Structured Markov chains solver: tool extension

D. A. Bini, B. Meini, S. Steffé
Dipartimento di Matematica
Università di Pisa, Pisa, Italy
bini, meini, steffe @dm.unipi.it

B. Van Houdt
Department of Mathematics and Computer Science
University of Antwerp, Antwerpen, Belgium
benny.vanhoudt@ua.ac.be

ABSTRACT

We expand and update the software tool SMCSolver, presented at the SMCTools workshop in 2006, for the numerical solution of structured Markov chains encountered in queuing models. In particular the new version of the package implements different transformation techniques and different shift strategies which are combined in order to speed up and optimize the solution of structured Markov chains. Numerical experiments show the effectiveness of the new implemented techniques.

1. INTRODUCTION

In the papers [5], [6] the authors have collected the most efficient algorithms for solving structured Markov chains encountered in queueing models, and implemented it in the software package SMCSolver. This software, in the form of a Matlab toolbox and in a Fortran 90 package with a graphical user interface, solves M/G/1 and GI/M/1-type Markov chains, QBD processes and more. For the analysis and the algorithmic treatment of this kind of problems we refer to the books [2], [10], [11], [12]. The spirit of the two papers [5], [6] was to provide a powerful tool for the practitioners who have difficulties to get through in very advanced and complex algorithmic techniques, and for the theoretical researchers who wish to have an easy tool for carrying out numerical experiments, in order to design new algorithmic advances.

This software tool has shown to be very effective in many practical situations. However, despite the sophisticated techniques and the advanced algorithms, there are situations in which some unexpected slowdown is encountered. This occurs especially when the techniques of evaluation-interpolation are applied with the algorithm of cyclic reduction for transient M/G/1-type Markov chains. In fact, in these problems the number of interpolation points grows excessively large and deteriorates the overall performance of the algorithm.

In our present contribution, we describe and implement some recent techniques which enable one to improve the efficiency of our package. More specifically, we implement the Bright transformation introduced in [7] and analyzed in [14], which maps a transient M/G/1-type Markov chain into a positive recurrent GI/M/1-type Markov chain, and conversely. In this way, the user may apply the Bright transformation or the Ramaswami transformation, both implemented in the package, or combine them. According to the positive recurrence properties or the kind of Markov chain, one can choose and combine these transformations in order to arrive at a more easily solvable problem.

The shift technique, introduced in the package SMCSolver in order to accelerate convergence, has been expanded with two new types of shifting, providing extra options for the user which improve the performance of our software.

The paper is organized as follows. In Section 2 we recall the basic definitions; in Section 3 we describe the Bright and the Ramaswami transformation, while the shift techniques are reported in Section 4. Section 5 discusses the acceleration of algorithms obtained by combining the new features of our tools. Implementation details are reported in Section 6 together with numerical results.

SMCTOOLS 2009, October 19, 2009 - Pisa, Italy

2. STRUCTURED MARKOV CHAINS

We recall that M/G/1-type Markov chains are defined by the transition matrix

$$P = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 & \dots \\ A_{-1} & A_0 & A_1 & A_2 & \dots \\ & A_{-1} & A_0 & A_1 & \ddots \\ & & A_{-1} & A_0 & \ddots \\ 0 & & & \ddots & \ddots \end{bmatrix},$$

where A_i, for $i \geq -1$, and B_i, for $i \geq 0$, are nonnegative matrices in $\mathbb{R}^{m\times m}$ such that $\sum_{i=-1}^{+\infty} A_i$, $\sum_{i=0}^{+\infty} B_i$, are stochastic. Throughout we assume $A = \sum_{i=-1}^{+\infty} A_i$ irreducible. The drift of an M/G/1-type Markov chain is defined by $\mu = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{a}$, where $\boldsymbol{\alpha}$ is the stationary probability vector of A and $\boldsymbol{a} = \sum_{i=-1}^{+\infty} iA_i\boldsymbol{e}$, $\boldsymbol{e} = (1, 1, \dots, 1)^T$. We recall that a Markov chain is recurrent iff $\mu \leq 0$, positive recurrent iff $\mu < 0$ and $\boldsymbol{b} = \sum_{i=1}^{+\infty} iB_i\boldsymbol{e} < \infty$, transient iff $\mu > 0$, null recurrent iff either $\mu = 0$ or $\mu < 0$ and $\sum_{i=-1}^{+\infty} iB_i\boldsymbol{e} = \infty$.

