networks by "branching" and by "control" features. Both extensions are new and substantially expand the modelling power of Jackson networks. On the other hand, the extensions raise computational questions, particularly concerning the stability of the networks, i.e, the ergodicity of the underlying Markov chain. We show for our extended model that it is decidable in polynomial time if there exists a controller that achieves stability. Moreover, if such a controller exists, one can efficiently compute a static randomized controller which stabilizes the network in a very strong sense; in particular, all moments of the queue sizes are finite.

**1998 ACM Subject Classification** G.3 Probability and Statistics

**Keywords and phrases** continuous-time Markov decision processes, infinite-state systems, performance analysis

**Digital Object Identifier** 10.4230/LIPIcs.STACS.2012.507

## 1 Introduction

Queueing theory plays a central role in the performance analysis of computer systems. In particular, *queueing networks* are gaining attraction as models of parallel systems. A queueing network is a set of processing units (called *servers*), each of which performs tasks (called *jobs*) of a certain type. Each server has its own queue of jobs waiting to be processed. The successful completion of a job may trigger one (or more) new jobs (of possibly different type) that need to be processed as well. In addition to this "internal" job creation, so-called *open* queueing networks allow for new jobs to arrive "externally", i.e., from outside.

Queueing networks are a popular model for both hardware and software systems because of their simplicity and generality. On the hardware side, queueing networks can, e.g., be used for modeling multi-core processors, see e.g. [28] and the references in [8]. One advantage of queueing-based analyses is their scalability with growing parallelism; e.g., it is said in [20]: "Cycle-accurate full-system performance simulators do not scale well beyond a few tens of processor cores at best. As such, analytical models based on the theory of queueing systems, are a logical choice for developing a basic understanding of the fundamental tradeoffs in future, large-scale multi-core systems." On the software side, queueing networks are used for modeling message passing. It is said in [27]: "Two natural classes of systems can be modeled

* Tomáš Brázdil is supported by the Czech Science Foundation, grant No. P202/10/1469. Stefan Kiefer is supported by a DAAD postdoctoral fellowship.


29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 507–518

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

using such a framework: asynchronous programs on a multi-core computer and distributed programs communicating on a network." Of course, the realm of queueing networks stretches far beyond computer science, see [3, 7].

The simplest queueing networks are so-called *Jackson networks* [15]: Given two servers $i, j \in \{1, \dots, n\}$, there is a "rule" of the form $i \xhookrightarrow{p_{ij}} j$ which specifies the probability $p_{ij}$ that the completion of an $i$-job results in the creation of a $j$-job. There are also rules $i \xhookrightarrow{p_{i0}} \varepsilon$ where $p_{i0} = 1 - \sum_j p_{ij}$ specifies the probability that no new job is created. Each server $i$ has a *rate* $\mu_i$ with which an $i$-job is processed if there is one. In addition, there is a rate $\alpha_i$ with which $i$-jobs arrive from outside the network. The processing times and the external arrivals are exponentially distributed, so that a Jackson network describes a *continuous-time Markov chain (CTMC)*. It was shown in Jackson's paper [15] that if the rate $\lambda_i$ of internal *and* external arrivals at server $i$ is less than $\mu_i$ for all $i$, then the network is *stable*, i.e., the average queue length is finite and almost surely all queues are empty infinitely often. Moreover, Jackson networks allow a *product form*, i.e., the steady-state distribution of the queue lengths can be expressed as a product of functions $\pi_i(k)$, where $\pi_i(k)$ is the steady-state probability that queue $i$ has length $k$.

▶ **Example 1** (network processor)**.** In [1], Jackson networks are used to model network processors, i.e., chips that are specifically targeted at networking applications—think of a router. We describe the model from [1] (sections 4.1 and 4.2, slightly adapted). Before packets are processed in the "master processor" $M$, they pass through the "data plane" $D$, from which a fraction $q$ of packets needs to be processed first in the "control plane" $C$:

$$D \xhookrightarrow{1-q} M \qquad D \xhookrightarrow{q} C \qquad C \xhookrightarrow{1} M \qquad M \xhookrightarrow{1} \varepsilon$$

An "arrival manager" $A$ sends some packets (fraction $d_0$) directly to $D$, but others (fractions $d_1, \dots, d_n$ with $d_0 + d_1 + \cdots + d_n = 1$) are sent to "slave processors" $S_1, \dots, S_n$ to assist the master. Some packets (fraction $b$) still need the attention of the master after having been processed by a slave:

$$A \xhookrightarrow{d_0} D \qquad A \xhookrightarrow{d_i} S_i \qquad S_i \xhookrightarrow{b} D \qquad S_i \xhookrightarrow{1-b} \varepsilon \quad , \quad i \in \{1, \dots, n\}.$$

Jackson networks and their extensions have been thoroughly studied, but they are restricted in their modelling capabilities, as (i) the completion of a job may trigger at most one job, and (ii) there is no nondeterminism that would allow to control the output probabilities of a server. Considering (i), it seems unnatural to assume that a distributed program communicating on a network produces at most one message at the end of its computation. Considering (ii), the "arrival manager" $A$ in Example 1 may want to flexibly pass incoming packets to the master or one of the slaves, possibly depending on the current load. These restrictions have not been fully addressed, not even in isolation. In this paper we introduce *controlled branching queueing networks*, which are Jackson-like networks but allow for both nondeterminism ("controlled") and the creation of more than one job ("branching").

Both extensions directly raise computational issues. We show in Example 2 on page 511 that even purely stochastic branching networks do not allow a product form, which illustrates the mathematical challenge[1] of this extension and poses the question for an effective criterion that allows to determine whether the network is stable, i.e., returns to the empty state

[1] It is noted in [14] that "[...] virtually all of the models that have been successfully analyzed in classical queueing network theory are models having a so-called product form stationary distribution."

infinitely often. Moreover, due to the nondeterminism, we now deal with *continuous-time Markov decision processes (CTMDPs)*. Our main theorem (Theorem 3) states that if there exists any scheduler resolving the nondeterminism in such a way that the controlled branching network is stable, then there exists a *randomized static* scheduler that achieves stability as well, where by "randomized static" we mean that the decisions may be randomized but not dependent on the current state (such as the load) of the system. Moreover, the existence of such a stabilizing scheduler and the scheduler itself can be determined in polynomial time, and, finally, the randomized static scheduler is stabilizing in a very strong sense, in particular, all moments of the queue sizes are finite.

