APPROXIMATE DISAGGREGATION–AGGREGATION SOLUTIONS FOR GENERAL QUEUEING NETWORKS

Johann Christoph Strelen
Rheinische Friedrich–Wilhelms–Universität Bonn
Römerstr. 164, D-53117 Bonn, Germany
E-mail: strelen@cs.uni-bonn.de

KEYWORDS

Computer Performance, Communications, Queueing, Markov-Chain, Approximation.

ABSTRACT

An approximate disaggregation–aggregation technique for finite state Markov chains is applied to queueing networks with different service disciplines, Coxian service and interarrival time distributions, blocking, and synchronization. Large Models with many states can be solved, so large that the state probabilities could neither be stored nor calculated in any computer.

Accurate approximations are obtained with different degrees of accuracy, depending on the investigated computational effort.

INTRODUCTION

Queueing networks are widely used in the stochastic modeling of computer, communication, and manufacturing systems. For the restricted class of separable queueing networks, efficient evaluation algorithms are known but if the models take into account things like different service disciplines, general service and interarrival time distributions, blocking, or synchronization, the analysis is expensive or even impossible due to computational complexity. In order to obtain performance measures like throughput of jobs, utilization and availability of system components, and response times, such general queueing networks are transformed into Markov chains which have typically a very large state space and a sparse transition probability matrix. The measures are calculated from the state probabilities of the Markov chain.

Unfortunately, the number of states grows quickly as the complexity of the model increases. This is why much research was done about the solution of large Markov chains (Stewart 1991, 1994, 1995 [1, 4, 2]). But the state space is subject to a combinatorial explosion law. Therefore exact methods will never suffice for all models, approximation techniques are needed. We propose the *disaggregation-aggregation (DA) iteration* (Strelen 1997 [6]). This is a general approximate technique for finite Markov chains with discrete time which is based on aggregation. The calculated state probabilities are in product form; we call them *disaggregation-aggregation product-form (DAΠ)* solutions. Here, neither the complete state space, nor all state probabilities, nor the transition probability matrix are handled explicitly, marginal probabilities are considered instead. Therefore Markov chains with really large state spaces can be analyzed, for example billions of states.

Time and space demands are by orders of magnitude lower than for the exact solution of large Markov chains. The results of DA iteration are usually accurate, and the more aggregates are considered the more accuracy can be obtained, if the aggregation is chosen suitably.

It is interesting to note that the steady state probabilities of separable queueing networks are equal to DAΠ solutions calculated according to suitably chosen aggregates. Here our DA method is exact.

In this paper we consider steady state solutions. In section 1 we speak about the transformation of queueing network models into Markov chain models, in section 2 DAΠ solutions are discussed, and some numerical examples are given in section 3.

1 MARKOV CHAINS FOR QUEUEING NETWORKS

Irreducible aperiodic Markov chains $(\mathbf{Z}(t),\ t \in I\!N)$, discrete and homogeneous in time, with finite state space $\mathcal{Z}^T$, $|\mathcal{Z}^T| = n \in I\!N$, are considered. They stem from stochastic models of computer, communication, or manufacturing systems. These systems consist of interacting subsystems, their *(system) components*. At every moment, each component is in one of its possible *component states*. This is why we denote the whole system states by *state tuples* $\mathbf{z} = (z_1, \ldots, z_K)$, where z_k indicates the state of component k. The states z_k of component k belong to the component state space $\mathcal{S}_k$ for all components $k = 1, \ldots, K$. Quite often we use $\mathcal{S}_k = [0 : N_k]$.

This work was supported in part by the Deutsche Forschungsgemeinschaft

Sometimes virtual system components are also usefull, for example a "component" which indicates the number of jobs in an open queueing network.

Aggregates are sets of system states. An aggregate can be defined by the state of a certain system component, for example the aggregate $\mathcal{Z}_{k,j}$ is the set of all system states where the component k is in state j, irrespective of the states of the other components:

$$\mathcal{Z}_{k,j} = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k = j\},\ j \in \mathcal{S}_k,\ k = 1, \ldots, K.$$

We call these aggregates *canonical*. For k fixed, all canonical aggregates $\mathcal{Z}_{k,0}, \ldots, \mathcal{Z}_{k,N_k}$ form a *canonical* partition $\mathcal{Z}_k$ of the state space. All the partitions $\mathcal{Z}_k,\ k = 1, \ldots, K$, constitute the *canonical aggregation*.

In general, a partition $\mathcal{Z}_c$ consists in aggregates $\mathcal{Z}_{c,j},\ j \in \mathcal{N}_c$. A class $\{\mathcal{Z}_1, \ldots, \mathcal{Z}_C\},\ C \in I\!N$, of partitions is an *aggregation*. For canonical aggregation, there are $C = K$ partitions, and $\mathcal{N}_c = [0 : N_c],\ c = 1, \ldots, C$, holds.