Our package computes the minimal nonnegative solution G of the matrix equation

$$G = \sum_{i=-1}^{+\infty} A_i G^{i+1}. \tag{1}$$

If the Markov chain is positive recurrent, G allows one to compute the invariant probability vector $\boldsymbol{\pi}$ of the chain.

A GI/M/1-type Markov chains is defined by the transition matrix

$$P = \begin{bmatrix} B_0 & A_1 & & & 0 \\ B_{-1} & A_0 & A_1 & & \\ B_{-2} & A_{-1} & A_0 & A_1 & \\ B_{-3} & A_{-2} & A_{-1} & A_0 & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots \end{bmatrix},$$

where A_{-i}, $i \geq -1$, and B_{-i}, $i \geq 0$ are nonnegative matrices in $\mathbb{R}^{m\times m}$ such that $\sum_{i=-1}^{n-1} A_{-i} + B_{-n}$ is stochastic for all $n \geq 0$.

If $A = \sum_{i=-1}^{+\infty} A_{-i}$ is not stochastic, then the Markov chain is positive recurrent. If A is stochastic then the Markov chain is positive recurrent if $\delta < 0$, null recurrent if $\delta = 0$, and transient if $\delta > 0$, where $\delta = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{a}$, $\boldsymbol{\alpha}$ is such that $\boldsymbol{\alpha}^{\mathrm{T}}A = \boldsymbol{\alpha}^{\mathrm{T}}$, $\boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{e} = 1$, and $\boldsymbol{a} = \sum_{i=-1}^{+\infty} iA_{-i}\boldsymbol{e}$.

Our package computes the minimal nonnegative solution R of the matrix equation

$$R = \sum_{i=-1}^{+\infty} R^{i+1}A_{-i}. \tag{2}$$

If the Markov chain is positive recurrent, then the knowledge of R allows one to compute the invariant probability vector of the chain.

3. TRANSFORMATIONS

Here we recall the Ramaswami and the Bright transformations between M/G/1-type and GI/M/1-type, positive recurrent and transient Markov chains. Assume that we are given a GI/M/1-type Markov chain where the matrix $A = \sum_{i=-1}^{+\infty} A_{-i}$ is irreducible and stochastic. The connection between this GI/M/1-type Markov chain and its duals is extremely simple [13], [7], [14], [4].

3.1 Ramaswami dual

Define $D = \mathrm{diag}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha}$ is the strictly positive stationary probability vector of A, and set

$$\widetilde{A}_i = D^{-1}A_{-i}^{\mathrm{T}}D, \quad i = -1, 0, 1, \ldots.$$

The Ramaswami dual [13] of the original GI/M/1-type Markov chain is the M/G/1-type Markov chain, with homogeneous part defined by the blocks $\widetilde{A}_i$. Now, the minimal nonnegative solution R of (2) is

$$R = D^{-1}\widetilde{G}^T D,$$

where $\widetilde{G}$ is the minimal nonnegative solution of

$$X = \sum_{i=-1}^{+\infty} \widetilde{A}_i X^{i+1}. \tag{3}$$

A consequence of this property is that, to determine R for a positive recurrent GI/M/1-type Markov chain is equivalent to determining G for a transient M/G/1-type Markov chain. Conversely, to determine R for a null recurrent or transient GI/M/1-type Markov chain is equivalent to determining G for a recurrent M/G/1-type Markov chain.