*Related work.* We use nondeterminism to describe systems whose behaviour is not completely specified. A system *designer* can then resolve the nondeterminism to achieve certain goals, in our case stability. Although nondeterminism is a very well established modelling feature of probabilistic systems (see e.g. [19]), the literature on automatic design of stabilizing controllers for queueing networks is sparse. *Flow-controlled* networks [26, 21] allow to control only the external arrival stream or the service rates (see also [2] and the references therein). The authors of [18, 13] consider queueing networks with fewer servers than job types, so that the controller needs to assign servers to queues. As in [18, 13], we also use linear programming to design a controller, but our aim is different: we allow the controller to influence the production of the individual queues, and we study the complexity of designing stabilizing controllers and the nature of such controllers. There has been a substantial amount of work in the last years analyzing probabilistic systems with "branching features", most prominently on *recursive Markov chains* [12, 11] and *probabilistic pushdown systems* [10, 5]. While these models allow for a probabilistic splitting of tasks by pushing new procedures on a stack, the produced tasks are processed in a strictly sequential manner, whereas the queues in a queueing network process jobs in parallel and in continuous time. Recently, *probabilistic split-join systems* were introduced [16], which allow for branching but not for external arrivals, and assume unlimited parallelism. In [17, chapter 8] a queueing model with multiple classes of tasks and "feedback" is discussed, which is similar to our branching except that there is only one server, hence there is no parallelism. Algorithmic theory of queueing systems has also attracted some attention in the past. In particular, for *closed* (i.e., without external arrivals) queueing systems, [24] shows EXP-completeness of minimizing a weighted throughput of the queues.

## 2 Preliminaries

*Numbers.* We use $\mathbb{Z}, \mathbb{Q}, \mathbb{R}$ for the sets of integer, rational, real numbers, respectively, and $\mathbb{N}, \mathbb{Q}_{\geq 0}, \mathbb{R}_{\geq 0}$ for their respective subsets of nonnegative numbers.

*Vectors and Matrices.* Let $n \geq 1$. We use boldface letters for denoting vectors $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) \in \mathbb{R}^n$. Vectors are row vectors per default, and we use superscript $^T$ for transpose, so that $\boldsymbol{x}^T$ denotes a column vector. If the dimension $n$ is clear from the context, we write $\mathbf{0} := (0, \ldots, 0)$, $\mathbf{1} := (1, \ldots, 1)$, and $\boldsymbol{e}^{(i)} = (0, \ldots, 0, 1, 0, \ldots, 0)$ for the vector with the 1 at the $i$th component ($1 \leq i \leq n$). It is convenient to define $\boldsymbol{e}^{(0)} := \mathbf{0}$. For two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ we write $\boldsymbol{x} \sim \boldsymbol{y}$ with $\sim \in \{=, <, \leq, >, \geq\}$ if the respective relation holds componentwise. For a vector $\boldsymbol{x} \in \mathbb{R}^n$ we denote its *1-norm* by $\|\boldsymbol{x}\| := \sum_{i=1}^{n} |\boldsymbol{x}_i|$. When $\boldsymbol{x} \in \mathbb{N}^n$ is a vector of queue sizes, we refer to $\|\boldsymbol{x}\|$ as the *total queue size*. For a matrix $A \in \mathbb{R}^{n \times n}$, we write $A_i$ for its $i$th row, i.e., $A_i = (A_{i1}, \ldots, A_{in})$.

*CTMDP.* A *continuous-time Markov decision process (CTMDP)* consists of an at most countable set $S$ of states, an initial state $s_1 \in S$, a set of actions $\Sigma$ [2], and a transition rate $q(s, \sigma, s') \geq 0$ for each pair of states $s, s' \in S$ and each action $\sigma \in \Sigma$ (here $q(s, \sigma, s') = 0$ means that the transition from $s$ to $s'$ never occurs). We define a *continuous-time Markov chain (CTMC)* to be a CTMDP whose set of actions $\Sigma$ is a singleton (we usually do not write the only action explicitly, so the transition rates of a CTMC are denoted by $q(s, s')$, etc.).

Intuitively, a run of a CTMDP starts in $s_1$ and then evolves in so-called epochs. Assume that (after the previous epoch) the system is in a state $s$. The next epoch consists of the following phases: First, a scheduler chooses an action $\sigma \in \Sigma$ to be executed. Second, a waiting time for transition to each state $s' \in S$ is chosen according to the exponential distribution with the rate $q(s, \sigma, s')$ (here we assume that if $q(s, \sigma, s') = 0$, then the waiting time is $\infty$). The transitions compete in a way such that the one with the least waiting time is executed and the state of the CTMDP is chosen accordingly (the other transitions are subsequently discarded).

Formally, a *run* is an infinite sequence $s_1, \sigma_1, t_1, s_2, \sigma_2, t_2, \ldots \in (S \times \Sigma \times \mathbb{R}_{\geq 0})^\omega$. We denote by *Run* the set of all runs. A *scheduler* is a function $\Theta$ which assigns to every finite path $s_1, \sigma_1, t_1, s_2, \sigma_2, t_2, \ldots s_n \in (S \times \Sigma \times \mathbb{R}_{\geq 0})^* \times S$ a probability distribution $\Theta(w)$ on actions (i.e. $\Theta(w) : \Sigma \to [0, 1]$ satisfies $\sum_{\sigma \in \Sigma} \Theta(w)(\sigma) = 1$). For technical reasons, we have to restrict ourselves to measurable schedulers (for details see e.g. [23]).

We work with a measurable space of runs $(\mathit{Run}, \mathcal{F})$ where $\mathcal{F}$ is the smallest $\sigma$-algebra generated by basic cylinders (i.e. sets of runs with common finite prefix) in a standard way. Every scheduler $\Theta$ induces a unique probability measure $\Pr_\Theta$ on $\mathcal{F}$ determined by the probabilities of the basic cylinders. For detailed definitions see [23]. Then each scheduler $\Theta$ *induces* a stochastic process $(x(t) \mid t \in \mathbb{R}_{\geq 0})$ on the probability space $(\mathit{Run}, \mathcal{F}, \Pr_\Theta)$ where $x(t)$ is the current state of the run in time $t$, i.e., each $x(t)$ is a random variable defined by

$$x(t)(s_1, \sigma_1, t_1, s_2, \sigma_2, t_2, \ldots) = s_i \quad , \quad \sum_{j=1}^{i-1} t_i \leq t \text{ and } \sum_{j=1}^{i} t_i \geq t \,.$$

A scheduler $\Theta$ is *memoryless* if for every path $w = s_1, \sigma_1, t_1, s_2, \sigma_2, t_2, \ldots s_{n+1} \in (S \times \Sigma \times \mathbb{R}_{\geq 0})^* \times S$ we have that $\Theta(w) = \Theta(s_{n+1})$.