A partition can be defined by two or more system components $k, k', \ldots, \hat{k}$, for example the partition with the aggregates

$$\mathcal{Z}_{(k,k'),(j,j')} = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k = j,\ z_{k'} = j'\},$$
$$j \in \mathcal{S}_k,\ j' \in \mathcal{S}_{k'},\ k \neq k',\ k, k' \in [1 : K], \quad (1)$$

combines the components k and k'. If more than one component define aggregates, the partition and the aggregation are termed *fine*. The sets of components which are used to define a partition $\mathcal{Z}_c$ are denoted $\mathcal{H}_c = \{k, k', \ldots, \hat{k}\}$, for example $\mathcal{H}_c = \{c\},\ c = 1, \ldots, K$, for the canonical aggregation.

$p^{(t)}(\mathbf{z}) \triangleq \mathrm{P}\{\mathbf{Z}(t) = \mathbf{z}\},\ \mathbf{z} \in \mathcal{Z}^T$, denotes the state probabilities of the Markov chain at time t. These probabilities are collected in the n-vector $\mathbf{p}^{(t)}$. $p(\mathbf{x}, \mathbf{y}),\ \mathbf{x}, \mathbf{y} \in \mathcal{Z}^T$, are the state transition probabilities. They are the elements of the transition probability matrix $\mathbf{P}$.

Sometimes, if we are not interested in a particular time, or if steady state probabilities are considered, we omit the specification (t).

Each aggregate $\mathcal{Z}_{c,j}$ defines a *macro probability* or *aggregate probability* $\mathrm{P}\{\mathbf{Z} \in \mathcal{Z}_{c,j}\}$:

$$r_c(j) = \sum_{\mathbf{z} \in \mathcal{Z}_{c,j}} p(\mathbf{z}),\ j \in \mathcal{N}_c,\ c = 1, \ldots, C. \quad (2)$$

We collect the aggregate probabilities of a partition $\mathcal{Z}_c$ into tuples $\mathbf{r}_c$ and, in turn, these into $\mathbf{R} \triangleq (\mathbf{r}_1, \ldots, \mathbf{r}_C)$.

By (2), the *aggregation function*

$$\mathcal{A} : [0, 1]^n \to [0, 1]^{|\mathcal{N}_1|} \times \ldots \times [0, 1]^{|\mathcal{N}_C|},\ \mathbf{p} \mapsto \mathbf{R},$$

is defined. It calculates the macro probabilities for given state probabilities.

The states of the system components are described by the random variables $Z_k : \mathcal{Z}^T \to \mathcal{S}_k,\ \mathbf{z} \mapsto z_k$. The marginal distribution $\mathbf{r}_k$, calculated according to the canonical aggregation, is the probability distribution of Z_k. The state of the whole system is given by the random vector $\mathbf{Z} = (Z_1, \ldots, Z_K)$ with the probability distribution $\mathrm{P}\{\mathbf{Z} = \mathbf{z}\} = p(\mathbf{z}),\ \mathbf{z} \in \mathcal{Z}^T$.

Example. In a queueing network with blocking, Poisson arrivals, if any, and $\cdot$/M/1/ν_k/FCFS nodes, the system components are the nodes. Their states are the numbers z_k of jobs in them, $z_k \leq \nu_k,\ k = 1, \ldots, K$. Therefore, a system state is given by the $K-$tuple $\mathbf{z} = (z_1, \ldots, z_K)$. In an open queueing network, $\mathcal{Z}^T$ equals $\mathcal{S}_1 \times \ldots \times \mathcal{S}_K$ where $\mathcal{S}_k = [0 : \nu_k]$ is the component state space, and in a closed queueing network

$$\mathcal{Z}^T = \left\{\mathbf{z} \in \mathcal{S}_1 \times \ldots \times \mathcal{S}_K \mid z_1 + \ldots + z_K = N\right\}, \quad (3)$$

where $\mathcal{S}_k = [\tau_k : \nu_k],\quad \tau_k = \max\{0, N - \sum_{j \neq k} \nu_j\}$: A state $\tau_k > 0$ in node k occurs if in no other node space is left. □

We assume that the Markov chain transitions from state $\mathbf{y}$ into state $\mathbf{z},\ \mathbf{y} \neq \mathbf{z},\ \mathbf{y}, \mathbf{z} \in \mathcal{Z}^T$, can be classified as follows: Each state transition belongs to one and only one of the *transition classes*, which are triplets $(\mathcal{U}_\nu, u_\nu, \alpha_\nu),\ \nu = 1, \ldots, \bar{\nu}$. The introduction of these transition classes helps to structure the design of the Markov models, renders possible to generate the Markov chain automatically, and turns out to be very usefull for the DA technique.