3.2 Bright dual

The Bright dual [7] of the GI/M/1-type Markov is an M/G/1-type Markov chain with homogeneous part characterized by the matrices

$$\widetilde{A}_i = \tau^{i+1}\Delta^{-1}A_{-i}^T\Delta, \quad i = -1, 0, 1, \ldots.$$

Here $\Delta = \mathrm{diag}(\boldsymbol{w})$, where $\boldsymbol{w}$ is the positive stochastic left eigenvector of $A(\tau)$ corresponding to the eigenvalue τ, and $A(z) = \sum_{i=-1}^{+\infty} z^{i+1}A_i$. The scalar τ depends on whether the GI/M/1-type Markov chain is positive recurrent or transient. In the positive recurrent case we have $\tau = \rho(R) < 1$ where $\rho(R)$ is the spectral radius of R, while in the transient case $\tau > 1$ is the smallest zero of $\det(zI - A(z))$ outside the unit circle. In both cases, τ can be obtained using a simple bisection algorithm, without the need to determine R first.

The G matrix of the Bright dual satisfies the following relation with the R matrix of the original GI/M/1-type Markov chain:

$$R = \tau\Delta^{-1}\widetilde{G}^T\Delta.$$

This relation enables us to obtain R from the G matrix of its dual. Therefore, determining R for a positive recurrent GI/M/1-type Markov chain is equivalent to determining G for a positive recurrent M/G/1-type Markov chain. Conversely, determining R for transient GI/M/1-type Markov chain is equivalent to determining G for a transient M/G/1-type Markov chain.

4. SHIFT TECHNIQUES FOR M/G/1-TYPE MARKOV CHAINS

For the sake of simplicity, assume that $A_i = 0$ for $i > M$. The basic shift, already implemented in [6], consists in setting

$$\begin{aligned} &\widetilde{A}_{-1} = A_{-1}(I - Q), \\ &\widetilde{A}_i = A_i - (\textstyle\sum_{j=-1}^{i} A_j - I)Q, \quad 0 \leq i \leq M, \end{aligned}$$

where $Q = \boldsymbol{e}\boldsymbol{u}^T$ and $\boldsymbol{u}$ is any vector such that $\boldsymbol{e}^T\boldsymbol{u} = 1$, if the Markov chain is recurrent. If the Markov chain is transient, it consists in setting

$$\begin{aligned} &\widetilde{A}_{-1} = A_{-1} \\ &\widetilde{A}_0 = A_0 + EA_{-1} \\ &\widetilde{A}_i = A_i - E(I - \textstyle\sum_{j=-1}^{i-1} A_j), \quad 1 \leq i \leq M, \end{aligned}$$

where $E = \boldsymbol{u}\boldsymbol{v}^T$, with $\boldsymbol{u}$ being any nonzero vector, and $\boldsymbol{v}$ such that $\boldsymbol{v}^T(\sum_{i=-1}^{M} A_i) = \boldsymbol{v}^T$, $\boldsymbol{v}^T\boldsymbol{u} = 1$.

Moreover, let us introduce the new equation

$$X = \sum_{i=-1}^{M} \widetilde{A}_i X^{i+1}. \tag{4}$$

It can be proved (see [2]) that the solution $\widetilde{G}$ of smallest spectral radius of (4) is

$$\begin{aligned} &\widetilde{G} = G - Q \quad \text{if } \mu \leq 0, \\ &\widetilde{G} = G \qquad\quad \text{if } \mu > 0. \end{aligned}$$

It has been proved that the roots of the polynomials $\widetilde{a}(z) = \det(zI - \sum_{i=-1}^{M} z^{i+1}\widetilde{A}_i)$, and $a(z) = \det(zI - \sum_{i=-1}^{M} z^{i+1}A_i)$, are the same except for the root $z = 1$ of $a(z)$ which is shifted to zero or to the infinity for $\widetilde{a}(z)$ according to the recurrent or transient nature of the chain under consideration.

This tiny difference on the roots of $a(z)$ and $\widetilde{a}(z)$ makes a great difference in the convergence speed of the algorithms for the solution of matrix equations if applied to (1) or to (4). In fact, iterative methods converge faster if applied to equation (4). More precisely, denoting by λ_i, $i = 1, 2, \ldots$, the roots of $a(z)$ ordered such that $|\lambda_i| \leq |\lambda_{i+1}|$, and $\xi = |\lambda_m|$, $\eta = |\lambda_{m+1}|$, one finds that $\xi \leq 1 \leq \eta$. Furthermore, if $\mu = 0$ then $\xi = \eta = 1$, if $\mu < 0$ then $\xi = 1 < \eta$ and if $\mu > 0$ then $\xi < 1 < \eta = 1$.