*Networks.* Define $\mathsf{R}^{(n,K)} := \{\boldsymbol{r} \in \mathbb{N}^n \mid \boldsymbol{r}_1 + \cdots + \boldsymbol{r}_n \leq K\}$. A *production function* for $(n, K)$ is a function $\mathit{Prob} : R \to \mathbb{Q} \cap (0, 1]$ with $R \subseteq \mathsf{R}^{(n,K)}$ such that $\sum_{\boldsymbol{r} \in R} \mathit{Prob}(\boldsymbol{r}) = 1$. A *controlled branching network with $n$ queues and branching factor $K$* consists of an arrival rate $\mu_0 \in \mathbb{Q} \cap (0, \infty)$, queue rates $\mu_i \in \mathbb{Q} \cap (0, \infty)$ for $i \in \{1, \ldots, n\}$, an arrival production function $\mathit{Prob}_0 : R_0 \to \mathbb{Q} \cap (0, 1]$ for $(n, K)$, and finite action sets $\Sigma_1, \ldots, \Sigma_n$ as follows. An action $\sigma_i \in \Sigma_i$ assigns to queue $i$ a production function $\mathit{Prob}_i(\sigma_i) : R_i(\sigma_i) \to \mathbb{Q} \cap (0, 1]$ for $(n, K)$. Define $\Sigma := \Sigma_1 \times \cdots \times \Sigma_n$. If $\sigma = (\sigma_1, \ldots, \sigma_n) \in \Sigma$, we write $R_i(\sigma)$, $\mathit{Prob}_i(\sigma)$ and $R_0(\sigma)$, $\mathit{Prob}_0(\sigma)$ to mean $R_i(\sigma_i)$, $\mathit{Prob}_i(\sigma_i)$ and $R_0$, $\mathit{Prob}_0$. Observe that the rates $\mu_i$ do not depend on actions. This simplification is without loss of generality.[3] We assume a nonzero

[2] Usually, each state has its own set of available actions. As this feature is not needed for queueing networks, we stick to the simpler version in which all actions are always available.

[3] To show that this assumption is w.l.o.g. one can employ the standard "uniformization" trick. More precisely, assume that the actions of a queue $i$ have different rates. Define $\mu_i$ to be the maximum of all rates of $\Sigma_i$ and compensate by "adding self-loops", i.e., make the actions of $\Sigma_i$ generate a new job for queue $i$ with a suitable probability. This effectively substitutes a transition with longer delay by possibly several transitions with delay $\mu_i$. As static schedulers can be easily translated between the original and the transformed system, our results remain valid.

arrival stream, i.e., there is $\boldsymbol{r} \in R_0$ with $\boldsymbol{r} \neq \mathbf{0}$. We define the *size* of a controlled network by $n + K + (\sum_{i=0}^{n} |\mu_i|) + |R_0| + |Prob_0| + \sum_{i=1}^{n} \sum_{\sigma_i \in \Sigma_i} |R_i(\sigma_i)| + |Prob_i(\sigma_i)|$, where $|\mu_i|$ etc. means the description size assuming the rationals are represented as fractions of integers in binary. A controlled branching network induces a CTMDP with state space $\mathbb{N}^n$ (the queue sizes), initial state $\mathbf{0}$, action set $\Sigma$, and transition rates

$$q(\boldsymbol{x}, \sigma, \boldsymbol{y}) = \sum_{i \in \{0,1,\ldots,n\} : i=0 \vee \boldsymbol{x}_i \neq 0} \; \sum_{\boldsymbol{r} \in R_i(\sigma) : \boldsymbol{y} = \boldsymbol{x} - \boldsymbol{e}^{(i)} + \boldsymbol{r}} \mu_i Prob_i(\sigma)(\boldsymbol{r}) \qquad \text{for } \boldsymbol{x}, \boldsymbol{y} \in \mathbb{N}^n,\ \sigma \in \Sigma.$$

Interpreting this definition, there is a "race" between external arrivals (rate $\mu_0$) and the nonempty queues (rates $\mu_i$); if the external arrivals win, new jobs are added according to $Prob_0(\sigma)$; if queue $i$ wins, one $i$-job is removed and new jobs are added according to $Prob_i(\sigma)$.

An *purely stochastic branching network* is a controlled branching network with $\Sigma = \{\sigma\}$, i.e., with a unique action for each queue. Hence, the induced CTMDP is a CTMC. In the purely stochastic case we write only $R_i$ for $R_i(\sigma)$ etc. If $Prob_i(\boldsymbol{r}) = p$ in the purely stochastic case, we use in examples the notation $i \overset{p}{\hookrightarrow} \boldsymbol{r}$, where we often write $\boldsymbol{r} \in \mathsf{R}^{(n,K)}$ as a multiset without curly brackets. For instance, if $n = 2$, we write $1 \overset{p}{\hookrightarrow} 1, 2$ and $1 \overset{p}{\hookrightarrow} 2, 2$ and $1 \overset{p}{\hookrightarrow} \varepsilon$ to mean $Prob_1(\boldsymbol{r}) = p$ with $\boldsymbol{r} = (1, 1)$ and $\boldsymbol{r} = (0, 2)$ and $\boldsymbol{r} = (0, 0)$, respectively.

Fixing a controlled network $\mathcal{N}$ and a scheduler $\Theta$ for the CTMDP induced by $\mathcal{N}$, we obtain a stochastic process $\mathcal{N}_\Theta = (\boldsymbol{x}(t) \mid t \in \mathbb{R}_{\geq 0})$, where $\boldsymbol{x}(0) = \mathbf{0} \in \mathbb{N}^n$, which evolves according to the dynamics of $\mathcal{N}$ and the scheduler $\Theta$. In the purely stochastic case we drop the subscript $\Theta$, and so we identify a network $\mathcal{N}$ with its induced stochastic process.

▶ **Example 2** (no product form)**.** Consider the purely stochastic branching network with $0 \overset{1}{\hookrightarrow} 1, 2$ and $1 \overset{1}{\hookrightarrow} \varepsilon$ and $2 \overset{1}{\hookrightarrow} \varepsilon$. If its stationary distribution $\pi$ (for a definition of stationary distribution see before Theorem 3) had product from, the queues would be "independent in steady-state", i.e., $\pi(\boldsymbol{x}_2 \geq 1 \mid \boldsymbol{x}_1 \geq 1) = \pi(\boldsymbol{x}_2 \geq 1)$, where by $\boldsymbol{x}$ we mean $\boldsymbol{x}(t)$ in steady state. However, if $\mu_0$ is much smaller than $\mu_1 = \mu_2$, then we have $\pi(\boldsymbol{x}_2 \geq 1 \mid \boldsymbol{x}_1 \geq 1) > \pi(\boldsymbol{x}_2 \geq 1)$, intuitively because $\boldsymbol{x}_1 \geq 1$ probably means that there was an arrival recently, so that $\boldsymbol{x}_2 \geq 1$ is more likely than usual. More concretely, let $\mu_0 = 1$ and $\mu_1 = \mu_2 = 3$ and consider the 2-state Markov chain obtained by assuming that each arrival leads to the state $(1, 1)$ and each completion of any job leads to the state $(0, 0)$. By computing the stationary distribution $\pi'$ of this 2-state Markov chain in the standard way, we obtain $\pi'((0, 0)) = 6/7$ and $\pi'((1, 1)) = 1/7$. Since this 2-state Markov chain "underapproximates" the CTMC induced by the network, we have $\pi(\boldsymbol{x}_1 \geq 1 \ \wedge \ \boldsymbol{x}_2 \geq 1) \geq 1/7$. On the other hand, by considering the two queues separately, the standard formula for the M/M/1 queue gives $\pi(\boldsymbol{x}_1 \geq 1) = \pi(\boldsymbol{x}_2 \geq 1) = 1/3$. Product form would imply $\pi(\boldsymbol{x}_1 \geq 1 \wedge \boldsymbol{x}_2 \geq 1) = \pi(\boldsymbol{x}_1 \geq 1) \cdot \pi(\boldsymbol{x}_2 \geq 1) = 1/9$, contradicting the inequality above.