A transition $\mathbf{y}$ into $\mathbf{z}$ of transition class ν may occur only if $\mathbf{y} \in \mathcal{U}_\nu$, where $\mathcal{U}_\nu \subset \mathcal{Z}^T$ denotes the *source states set* of this class. The *destination state function* $u_\nu : \mathcal{Z}^T \to \mathcal{Z}^T$ defines the new states, i.e. a transition of class ν leads from state $\mathbf{y} \in \mathcal{U}_\nu$ into state $u_\nu(\mathbf{y})$. The tuple u_ν consists of functions $u_{\nu,k}$, $u_\nu(\mathbf{y}) = \left(u_{\nu,1}(\mathbf{y}), \ldots, u_{\nu,K}(\mathbf{y})\right)$. Transitions of class ν occur with probability $\alpha_\nu(\mathbf{y})$. For DA iteration some restrictions on the transition classes will be required later on.

Example. We consider a queueing network with repetitive service blocking. It consists of $K \in I\!N$ nodes. In node $k,\ k = 1, \ldots, K$, there is a single exponential server, service rate μ_k. Buffer space is available for no more than ν_k jobs. If the model is open, jobs arrive according to a Poisson process with parameter λ_k but only if buffer space is available. When the service of a job in node k is finished the job tries to go to node l with probability $t_{k,l},\ l \in [1 : K]$. If there is no buffer space available, the job remains in node k and instantaneously receives another service (*repetitive-service with random destination policy*). In an open network, a job which is completely served in node k leaves the network with probability $t_{k,0}$. In a closed network, there are $N \in I\!N$ jobs, and we assume without loss of generality $\nu_k \leq N,\ k = 1, \ldots, K$. We will term the just explained queues *standard nodes*.

The system may change its state due to an arrival at node k from outside. The source state set of this transition class is $\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k < \nu_k\}$. That means, there is space left in node k for an arriving job. Only the state of node k changes, the other remain unchanged. Hence the destination state function is given by $u_{\nu,k}(\mathbf{z}) = z_k + 1$, identity otherwise: $u_{\nu,i}(\mathbf{z}) = z_i$ if $i \neq k$. The transition probability is $\alpha_\nu = \lambda_k \Delta$. Δ is the randomization constant: Initially, the Markov chain is continuous in time. Let $\mathbf{Q}$ denote its generator matrix. We consider the discrete time Markov chain with the transition probability matrix $\mathbf{P} = \mathbf{Q}\Delta + \mathbf{I}$ instead, where $\Delta = 0.99/\max_x |q_{x,x}|$. It has the same steady state probabilities as the continuous time Markov chain. By the factor 0.99, numerical problems are avoided, which may occur due to rounding errors.

Other state transitions take place when a job of node k leaves the network. The according transition class ν is defined as follows. The source states set is

$$\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0\}, \tag{4}$$

the destination state function $u_{\nu,k}(\mathbf{z}) = z_k - 1$, identity otherwise, and the transition probability is $\alpha_\nu = \mu_k t_{k,0} \Delta$.

The third kind of state transitions occur when a job leaves node k for node i, $i \neq k$. In node k, there must be at least one job, and in node i must be buffer space available. Therefore, the source states set is

$$\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_i < \nu_i\}. \tag{5}$$

Only the states of the nodes k and i change, $u_{\nu,k}(\mathbf{z}) = z_k - 1$, $u_{\nu,i}(\mathbf{z}) = z_i + 1$, and the transition probability is $\alpha_\nu = \mu_k t_{k,i} \Delta$. □

Applying transition classes, the state transition probabilities of a Markov chain can be expressed as follows:

$$\begin{aligned} p(\mathbf{y},\mathbf{z}) &= \sum_{\nu=1}^{\bar{\nu}} I(\mathbf{y} \in \mathcal{U}_\nu) I\Big(u_\nu(\mathbf{y}) = \mathbf{z}\Big) \alpha_\nu(\mathbf{y}), \\ & \quad \mathbf{y} \neq \mathbf{z},\ \ \mathbf{y},\mathbf{z} \in \mathcal{Z}^T, \\ p(\mathbf{y},\mathbf{y}) &= 1 - \sum_{\nu=1}^{\bar{\nu}} I(\mathbf{y} \in \mathcal{U}_\nu) \alpha_\nu(\mathbf{y}),\ \mathbf{y} \in \mathcal{Z}^T, \end{aligned} \tag{6}$$

where $I : \{\text{true, false}\} \to \{0,1\}$ denotes the indicator function.