The convergence speed of certain algorithms like cyclic reduction (CR) and logarithmic reduction (LR) or fixed point iterations, depends on the ratio ξ/η. The smaller is this ratio the faster is the convergence.

In the positive recurrent case, the roots $\widetilde{\lambda}_i$ of $\widetilde{a}(z)$ are given by $\widetilde{\lambda}_i = \lambda_{i-1}$, for $i = 2, \ldots, m+1$, $\widetilde{\lambda}_1 = 0$, $\widetilde{\lambda}_i = \lambda_i$, for $i > m+1$. Therefore the ratio $\widetilde{\xi}/\widetilde{\eta}$ turns out to be $|\lambda_{m-1}|/\eta < \xi/\eta$.

The most effective case is the null-recurrent case where $\xi = \eta = 1$. After shifting the problem one finds that the ratio $\widetilde{\xi}/\widetilde{\eta} < 1$.

The basic shift enables one to move the root 1 either to zero or to infinity according to the recurrent or transient nature of the chain. In the new version of SMCSolver, we introduce a more general shift which consists in moving either ξ to zero or η to infinity or both. This may lead to a further speed up of the algorithm.

For more details on this shift technique we refer the reader to [8], [2], [1], [14].

5. ACCELERATION OF ALGORITHMS

Dual transformations and the shift techniques together with their combinations, enable one to improve the efficiency of the known algorithms implemented in SMCSolver. In particular, as pointed out in [14], the cyclic reduction algorithm implemented with the strategy of evaluation-interpolation at the roots of unity may dramatically reduce the computational cost due to the very low number of knots that are needed to approximate the solution. Here we mostly report the comments of [14] and complement them with further remarks.

The algorithm CR applied to M/G/1 Markov chains generates a sequence of matrix power series which can be approximated with high precision by means of the evaluation-interpolation technique. More details can be found in [2], [3].

Given a matrix power series $F(z) = \sum_{i=0}^{+\infty} z^i F_i$ define the matrix power series

$$F_{\rm even}(z) = \frac{1}{2}(F(\sqrt{z}) + F(-\sqrt{z})) = \sum_{i=0}^{+\infty} z^i F_{2i},$$

$$F_{\rm odd}(z) = \frac{1}{2\sqrt{z}}(F(\sqrt{z}) - F(-\sqrt{z})) = \sum_{i=0}^{+\infty} z^i F_{2i+1}.$$

Let $A^{(0)}(z) = \sum_{i=-1}^{+\infty} z^{i+1} A_i$, $\widehat{A}^{(0)}(z) = \sum_{i=0}^{+\infty} z^i A_i$. The algorithm CR generates the sequences

$$\begin{aligned} K^{(n)}(z) &= (I - A^{(n)}_{\rm odd}(z))^{-1}, \\ A^{(n+1)}(z) &= zA^{(n)}_{\rm odd}(z) + A^{(n)}_{\rm even}(z)K^{(n)}(z)A^{(n)}_{\rm even}(z), \\ \widehat{A}^{(n+1)}(z) &= \widehat{A}^{(n)}_{\rm even}(z) + \widehat{A}^{(n)}_{\rm odd}(z)K^{(n)}(z)A^{(n)}_{\rm even}(z). \end{aligned} \quad (5)$$

It is possible to show that the sequence

$$G_n = (I - \widehat{A}^{(n)}(0))^{-1} A_{-1} \quad (6)$$

converges to G. Convergence is quadratic if $\mu \neq 0$, is linear if $\mu = 0$.

The computation of the coefficients of the matrix power series $A^{(n+1)}(z)$ and $\widehat{A}^{(n+1)}(z)$ is performed by means of the evaluation-interpolation technique. More specifically, the matrix power series $A^{(n)}(z)$ and $\widehat{A}^{(n)}(z)$ are computed at the N-th roots of 1 for sufficiently large N, then (5) are computed point-wise and the values of $A^{(n+1)}(z)$ and $\widehat{A}^{(n+1)}(z)$ at the N-th roots of 1 are obtained. Interpolation to these values is used for computing the coefficients of $A^{(n+1)}(z)$ and $\widehat{A}^{(n+1)}(z)$. In this process, the number N must be sufficiently large. If $A^{(n+1)}(z)$ were a polynomial of degree K then $N \geq K+1$ would be enough. If $A^{(n+1)}(z)$ is a power series with negligible coefficients for degree greater than K, then it can be approximated by a polynomial. Therefore, in the general case the decay properties of the coefficients of $A^{(n)}(z)$ and $\widehat{A}^{(n)}(z)$ play a fundamental role.