## 3 Results

We focus on the stability of purely stochastic and controlled branching networks. Our notion of stability requires that the network is completely empty infinitely many times. Given a stochastic process $(\boldsymbol{x}(t) \mid t \in \mathbb{R}_{\geq 0})$, we say that the process is *ergodic* if the expected return time to $\mathbf{0}$ is finite. More formally, define a random variable $R$ by

$$R \quad := \quad \inf_{t>0} \{t \mid \boldsymbol{x}(t) = \mathbf{0}, \exists t' < t : \boldsymbol{x}(t') \neq \mathbf{0}\} \ .$$

Then the process is ergodic iff $\mathbb{E}[R] < \infty$. In the controlled case, we say that a scheduler $\Theta$ for $\mathcal{N}$ is *ergodic for* $\mathcal{N}$ if $\mathcal{N}_\Theta$ is ergodic. In the following we use stability and ergodicity

interchangeably. A scheduler $\Theta$ is *static* if it always chooses the same fixed distribution on actions. Note that static schedulers are memoryless. If in a stochastic process $(\boldsymbol{x}(t) \mid t \in \mathbb{R}_{\geq 0})$ the limit $\pi(\boldsymbol{x}) := \lim_{t\to\infty} \Pr(\boldsymbol{x}(t) = \boldsymbol{x})$ exists for all $\boldsymbol{x} \in \mathbb{N}^n$ and $\sum_{\boldsymbol{x}\in\mathbb{N}^n} \pi(\boldsymbol{x}) = 1$, then $\pi : \mathbb{R}_{\geq 0} \to [0, 1]$ is called the *stationary distribution*.

▶ **Theorem 3.** *Let $\mathcal{N}$ be a controlled branching network. It is decidable in polynomial time whether there exists an (arbitrary) ergodic scheduler for $\mathcal{N}$. If it exists, one can compute, in polynomial time, a* static randomized *ergodic scheduler for $\mathcal{N}$ with stationary distribution $\pi$ such that there exists an exponential moment of the total number of waiting jobs, i.e., there is $\delta > 0$ such that $\sum_{\boldsymbol{x}\in\mathbb{N}^n} \exp(\delta \|\boldsymbol{x}\|)\pi(\boldsymbol{x})$ exists.*

To prove Theorem 3 we generalize the concept of *traffic equations* (see e.g. [6]) from the theory of Jackson networks. Intuitively, the traffic equations express the fact that the inflow of jobs to a given queue must be equal to the outflow. Remarkably, the traffic equations characterize the stability of the Jackson network. More precisely, a Jackson network is stable if and only if there is a solution of the traffic equations whose components are strictly smaller than the rates of the corresponding queues (we call such a solution *deficient*).

We show how to extend the traffic equations so that they characterize the stability of controlled branching networks. For a smooth presentation, we start with *purely stochastic* branching networks and add control later on. Hence, the overall plan of the proof of Theorem 3 is as follows: Set up traffic equations for purely stochastic branching networks and show that if there is a deficient solution of these equations, then the network is stable. This result, presented in Section 3.1 (Proposition 4), is of independent interest and requires the construction of a suitable *Lyapunov function*. Then, in Section 3.2, we generalize the traffic equations to controlled branching networks and show that any ergodic scheduler determines a deficient solution (Proposition 10). This solution naturally induces a static scheduler, which, when fixed, determines an purely stochastic network with deficiently solvable traffic equations. Propositions 4 and 10 imply Theorem 3 and provide some additional results.

## 3.1 Purely stochastic branching networks

Assume that $\mathcal{N}$ is purely stochastic, i.e., there is a single action for each queue. In such a case the CTMDP induced by the network is in fact a CTMC. We associate the following quantities to a network, which will turn out to be crucial for its performance. Let $\boldsymbol{\mu} := (\mu_1, \ldots, \mu_n)$. Let $\boldsymbol{\alpha} \in \mathbb{R}^n_{\geq 0}$ be the vector with $\boldsymbol{\alpha}_i := \mu_0 \sum_{\boldsymbol{r}\in R_0} \mathit{Prob}_0(\boldsymbol{r})\boldsymbol{r}_i$; i.e., $\boldsymbol{\alpha}_i$ indicates the expected number of external arrivals at queue $i$ per time unit. Note that $\boldsymbol{\alpha} \neq \mathbf{0}$, as we assume a nonzero arrival stream. Let $A \in \mathbb{R}^{n\times n}_{\geq 0}$ be the matrix with $A_{ij} := \sum_{\boldsymbol{r}\in R_i} \mathit{Prob}_i(\boldsymbol{r})\boldsymbol{r}_j$; i.e., $A_{ij}$ indicates the expected production of $j$-jobs when queue $i$ fires. W.l.o.g. we assume that all queues are "reachable", i.e., for all queues $i$ there is $j \in \mathbb{N}$ with $(\boldsymbol{\alpha}A^j)_i \neq 0$. We define a set of *traffic equations*

$$\lambda_j \quad = \quad \alpha_j + \sum_{i=1}^{n} \lambda_i \cdot A_{ij} \quad , \quad j \in \{1, \ldots, n\}, \tag{1}$$

in matrix form:

$$\boldsymbol{\lambda} = \boldsymbol{\alpha} + \boldsymbol{\lambda}A\,. \tag{2}$$

We prove the following proposition.