It is not difficult to desin an algorithm which enumerates the system states by application of the source states sets and the destination state functions. Using such an algorithm and the transition probabilities (6), the Markov chain can be generated automatically.

Now we present more general queueing networks by defining suitable transition classes.

Interarrival times with Coxian distributions. Here we consider a node k of an open queueing network where arrivals occur. The interarrival times are $T_1 + \ldots + T_i$ with probability $\beta_1 \cdot \ldots \cdot \beta_{i-1}(1-\beta_i)$, $0 \leq \beta_i \leq 1$, $1 \leq i \leq \kappa$. The stages T_i are exponentially distributed with parameter λ_i, and κ is the maximum number of stages; consequently $\beta_\kappa = 0$ holds.

There is a (virtual) system component k' which indicates the actual stage of the distribution, $z_{k'} \in [1 : \kappa]$. Two transition classes are used, arrivals and steps into next stage without arrival:

$$\begin{aligned} \mathcal{U}_\nu &= \{\mathbf{z} \in \mathcal{Z}^T \mid z_k < \nu_k\}, \\ u_{\nu,k}(\mathbf{z}) &= z_k + 1, \quad u_{\nu,k'}(\mathbf{z}) = 1, \\ \alpha_\nu &= \lambda_{z_{k'}} \Delta (1 - \beta_{z_{k'}}), \end{aligned}$$

and

$$\begin{aligned} \mathcal{U}_\nu &= \{\mathbf{z} \in \mathcal{Z}^T \mid z_{k'} < \kappa\}, \\ u_{\nu,k'}(\mathbf{z}) &= z_{k'} + 1, \\ \alpha_\nu &= \lambda_{z_{k'}} \Delta \beta_{z_{k'}}. \end{aligned}$$

Coxian service time distributions or Erlang or hyperexponential distributions are modeled similarly.

Preemptive-resume service discipline. We consider queues $h, h+1, \ldots, h+V$ which are served by a single exponential server. This server serves always the non-empty queue with the lowest number, if any. The transition classes are similar to those of the standard nodes, the only difference concerns the end of a service: If a job leaves the network the source states set (4) is replaced by $\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_h = 0, \ldots, z_{k-1} = 0\}$, $h < k \leq h+V$, and if a job transfers from such a node k to node i, the source states set is $\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_i < \nu_i,\ z_h = 0, \ldots, z_{k-1} = 0\}$, $h < k \leq h+V$, instead of (5).

Head-of-the-line priorities. Again we consider queues $h, h+1, \ldots, h+V$ which are served by a single exponential server. If a service is finished, the server goes to the non-empty queue with the lowest number, if any. There is a (virtual) system component h' which indicates the actual position of the server, $z_{h'} \in [h : h+V]$. The transition classes are similar to those for standard nodes with the following modifications. The position of the server must be updated. When a job arrives at one of the considered priority nodes, say i, this is accomplished by

$$u_{\nu,h'}(\mathbf{z}) = \begin{cases} i & \text{if } z_h = 0, \ldots, z_{h+V} = 0, \\ z_{h'} & \text{otherwise,} \end{cases}$$

$h \leq i \leq h+V$. A job can leave node k, $h \leq k \leq h+V$, only if this node is actually served. Therefore for such transition classes the source states sets are $\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_{h'} = k\}$ if the job leaves the network, and $\mathcal{U}_\nu = \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_i < \nu_i,\ z_{h'} = k\}$ if the job goes to node i. Here the new position of the server is

$$u_{\nu,h'}(\mathbf{z}) = \begin{cases} k' & \text{if } h \leq k' \leq h+V,\ z_{k'} > 0, \\ & \quad z_h = 0, \ldots, z_{k'-1} = 0, \\ z_{h'} & \text{otherwise.} \end{cases}$$

Polling service discipline. Again we consider queues $h, h+1, \ldots, h+V$ which are served by a single exponential server. If a service is finished, the server looks in cyclic order for the next non-empty node, if any. There is a (virtual) system component h' which indicates the actual position of the server, $z_{h'} \in [h : h+V]$. The transition classes are similar to those for the just discussed head-of-the-line system. Only if a service is finished in node k, $h < k \leq h+V$, the server chooses the next node according to the cyclic order discipline. This is accomplished by

$$u_{\nu,h'}(\mathbf{y}) = \begin{cases} k' & \text{if } k < k' \le h+V,\ z_{k'} > 0, \\ & \quad z_{k+1} = 0, \ldots, z_{k'-1} = 0, \\ & \text{or if } h \le k' \le k,\ z_{k'} > 0, \\ & \quad z_{k+1} = 0, \ldots, z_{h+V} = 0, \\ & \quad z_h = 0, \ldots, z_{k'-1} = 0, \\ z_{h'} & \text{otherwise.} \end{cases}$$