A nice feature of this algorithm is that for positive recurrent Markov chains the matrix power series $A^{(n)}(z)$ quadratically converges to a matrix polynomial $A_0^* + zA_1^*$. This implies that in practice, the power series $A^{(n)}(z)$ are well approximated by polynomials of low degree. A similar property holds for $\widehat{A}^{(n)}(z)$. In this way, the number of interpolation points needed during the iteration can be kept small.

On the other hand, for transient Markov chains the limit of $A^{(n)}(z)$ is a matrix power series and not a polynomial. This fact implies that a large number of interpolation points may be required by the iteration with a consequent slow down of the algorithm even though convergence still remains quadratic.

In the case we are given a transient M/G/1-type Markov chain we can transform it into a positive recurrent M/G/1-type Markov chain by the Bright dual. This fact enables us to keep low the number of interpolation points for transient Markov chains.

For a GI/M/1-type Markov chain, if the chain is transient then the Ramaswami dual is a positive recurrent M/G/1-type Markov chain, for which CR works fine. If the chain is positive recurrent, then the Ramaswami dual is transient and we may apply the Bright dual to the latter M/G/1-type transient Markov chain. In this way we get a positive recurrent M/G/1-type Markov chain.

Further acceleration of the convergence can be obtained by applying the shift technique of ξ and η. In fact, cyclic reduction can be applied to the shifted equation (4). It is interesting to point out that if the shifted problem is such that the function $a(z)$ has no more zeros of modulus 1 then the sequence $A^{(n)}(z)$ generated by CR still converges to a polynomial of degree 1. In this way the number of interpolation points is kept small. Moreover, the convergence occurs with a higher speed.

6. IMPLEMENTATION AND NUMERICAL EXPERIMENTS

The transformations and the shift techniques introduced in the previous sections have been implemented in the package SMCSolver both in the Matlab version and in the Fortran 90 version.

In the first edition of this tool, the R matrix of a GI/M/1-type Markov chain was computed by computing the G matrix of its Ramaswami dual, from which R can be obtained easily. In the current Matlab version of the tool, the user can select one of two duals: the Ramaswami or Bright dual, by setting the "Dual" parameter to "R" or "B", respectively. Setting the "Dual" parameter to "A" (Automatic) causes the software to select the Bright dual for positive recurrent chains and the Ramaswami dual for transient ones as this choice is typically the most efficient (for null recurrent chains, both duals are identical). The shift techniques have been implemented in the following way. A "ShiftType" option has been introduced. The option may be selected "one", "tau" and "dbl"; with the "one" option the root equal to 1 is shifted to zero, or infinity, according to the recurrent/transient nature of the chain; with the "tau" option the root $\eta \geq 1$ is shifted to infinity in the recurrent case, and the root $\xi < 1$ is shifted to zero in the transient case; the "dbl" option combines both the "one" and the "tau" shift.

In the Fortran 90 package we have inserted a new button "Preprocessing" in the main window of the GUI. By clicking on this button the user can select the kind of shift ("No Shift", "Simple Shift", "Tau Shift" or "Double Shift"), and the kind of dual transformation ("Auto Dual", "Ramaswami Dual", "Bright Dual", "Both Duals") with the same behavior as in the Matlab version. These new features are shown in Figure 2.