▶ **Proposition 4.** *Assume that* $\boldsymbol{\lambda} \in \mathbb{R}^n_{\geq 0}$ *solves the traffic equations* (2) *and satisfies* $\boldsymbol{\lambda} < \boldsymbol{\mu}$. *Then the following conclusions hold:*

1. *The process* $\mathcal{N}$ *is ergodic, i.e., the expected return time to* $\mathbf{0}$ *is finite.*
2. *There exists a stationary distribution* $\pi$ *such that there exists an exponential moment of the total queue size, i.e., there is* $\delta > 0$ *such that* $\sum_{\boldsymbol{x} \in \mathbb{N}^n} \exp(\delta \, \|\boldsymbol{x}\|)\pi(\boldsymbol{x})$ *exists.*

The key step to the proof of Proposition 4 is to construct a so-called *Lyapunov function* with respect to which the process $\mathcal{N}$ exhibits a "negative drift". This is in fact a classical technique for showing the stability of queueing systems [22]; the difficulty lies in finding a suitable Lyapunov function. The "drift" of $\mathcal{N}$ is given by the *mean velocity vector* $\boldsymbol{\Delta}(\boldsymbol{x}) \in \mathbb{R}^n_{\geq 0}$ of $\mathcal{N}$, defined by $\boldsymbol{\Delta}(\boldsymbol{x}) := \lim_{h \to 0^+} \mathbb{E}\left[\boldsymbol{x}(t+h) - \boldsymbol{x}(t) \mid \boldsymbol{x}(t) = \boldsymbol{x}\right]/h$. The limit exists, is independent of $t$, and we have

$$\boldsymbol{\Delta}(\boldsymbol{x}) = \boldsymbol{\alpha} + \sum_{i : \boldsymbol{x}_i \neq 0} \mu_i(-\boldsymbol{e}^{(i)} + A_i)\,. \tag{3}$$

The following lemma is implicitly proved in [9, theorem 1].

▶ **Lemma 5.** *Suppose that a function* $\widetilde{V} : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ *is two times continuously differentiable,* $\widetilde{V}(\boldsymbol{x}) = 0$ *implies* $\boldsymbol{x} = \mathbf{0}$, *and that there is* $\gamma > 0$ *such that we have*

$$\boldsymbol{\Delta}(\boldsymbol{x})\big(\widetilde{V}'(\boldsymbol{x})\big)^T \leq -\gamma \qquad \text{for all } \boldsymbol{x} \neq \mathbf{0},$$

*where* $\widetilde{V}'(\boldsymbol{x})$ *denotes the gradient of* $\widetilde{V}$ *at* $\boldsymbol{x}$. *Then the conclusions of Proposition 4 holds.*

Following [9], we construct the Lyapunov function $\widetilde{V}$ in two stages: we first define a suitable *piecewise linear* function $V : \mathbb{N}^n \to \mathbb{R}_{\geq 0}$; then $V$ is *smoothed* to obtain $\widetilde{V}$. For the definition of $V$ we need the following lemma.

▶ **Lemma 6.** *The matrix series* $A^* := \sum_{i=0}^{\infty} A^i$ *converges ("exists") in* $\mathbb{R}^{n \times n}_{\geq 0}$ *and is equal to* $(I - A)^{-1}$.

Define vectors $\boldsymbol{q}^{(1)}, \ldots, \boldsymbol{q}^{(n)} \in \mathbb{R}^n_{\geq 0}$ by setting $\boldsymbol{q}^{(i)T} := \boldsymbol{a}^{(i)T} / \left\|\boldsymbol{a}^{(i)}\right\|$, where $\boldsymbol{a}^{(i)T}$ is the $i$th column of $A^*$. Observe that we have $\mathbf{1}\boldsymbol{q}^{(i)T} = 1$ for all $i$. Define the function $V : \mathbb{R}^n_{\geq 0} \to \mathbb{R}_{\geq 0}$ by $V(\boldsymbol{x}) := \max_i\{\boldsymbol{x}\boldsymbol{q}^{(i)T}\}$. We will use the following property of $V$:

▶ **Lemma 7.** *If* $\mathbf{0} \neq \boldsymbol{x} \in \mathbb{R}^n_{\geq 0}$ *and* $\boldsymbol{x}_i = 0$, *then* $\boldsymbol{x}\boldsymbol{q}^{(i)T} < V(\boldsymbol{x})$.

Lemma 7 is not obvious; in [4] we use Farkas' lemma for the proof. The following lemma describes the crucial "negative drift" property of $V$:

▶ **Lemma 8.** *There is* $\gamma > 0$ *such that we have*

$$\boldsymbol{\Delta}(\boldsymbol{x})\big(V'(\boldsymbol{x})\big)^T \leq -\gamma \qquad \text{for all } \boldsymbol{x} \neq \mathbf{0}$$

*and all subgradient vectors* $V'(\boldsymbol{x})$ *of* $V$ *at* $\boldsymbol{x}$. *More precisely, one can choose*

$$\gamma := \min_i(\mu_i - \boldsymbol{\lambda}_i)/\left\|\boldsymbol{a}^{(i)}\right\|\,,$$

*where* $\boldsymbol{a}^{(i)T}$ *is the* $i$*th column of* $A^*$.

▶ **Example 9.** Consider the network with

$$0 \overset{1}{\hookrightarrow} 1 \qquad \begin{matrix} 1 \overset{1/5}{\longrightarrow} 2, 2 \\ 1 \overset{4/5}{\longrightarrow} \varepsilon \end{matrix} \qquad \begin{matrix} 2 \overset{1/6}{\longrightarrow} 1, 2 \\ 2 \overset{5/6}{\longrightarrow} \varepsilon \end{matrix}\,,$$