Fork-join synchronization. We consider queues $h, h+1, \ldots, h+V$ which constitute a fork-join system: A job which arrives from outside of this system is split up into $V+1$ fellow jobs, and to each fork-join queue one of them is sent. There the fellow jobs are served independently. If such a service is finished it is looked for the other fellows. If one of them is not yet completely served the just finished fellow goes waiting. As soon as the last of them is completed they all join into a single job again. Blocking can be considered using a virtual system component h' which counts the jobs within the fork-join system, $z_{h'} \in [0 : \nu_{h'}]$, where $\nu_{h'}$ is the maximum buffer capacity.

An arrival from a node k outside may occur if there is at least one job and if in the fork-join system buffer space is left:

$$\begin{aligned} \mathcal{U}_\nu &= \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_{h'} < \nu_{h'}\}, \\ u_{\nu,h}(\mathbf{z}) &= z_h + 1, \ldots, u_{\nu,h+V}(\mathbf{z}) = z_{h+V} + 1, \\ u_{\nu,h'}(\mathbf{z}) &= z_{h'} + 1, \\ \alpha_\nu &= \mu_k t_{k,h} \Delta, \end{aligned}$$

where $t_{k,h}$ denotes the transfer probability to the fork-join system.

In the fork-join system a service in node k may end without join if there is another node with at least z_k jobs:

$$\begin{aligned} \mathcal{U}_\nu &= \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0, \\ &\qquad \exists k' \in [h : h+V] \setminus \{k\} : z_{k'} \ge z_k\}, \\ u_{\nu,k}(\mathbf{z}) &= z_k - 1, \\ \alpha_\nu &= \mu_k \Delta. \end{aligned}$$

The service ends with join if there is no such node, and if in the target node i which lies outside of the fork-join system buffer space is left for the united job:

$$\begin{aligned} \mathcal{U}_\nu &= \{\mathbf{z} \in \mathcal{Z}^T \mid z_k > 0,\ z_i < \nu_i, \\ &\qquad \forall k' \in [h : h+V] \setminus \{k\} : z_{k'} < z_k\}, \\ u_{\nu,k}(\mathbf{z}) &= z_k - 1, \quad u_{\nu,i}(\mathbf{z}) = z_i + 1, \\ u_{\nu,h'}(\mathbf{z}) &= z_{h'} - 1, \\ \alpha_\nu &= \mu_k t_{h,i} \Delta, \end{aligned}$$

where $t_{h,i}$ denotes the transfer probability to node i.

2 DISAGGREGATION–AGGREGATION ITERATION

In this section we survey the DA method, details are given in (Strelen 1997 [6]).

Our proposed solution technique needs *disaggregation*: Given positive macro probabilities $\mathbf{R}$, a distribution $\boldsymbol{\pi}$ over $\mathcal{Z}^T$ is defined as follows where $\boldsymbol{\pi}$ is a n-vector with the probabilities $\pi(\mathbf{z})$, $\mathbf{z} \in \mathcal{Z}^T$. First, the distribution satisfies the macro probabilities, $\mathbf{R} = \mathcal{A}(\boldsymbol{\pi})$, and secondly, $\boldsymbol{\pi}$ maximizes the entropy. According to the *product-form theorem*, see (Kagan et al. 1973 [3, theorem 13.2.1, page 409]), this maximum entropy distribution (MED) $\boldsymbol{\pi}$ is uniquely determined and has a product form,

$$\pi(\mathbf{z}) = \rho_{1,z_1} \cdot \ldots \cdot \rho_{C,z_C} / G, \tag{7}$$

where the $\rho_{c,z_c} > 0$ are some real numbers, and G is a normalizing constant. In fact, the MED is completely determined by all *MED parameters* $\rho_{c,j}$, $\boldsymbol{\rho}$ for short.

By (7) and (2), the *disaggregation function* is defined:

$$\mathcal{D} : [0,1]^{|\mathcal{N}_1|} \times \ldots \times [0,1]^{|\mathcal{N}_C|} \to [0,1]^n,\ \mathbf{R} \mapsto \boldsymbol{\pi}.$$

It calculates the MED over the state space for given aggregate probabilities.