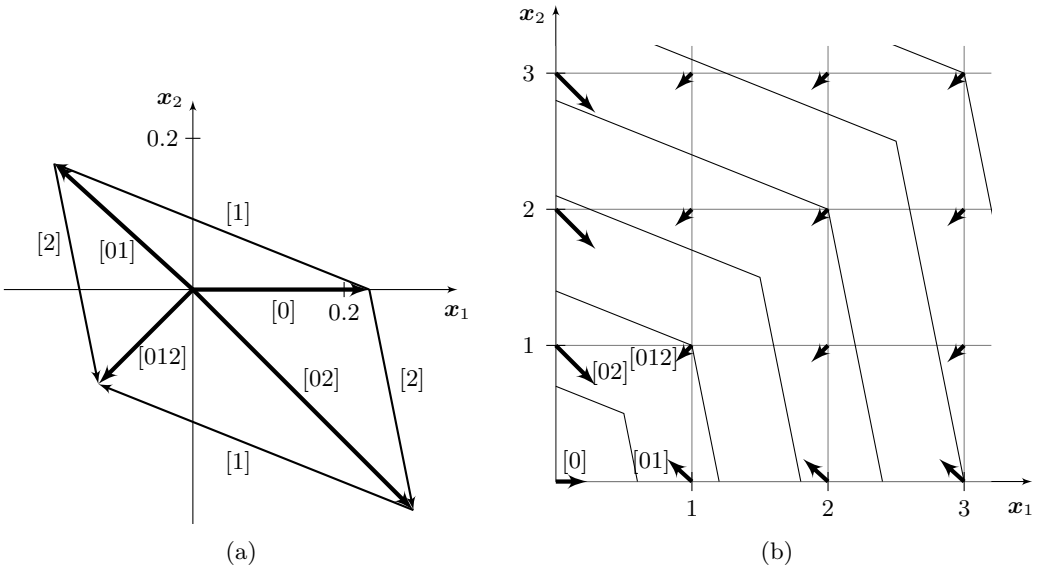


**Figure 1** Illustration of negative drift.

arrival rate $\mu_0 = 7/30$, and $\boldsymbol{\mu} = (5/12, 7/20)$, so that $\mu_0 + \mu_1 + \mu_2 = 1$. Let us write $[0] := \boldsymbol{\alpha} = \mu_0(1,0)$, $[1] := \mu_1(-\boldsymbol{e}^{(1)} + A_1)$, $[2] := \mu_2(-\boldsymbol{e}^{(2)} + A_2)$, $[01] := [0] + [1]$, $[02] := [0] + [2]$, $[012] := [0] + [1] + [2]$. These vectors are shown in Figure 1 (a). The mean velocity vector $\boldsymbol{\Delta}(\boldsymbol{x})$ is one of the vectors $[0], [01], [02], [012]$, depending on which components of $\boldsymbol{x}$ are nonzero. The vector field in Figure 1 (b) shows the corresponding vectors for several $\boldsymbol{x} \in \mathbb{N}^2$. The connected line segments indicate points $\boldsymbol{x}$ with the same value of $V(\boldsymbol{x}) = \max\{\boldsymbol{x}\boldsymbol{q}^{(1)T}, \boldsymbol{x}\boldsymbol{q}^{(2)T}\} = \max\{\frac{5}{6}\boldsymbol{x}_1 + \frac{1}{6}\boldsymbol{x}_2, \frac{2}{7}\boldsymbol{x}_1 + \frac{5}{7}\boldsymbol{x}_2\}$ (values $0.5, 1, 1.5, \ldots$). It can be seen from the figure that the drift is negative with respect to the gradient of $V$, if $\boldsymbol{x} \neq \boldsymbol{0}$.

**Proof of Lemma 8.** Let $\boldsymbol{x} \neq \boldsymbol{0}$. We need to show $\boldsymbol{\Delta}(\boldsymbol{x})\boldsymbol{q}^{(i)T} \leq -\gamma$ for all $i$ with $\boldsymbol{x}\boldsymbol{q}^{(i)T} = V(\boldsymbol{x})$. W.l.o.g. we assume that $\boldsymbol{x}\boldsymbol{q}^{(1)T} = V(\boldsymbol{x})$ and show only $\boldsymbol{\Delta}(\boldsymbol{x})\boldsymbol{q}^{(1)T} \leq -\gamma$. By Lemma 7 we have $\boldsymbol{x}_1 \neq 0$. It follows from the property $(I - A)A^* = I$ and the definition of $\boldsymbol{q}^{(1)}$ that we have

$$(-\boldsymbol{e}^{(1)} + A_1)\boldsymbol{q}^{(1)T} = -1/\left\|\boldsymbol{a}^{(1)}\right\| \qquad \text{and} \qquad (-\boldsymbol{e}^{(i)} + A_i)\boldsymbol{q}^{(1)T} = 0 \quad \text{for } 2 \leq i \leq n. \tag{4}$$

Hence we have:

$$\begin{aligned}
\boldsymbol{\Delta}(\boldsymbol{x})\boldsymbol{q}^{(1)T} &= \left(\boldsymbol{\alpha} + \sum_{i:\boldsymbol{x}_i \neq 0} \mu_i(-\boldsymbol{e}^{(i)} + A_i)\right)\boldsymbol{q}^{(1)T} && \text{by (3)} \\
&= \boldsymbol{\alpha}\boldsymbol{q}^{(1)T} - \mu_1/\left\|\boldsymbol{a}^{(1)}\right\| && \text{by (4) and } \boldsymbol{x}_1 \neq 0 \\
&\leq -\gamma + \boldsymbol{\alpha}\boldsymbol{q}^{(1)T} - \boldsymbol{\lambda}_1/\left\|\boldsymbol{a}^{(1)}\right\| && \text{by the definition of } \gamma \\
&= -\gamma + \left(\boldsymbol{\alpha} + \sum_{i=1}^{n} \boldsymbol{\lambda}_i(-\boldsymbol{e}^{(i)} + A_i)\right)\boldsymbol{q}^{(1)T} && \text{by (4)} \\
&= -\gamma + (\boldsymbol{\alpha} + \boldsymbol{\lambda}(-I + A))\,\boldsymbol{q}^{(1)T} \\
&= -\gamma + \boldsymbol{0}\boldsymbol{q}^{(1)T} = -\gamma && \text{by the traffic equation.}
\end{aligned}$$

◀

Using integration, one can obtain a two times continuously differentiable function $\widetilde{V}$ satisfying the conditions in Lemma 5: the function $V$ is smoothed by defining $\widetilde{V}(\boldsymbol{x})$, for all $\boldsymbol{x}$, as an "average" of the values $V(\boldsymbol{y})$ where $\boldsymbol{y}$ belongs to a small ball around $\boldsymbol{x}$; see the appendix of [9] for the formal details. This concludes the proof of Proposition 4.

## 3.2 Controlled branching networks

In this subsection we generalize the traffic equations (2) to deal with an arbitrary controlled branching network $\mathcal{N}$. To obtain a distribution on actions for a static randomized ergodic scheduler, we assign variables to actions instead of queues, i.e., for every action $\xi$ of the network we introduce a variable $\lambda_\xi$ capturing the *rate of firing the action* $\xi$. Denote by $\bar{\Sigma}$ the set $\bigcup_{i=1}^{n} \Sigma_i$. Given $\zeta \in \bar{\Sigma}$ and $j \in \{1, \ldots, n\}$, we denote by $A_{\zeta j}$ the average number of jobs added to the queue $j$ when the action $\zeta$ fires, i.e., for $\zeta \in \Sigma_i$ we set

$$A_{\zeta j} := \sum_{\boldsymbol{r} \in R_i(\zeta)} \mathit{Prob}_i(\zeta)(\boldsymbol{r}) \cdot \boldsymbol{r}_j \,.$$

We generalize (2) to the *traffic LP* presented in Figure 2, where the variable $\delta$ is intended to bound, for all $j$, the probability that queue $j$ is busy.

min $\delta$ subject to

$$\begin{aligned}
\sum_{\xi \in \Sigma_j} \lambda_\xi &= \boldsymbol{\alpha}_j + \sum_{i=1}^{n} \sum_{\zeta \in \Sigma_i} \lambda_\zeta \cdot A_{\zeta j} && j \in \{1, \ldots, n\} \\
\delta &\geq \frac{\sum_{\xi \in \Sigma_j} \lambda_\xi}{\mu_j} && j \in \{1, \ldots, n\} \\
\lambda_\xi &\geq 0 && \xi \in \bar{\Sigma}
\end{aligned}$$