Instead of calculating the steady state probabilities $\mathbf{q}$ of a Markov chain we determine an approximation $\mathbf{p} = \mathcal{D}(\mathbf{R})$ where $\mathbf{R} = \mathcal{T}(\mathbf{R})$ holds. Here the function $\mathcal{T}$ is defined as follows:

$$\begin{aligned} \mathcal{T} \quad &: \quad [0,1]^{|\mathcal{N}_1|} \times \ldots \times [0,1]^{|\mathcal{N}_C|} \to \\ &\qquad [0,1]^{|\mathcal{N}_1|} \times \ldots \times [0,1]^{|\mathcal{N}_C|}, \\ \mathbf{R} \quad &\mapsto \quad \mathcal{A}\Big(\mathcal{D}(\mathbf{R})\mathbf{P}\Big). \end{aligned}$$

This approximation $\mathbf{p}$ and its marginal probabilities $\mathbf{R}$ are referred to as *steady state disaggregation-aggregation product-form (DAΠ) solution*. In fact, only the marginal probabilities $\mathbf{R}$ and the MED parameters $\boldsymbol{\rho}$ and G must be calculated. This is practicable even if the state space is large, so large that neither all state probabilities nor the transition probabilities could be handled.

We calculate the DAΠ solutions using *disaggregation-aggregation (DA) iteration*,

$$\mathbf{R}^{(t+1)} = \mathcal{T}(\mathbf{R}^{(t)}),\ t = 1, 2, \ldots \tag{8}$$

until $\max_{c,j} |r_c^{(t+1)}(j) - r_c^{(t)}(j)|$ is sufficiently small, beginning with feasible marginal probabilities $\mathbf{R}^{(1)}$.

This DA iteration resembles vector iteration, $\mathbf{q}^{(t+1)} = \mathbf{q}^{(t)}\mathbf{P}$, $t = 1, 2, \ldots$, beginning with any distribution $\mathbf{q}^{(1)}$. $\mathbf{q}^{(t)}$ are the time-dependent state probabilities and $\mathcal{A}(\mathbf{q}^{(t)})$ the time-dependent aggregate probabilities. $\mathbf{q}^{(t)}$ converges to the steady state probability vector and $\mathcal{A}(\mathbf{q}^{(t)})$ to the steady state aggregate probabilities. Accordingly, when DA iteration $\mathbf{R}^{(t+1)} = \mathcal{T}(\mathbf{R}^{(t)})$, $t = 1, 2, \ldots$, is performed, $\mathbf{R}^{(t)}$ are approximations of the time-dependent aggregate probabilities and $\mathcal{D}(\mathbf{R}^{(t)})$ approximations of the time-dependent state probabilities. If DA iteration is done for a sufficient number of iterations, $\mathbf{R}^{(t)}$ is an approximation of the steady state aggregate probabilities, and $\mathcal{D}(\mathbf{R}^{(t)})$ an approximation of the steady state probabilities.

The function $\mathcal{T}$ is evaluated with the *aggregate transition equations*

$$r_c^{(t+1)}(l) = \sum_{j \in \mathcal{N}_c} \pi_c(j,l) r_c^{(t)}(j), \; l \in \mathcal{N}_c, \; c = 1, \ldots, C, \tag{9}$$

using the *aggregate transition probabilities*

$$\begin{aligned} \pi_c(j,l) \;&\stackrel{\triangle}{=}\; \mathrm{P}\{\mathbf{Z}_c^{(t+1)} \in \mathcal{Z}_{c,l} \mid \mathbf{Z}_c^{(t)} \in \mathcal{Z}_{c,j}\}, \\ & j, l \in \mathcal{N}_c, \; c = 1, \ldots, C. \end{aligned} \tag{10}$$

For canonical aggregation these are efficiently calculated with

$$\pi_k(j,l) = \frac{1}{r_k(j)} \sum_{\nu=1}^{\bar{\nu}} \begin{cases} \alpha_\nu \mathrm{P}\{\mathbf{Z} \in \mathcal{U}_\nu \cap \mathcal{Z}^T \cap \mathcal{X}\} \\ \quad \text{if } j \in \mathcal{U}_{\nu,k}, \; u_{\nu,k} = l, \\ 0 \text{ otherwise}, \end{cases} \tag{11}$$

where $\mathcal{X} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_{k-1} \times \{j\} \times \mathcal{S}_{k+1} \times \ldots \times \mathcal{S}_K$, and similarly if a partition $\mathcal{Z}_c$ is defined by the states of more system components.

Here we must observe some restrictions on the transition classes. Each source states set must be a cartesian product, $\mathcal{U}_\nu = \mathcal{U}_{\nu,1} \times \ldots \times \mathcal{U}_{\nu,K}$ or the union of not too many such products. Under these circumstances the probabilities in (11) can be evaluated efficiently. Furthermore if there is a partition $\mathcal{Z}_c$ which is defined by the system components $h, h', \ldots, \hat{h} \in \mathcal{H}_c$, and if the transition class ν changes the state z_k of the component k where $k \in \mathcal{H}_c$, then the transition probability $\alpha(\mathbf{z})$ and the destination state function $u_\nu(\mathbf{z})$ must not depend on component states $z_i, \; i \notin \mathcal{H}_c$.