**Figure 2** The traffic LP.

We prove the following

▶ **Proposition 10.**

1. *If there exists an arbitrary ergodic scheduler for $\mathcal{N}$, then the traffic LP can be solved with* $\min \delta < 1$.
2. *If the traffic LP is solved with* $\min \delta < 1$, *one can compute in polynomial time a static randomized ergodic scheduler $\Theta_s$ for $\mathcal{N}$. Moreover, denoting by $\rho_i$ the* utilization $\lim_{t \to \infty} \Pr(\boldsymbol{x}_i(t) \neq 0)$ *of the queue $i$, the scheduler $\Theta_s$ minimizes* $\max_i \rho_i$ *among all memoryless ergodic schedulers.*

*Hence one can decide in polynomial time whether an arbitrary ergodic scheduler exists; if yes, one can compute in polynomial time a static randomized ergodic scheduler.*

Let us first concentrate on part 1. Let $\Theta$ be an ergodic scheduler. Roughly speaking, we prove that a feasible solution of the traffic LP can be constructed using (limit) frequencies of firing individual actions in $\mathcal{N}_\Theta$. Formally, given a run $\omega$ of $\mathcal{N}_\Theta$, $t \in \mathbb{R}_{\geq 0}$ and $\xi \in \bar{\Sigma}$, we denote by $O_\xi^{\leq t}(\omega)$ the number of times the action $\xi$ is fired up to time $t$ on $\omega$. For memoryless $\Theta$ we have the following result.

▶ **Lemma 11.** *Assume that* $\Theta$ *is a* memoryless *ergodic scheduler. For every* $\xi \in \bar{\Sigma}$ *there is a constant* $O_\xi$ *such that for almost all runs* $\omega$ *of* $\mathcal{N}_\Theta$ *the limit*

$$\lim_{t\to\infty} \frac{O_{\bar{\xi}}^{\leq t}(\omega)}{t}$$

*exists and is equal to* $O_\xi$*. There is* $\bar{\delta} < 1$ *such that* $\left(\bar{\delta}, O_\xi \mid \xi \in \bar{\Sigma}\right)$ *solves the traffic LP. Moreover, for every* $i \in \{1, \ldots, n\}$ *the utilization* $\rho_i$ *of the queue* $i$ *in* $\mathcal{N}_\Theta$ *is equal to* $\frac{\sum_{\xi\in\Sigma_i} O_\xi}{\mu_i}$.

We prove Lemma 11 in [4]. If there exists an arbitrary (i.e. possibly history-dependent) ergodic scheduler, then by Theorem 7.3.8 of [25] there exists also a memoryless (and deterministic) ergodic scheduler.[4] This fact, combined with Lemma 11, implies part 1. of Proposition 10.

Now let us concentrate on part 2. of Proposition 10.

▶ **Lemma 12.** *Any feasible solution* $\left(\bar{\delta}, \bar{\lambda}_\xi \mid \xi \in \bar{\Sigma}\right)$ *of the traffic LP with* $\bar{\delta} < 1$ induces *a static randomized ergodic scheduler whose utilization of any queue* $i$ *is equal to* $\frac{\sum_{\xi\in\Sigma_i} \bar{\lambda}_\xi}{\mu_i}$.

**Proof.** We construct a static randomized scheduler $\Theta$ which chooses an action $\xi \in \Sigma_i$ for the queue $i$ with probability

$$P_\xi = \frac{\bar{\lambda}_\xi}{\sum_{\zeta\in\Sigma_i} \bar{\lambda}_\zeta} \quad , \quad \sum_{\zeta\in\Sigma_i} \bar{\lambda}_\zeta > 0\,. \tag{5}$$

Otherwise, if $\sum_{\zeta\in\Sigma_i} \bar{\lambda}_\zeta = 0$, we may control the queue $i$ arbitrarily because no jobs ever come to the queue. We further assume (w.l.o.g.) that such queues have been removed from the network, i.e., that $P_\xi$ is defined using (5) for all $\xi \in \bar{\Sigma}$. Note that $\sum_{\xi\in\Sigma_i} P_\xi = 1$ for every $i \in \{1, \ldots, n\}$.

Fixing the scheduler $\Theta$ we obtain a purely stochastic branching network whose traffic equations are deficiently solvable. Formally, we define a new purely stochastic branching network $\mathcal{N}'$ with $n$ queues with the same arrival rate, the same arrival production function and the same queue rates as $\mathcal{N}$. Further, $\mathcal{N}'$ has $R'_i = \bigcup_{\xi\in\Sigma_i} R_i(\xi)$ and the following production functions $Prob'_i$ associated to queues:

$$Prob'_i(\boldsymbol{r}) = \sum_{\xi\in\Sigma_i} P_\xi \cdot Prob_i(\xi)(\boldsymbol{r}) \quad , \quad \boldsymbol{r} \in R'_i$$

(Here we formally assume $Prob_i(\xi)(\boldsymbol{r}) = 0$ for $\boldsymbol{r} \notin R_\xi$.) The traffic equations (1) for $\mathcal{N}'$ have the following form:

$$\lambda_j \quad = \quad \alpha_j + \sum_{i=1}^{n} \lambda_i \cdot A'_{ij} \quad , \quad j \in \{1, \ldots, n\} \tag{6}$$

with

$$A'_{ij} := \sum_{\boldsymbol{r}\in R'_i} Prob'_i(\boldsymbol{r}) \cdot \boldsymbol{r}_j = \sum_{\xi\in\Sigma_i} P_\xi \sum_{\boldsymbol{r}\in R'_i} Prob_i(\xi)(\boldsymbol{r}) \cdot \boldsymbol{r}_j =$$

$$= \sum_{\xi\in\Sigma_i} \frac{\bar{\lambda}_\xi}{\sum_{\zeta\in\Sigma_i} \bar{\lambda}_\zeta} \sum_{\boldsymbol{r}\in R'_i} Prob_i(\xi)(\boldsymbol{r}) \cdot \boldsymbol{r}_j = \sum_{\xi\in\Sigma_i} \frac{\bar{\lambda}_\xi}{\sum_{\zeta\in\Sigma_i} \bar{\lambda}_\zeta} \cdot A_{\xi j}\,.$$

[4] To be formally correct, we apply Theorem 7.3.8 of [25] to the *embedded* discrete time MDP and obtain a scheduler which returns to the state $\mathbf{0}$ in finitely many steps (on average). As there are only finitely many rates in our system, this means that also the expected return time to $\mathbf{0}$ is finite.

Setting $\lambda_i := \sum_{\xi \in \Sigma_i} \bar{\lambda}_\xi$ for every $i \in \{1, \ldots, n\}$, we obtain $\lambda_i A'_{ij} = \sum_{\xi \in \Sigma_i} \bar{\lambda}_\xi A_{\xi j}$. If we put this equality into the first equation of the traffic LP, we see that $(\lambda_1, \ldots, \lambda_n)$ solves (6). Also, $\lambda_j < \mu_j$ for all $j \in \{1, \ldots, n\}$. Proposition 4 then implies that the scheduler $\Theta$ is ergodic.

Finally, let us concentrate on the utilization. Note that the utilization of any queue $i$ is the same in $\mathcal{N}'$ as in $\mathcal{N}_\Theta$, so it suffices to concentrate on $\mathcal{N}'$. Observe that the matrix $I - A'$ is invertible by Lemma 6. This means that $(\lambda_1, \ldots, \lambda_n)$ is, in fact, the *unique* solution of (6). Then however, by Lemma 11, the utilization $\rho_i$ of queue $i$ in $\mathcal{N}'$ (and thus also in $\mathcal{N}_\Theta$) is equal to $\frac{\lambda_i}{\mu_i} = \frac{\sum_{\xi \in \Sigma_i} \bar{\lambda}_\xi}{\mu_i}$. ◀

To complete the proof of Proposition 10, we consider the problem of minimizing the maximal utilization $\max_i \rho_i$. Let $\Theta_s$ be a static randomized ergodic scheduler induced by a solution of the traffic LP in the sense of Lemma 12 (here we consider a solution which minimizes $\delta$). Observe that the scheduler $\Theta_s$ minimizes $\max_i \rho_i$ among all schedulers induced by solutions of the traffic LP. However, by Lemmas 11 and 12, for every memoryless scheduler $\Theta$ there exists a static randomized scheduler induced by a solution of the traffic LP which has the same utilization of each queue as $\Theta$. Thus $\Theta_s$ minimizes $\max_i \rho_i$ among all memoryless ergodic schedulers.

## 4 Conclusions

We have suggested and studied controlled branching networks, a queueing model which extends Jackson networks by nondeterministic and branching features as required to model parallel systems. Although much of the classical theory (such as product-form stationary distributions) no longer holds for controlled branching networks, we have shown that the traffic equations can be generalized. This enabled us to construct a suitable Lyapunov function which we have used to establish strong stability properties. We have shown for the controlled model that static randomized schedulers are sufficient to achieve those strong stability properties. Linear programming can be used to efficiently compute such a scheduler, which at the same time minimizes the maximal queue utilization.

Future work should include the investigation of more performance measures, e.g., the long-time average queue size. Can non-static schedulers help to minimize it?

**Acknowledgements:** The authors thank Javier Esparza for helpful discussions, and anonymous reviewers for valuable comments.

## References

1 M. Ahmadi and S. Wong. A performance model for network processor architectures in packet processing system. In *Proceedings of the 19th IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 176–181. ACTA Press, 2007.

2 A. Azaron and S.M.T. Fatemi Ghomi. Optimal control of service rates and arrivals in Jackson networks. *European Journal of Operational Research*, 147(1):17–31, 2003.

3 G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, 2006.

4 T. Brázdil and S. Kiefer. Stabilization of branching queueing networks. Technical report, arxiv.org, 2011. Available at `http://arxiv.org/abs/1112.1041`.

5 T. Brázdil, S. Kiefer, A. Kučera, and I. Hutařová Vařeková. Runtime analysis of probabilistic programs with unbounded recursion. In *Proceedings of ICALP*, volume 6756 of *LNCS*, pages 319–331, 2011.

6 H. Chen and D. Yao. *Fundamentals of Queueing Networks*. Springer, 2001.

7 J. Daigle. *Queueing Theory with Applications to Packet Telecommunication*. Springer, 2010.

8 J.D. Deng and M.K. Purvis. Multi-core application performance optimization using a constrained tandem queueing model. *Journal of Network and Computer Applications*, In Press, Corrected Proof, 2011. DOI: 10.1016/j.jnca.2011.07.004.

9 D. Down and S.P. Meyn. Piecewise linear test functions for stability and instability of queueing networks. *Queueing Systems*, 27:205–226, 1997.

10 J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. In *LICS 2004*, pages 12–21. IEEE, 2004.

11 K. Etessami, D. Wojtczak, and M. Yannakakis. Recursive stochastic games with positive rewards. In *Proceedings of ICALP 2008*, pages 711–723, 2008.

12 K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1):1–66, 2009.

13 K.D. Glazebrook and J. Niño-Mora. A linear programming approach to stability, optimisation and performance analysis for Markovian multiclass queueing networks. *Annals of Operations Research*, 92:1–18, 1999.

14 J.M. Harrison and R.J. Williams. Brownian models of feedforward queueing networks: Quasireversibility and product form solutions. *Annals of Applied Probability*, 2(2):263–293, 1992.

15 J.R. Jackson. Networks of waiting lines. *Operations Research*, 5(4):518–521, 1957.

16 S. Kiefer and D. Wojtczak. On probabilistic parallel programs with process creation and synchronisation. In *Proceedings of TACAS*, volume 6605 of *LNCS*, pages 296–310. Springer, 2011.

17 M.Y. Kitaev and V.V.Rykov. *Controlled queueing systems*. CRC Press, 1995.

18 P.R. Kumar and S.P. Meyn. Duality and linear programs for stability and performance analysis of queueing networks and scheduling policies. *IEEE Transactions on Automatic Control*, 42(1):4–17, 1996.

19 M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *Proceedings of CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.

20 N. Madan, A. Buyuktosunoglu, P. Bose, and M. Annavaram. A case for guarded power gating for multi-core processors. In *High Performance Computer Architecture (HPCA)*, pages 291–300, 2011.

21 W.A. Massey and R. Srinivasan. A heavy traffic analysis for semi-open networks. *Performance Evaluation*, 13(1):59–66, 1991.

22 S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability*. Springer, 1993.

23 M. Neuhäußer, M. Stoelinga, and J.-P. Katoen. Delayed nondeterminism in continuous-time Markov decision processes. In *Proceedings of FoSSaCS 2009*, volume 5504 of *LNCS*, pages 364–379. Springer, 2009.

24 C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of optimal queueing network control. *Mathematics of Operations Research*, 24:293–305, 1994.

25 M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 2008.

26 S. Stidham, Jr. Optimal control of admission to a queueing system. *IEEE Transactions on Automatic Control*, 30(8):705–713, 1985.

27 S. La Torre, P. Madhusudan, and G. Parlato. Context-bounded analysis of concurrent queue systems. In *Proceedings of TACAS*, volume 4963 of *LNCS*, pages 299–314, 2008.

28 H. Zisgen, I. Meents, B.R. Wheeler, and T. Hanschke. A queueing network based system to model capacity and cycle time for semiconductor fabrication. In *Proceedings of the 40th Conference on Winter Simulation*, pages 2067–2074, 2008.