For the calculation of the probabilities in (11), the MED parameters $\boldsymbol{\rho}$ are needed. In general, the nonlinear equations (2) must be solved in order to calculate all MED parameters $\rho_{k,j}$ of the MED $\boldsymbol{\pi}$, given the marginal probabilities $\mathbf{R}$. This is easily accomplished in some special cases. For example, if the aggregation is canonical and if the state space is *rectangular*, i.e. if $\mathcal{Z}^T$ is a cartesian product, $\mathcal{Z}^T = \mathcal{S}_1 \times \ldots \times \mathcal{S}_K$, the MED is given by $\pi(\mathbf{z}) = r_1(z_1) \cdot \ldots \cdot r_K(z_K), \; \mathbf{z} \in \mathcal{Z}^T$. If such a simple disaggregation is not available, an approximation is sufficient which becomes more and more accurate during the DA iteration:

$$\rho_{c,j}^{(t+1)} = 0.5 \left(1 + r_c^{(t+1)}(j)/r_c^{(t)}(j)\right) \rho_{c,j}^{(t)}.$$

Unfortunately, we have not yet found error estimates or a proof of convergence of the iteration. But in (Strelen 1997 [6]) the existence of DAΠ solutions is proven, and it is proven that there is always an aggregation with only a small number of aggregates for which an approximating product form (7) exists which is as accurate as desired. Accuracy is verfied empirically: We solved many models using DA iteration, among others polling, fork join, stochastic Petri net, and blocking models, and we found very accurate results; the iteration did never diverge. Moreover, the accuracy could be substantially ameliorated in all examples if more and suitably chosen aggregates are considered. These matters are discussed in (Strelen 1997 [6]).

3 NUMERICAL EXAMPLES

Here we consider the steady state probabilities of the component states. Other performance measures are calculated from them. Therefore their accuracy is of particular interest. In the examples with different aggregations the reader may note the gain in accuracy when finer aggregation with more aggregates is used.

Example: Open network with repetitive service blocking. An open tandem network with repetitive service blocking is considered: $K = 4$ nodes, capacity $\nu = 3$ in all nodes, arrivals at node 1, $\lambda_1 = 1$, service rates $\mu_1 = \mu_2 = \mu_4 = 1, \; \mu_3 = 1, 2, \ldots, 20$, transfer probabilities $t_{1,2} = t_{2,3} = t_{3,4} = t_{4,0} = 1$.

The maximum absolute error of the stationary node state probabilities occurs when $\mu_3 = 20$. It is 0.13... using canonical aggregation where the states of any node define the aggregates of an according partition. If an aggregation with partitions according to $\mathcal{H}_1 = \{1,2\}, \; \mathcal{H}_2 = \{2,3\}, \; \mathcal{H}_3 = \{3,4\}$ is used where each pair of connected nodes defines a partition, the maximum absolute error is 0.022... . If triplets of connected nodes are taken instead, i.e. the aggregation is according to $\mathcal{H}_1 = \{1,2,3\}, \; \mathcal{H}_2 = \{2,3,4\}$, the error is 0.0023... . □

The next example demonstrates that Markov models with a really large state space can be solved with the DA iteration technique.

Example: A large model. The tandem topology was analyzed, but here with 9 nodes, $\mu_3 = 1$, and the capacity $\nu = 9$ in all nodes, which has 10^9 system states. On an Atari 1040ST, the computing time for DA iteration with canonical aggregation was 238 seconds. □

Example: Queueing network with preemptive resume service discipline. An open tandem network with 4 nodes and repetitive service blocking is considered. The service discipline is preemptive-resume with two priority classes, the capacity is $\nu = 2$ in all nodes for each priority class, Poisson arrivals occur at node 1 with rate 0.5 for each of the two priority classes, and all service times are exponentially distributed with rate 2. The jobs travel from node 1 to node 2, from node 2 to node 3, and from node 3 to node 4.

System component 1 models the waiting room of priority-0 jobs in node 1, system component 2 the waiting room of priority-1 jobs in node 1, ... , system component 8 the waiting room of priority-1 jobs in node 4.

Different aggregations were tried. First the states of each node which consists of two waiting rooms define a partition, i.e. $\mathcal{H}_1 = \{1,2\}, \; \mathcal{H}_2 = \{3,4\}, \; \mathcal{H}_3 = \{5,6\}, \; \mathcal{H}_4 = \{7,8\}$. The maximum error of the steady state probabilities of job numbers in waiting rooms is 0.03... .

If the joint states of all pairs of connected nodes are taken for the definition of the aggregation, i.e. $\mathcal{H}_1 = \{1,2,3,4\}, \; \mathcal{H}_2 = \{3,4,5,6\}, \; \mathcal{H}_3 = \{5,6,7,8\}$, this maximum error is 0.01... .

If the joint states of all triplets of connected nodes are taken for the definition of the aggregation, i.e. $\mathcal{H}_1 = \{1,2,3,4,5,6\}$, $\mathcal{H}_2 = \{3,4,5,6,7,8\}$, this maximum error is 0.002... . □

Example: A loss network with Erlang-2 distributed service times. An open tandem network with 5 nodes is considered, the capacity is $\nu = 2$ in all nodes, Poisson arrivals occur at node 1 with rate 1, and all service times are Erlang-2 distributed with mean service time 0.5. The jobs travel from node 1 to node 2, then to node 3, node 4, and finally to node 5, if they are not lossed before due to full storage in a target node.

First the states of each node with the waiting room and the system component which describes its service process are used to define a partition. The maximum error of the steady state probabilities of job numbers in waiting rooms or of a service process is 0.06... .

If the joint states of all pairs of connected nodes are used for the definition of the aggregation, this maximum error is 0.007... .

If the joint states of all triplets of connected nodes are used for the definition of the aggregation, this maximum error is 0.00008... . □

Example: A loss network with bursty arrivals. An open tandem network with 8 nodes is considered. The capacity is $\nu = 2$ in all nodes. All service times are exponentially distributed with rate 2. The arrival process is defined by a hypergeometric distribution with mean arrival rate 1 and squared coefficient of variation 80.8... . The arrivals occur in bursts, the mean number of jobs per burst is 10, and an epoch without arrivals is ten times as long as an epoch with arrivals, in mean.

First the states of each node are used to define a partition. Only in node 1 the system component which describes the arrival process is accounted for. The maximum error of the steady state probabilities of job numbers in waiting rooms or of the arrival process is 0.048... .

If the joint states of all pairs of connected nodes are taken for the definition of the aggregation, this maximum error is 0.019... .

If the joint states of all triplets of connected nodes are taken for the definition of the aggregation, this maximum error is 0.009... . □

More numerical results of queueing networks with blocking can be found in (Strelen et al. 1997 [7]). Numerical results for fork-join systems and for polling systems are presented in (Strelen 1991 [5]).

CONCLUSION

Product form approximations for the steady state probabilities of a quite general class of queueing networks are proposed, which are suitable for very large state spaces. Accuracy was empirically verified for many models, among others priority, polling, fork join, and blocking models. By choosing aggregations with more aggregates – fine grain aggregation – the accuracy can be substantially ameliorated.

Further research concerns applications of the DA method on other Markov models, stiff Markov chains, and a tool.

References

[1] W. J. Stewart (ed.). *Numerical Solution of Markov Chains.* Marcel Dekker, New York, Basel, Hong Kong, 1991.

[2] W. J. Stewart (ed.). *Computation with Markov Chains.* Kluwer, 1995.

[3] A. M. Kagan, J. V. Linnik, and C. R. Rao. *Characterization Problems in Mathematical Statistics.* Wiley, New York, 1973.

[4] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Priceton University Press, 1994.

[5] J. Ch. Strelen. Iterative Analyse von Markov-Modellen mit alternierender Aggregation und Disaggregation. In A. Lehmann and F. Lehmann, editors, *Messung, Modellierung und Bewertung von Rechensystemen IFB 286*, pages 320–336. Springer, Berlin, 1991.

[6] J. Ch. Strelen. Approximate product form solutions for Markov chains. *Accepted for publication in Performance Evaluation*, 1997.

[7] J. Ch. Strelen, B. Bärk, J. Becker, and V. Jonas. Analysis of queueing networks with blocking using a new aggregation technique. *Accepted for publication in Annals of Operations Research*, 1997.

BIOGRAPHY

Johann Christoph Strelen was born in Wiesbaden, Germany, in 1941. He received the Dipl.-Mat. and Dr.rer.nat. degrees in mathematics and the Habilitation degree in Computer Science from the Technische Hochschule Darmstadt, Germany, in 1968, 1973, and 1981, respectively. There he was affiliated to the Computing Center (1968-1973), and assistant at the Computer Science Department (1974-1982). In 1973 he was a post doctoral fellow at the IBM Scientific Center, Grenoble, France. Since 1982 he has been a Professor of Computer Science at the Rheinische Friedrich–Wilhelms–Universität Bonn, Germany. His reseach interests include performance evaluation, distributed systems, and simulation. Dr. Strelen is a member of the Gesellschaft für Informatik (GI) and the Gesellschaft für angewandte Mathematik und Mechanik (GAMM).