The numerical experiments reported in this section are based on the GI/M/1-type Markov chain discussed in [9] to compute the waiting time and queue length distribution in a discrete-time multitype queueing system with semi-Markovian arrivals, phase-type services and correlated customer types, i.e., the SM[K]/PH[K]/1 queue. A discrete-time Markov chain of the GI/M/1-type is constructed by observing the system only when the queue is nonempty. The level of the chain corresponds to the age of the customer c in service, i.e., the difference between the current time instant t and the arrival time t_a of the customer in service. Whereas the m states at each level keep track of the type of the customer in service, the phase of its service and the state of the semi-Markovian arrival process at time t_a. If the same customer c remains in service, its age increases by one, meaning the level goes up by one, while if customer c completes service, a new customer c' enters the server, the age of which is smaller than the age of c, meaning the level can decrease by many, but cannot increase, resulting in a GI/M/1-type Markov chain. For more details on this chain and on how to compute the waiting time and queue length distribution from its steady state, we refer to [9].

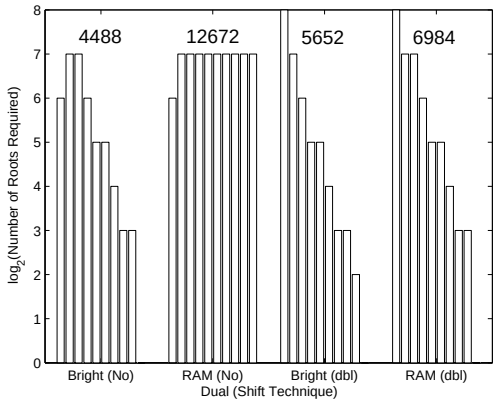
The arrival process considered in this section has $m = 3$ states and the interarrival time in state i is uniformly distributed between 1 and U_i, with $U_1 = 12$, $U_2 = 20$ and $U_3 = 32$. At each arrival instant the state changes from i to $i \bmod 3+1$ with probability $1/s_i$, with $s_1 = 20x$, $s_2 = 2x$ and $s_3 = 8x$. Thus, as x increases the arrival process becomes more correlated. We will consider both $x = 1$ and $x = 100$. We consider three types of customers. A customer generated in phase i is of type i with probability 1/2 and of type $j \neq i$ with probability 1/4, meaning while in state i, twice as many type i customers are generated on average, than type $j \neq i$ customers. The service time of a type i customer has an discrete Erlang distribution with k_i phases and parameter p_i, meaning its mean service time is k_i/p_i. We set $p_1 = 1/2$, $p_2 = 1/3$ and $p_3 = 2/3$, while $k_1 = 2 + y$, $k_2 = 4 + y$ and $k_3 = 3 + y$, with $y = 0$ or 1. Having $y = 0$ results in an overall load of 0.6669 (medium load), while $y = 1$ gives a load of 0.8904 (high load). This results in a GI/M/1-type Markov chain with 33 nonzero blocks A_1 to A_{-31}, each of size 27 (for $y = 0$) or 36 (for $y = 1$).

Figure 1 depicts the number of roots required per iteration to compute the R matrix of the GI/M/1-type Markov chain, by taking the dual M/G/1-type Markov chain and running the cyclic reduction algorithm with or without shifting. Four combinations are considered: the Bright or Ramaswami dual and no shift or the double shift. The two other types of shifting were by far inferior to the double shift as they required many more roots per iteration, therefore they are not included in the figure. For each of these four combinations, we consider a medium ($y = 0$) and high ($y = 1$) load scenario in combination with low ($x = 1$) and high ($x = 100$) correlation.

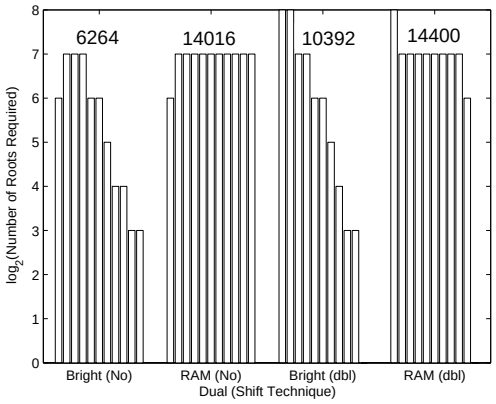
Apart from depicting a bar per iteration that reflects the number of roots required, we also added an estimate (instead of a machine dependent measurement) for the number of flops needed to perform a single FFT of the required size for each iteration (above the bars), where we estimated the number of flops of a size N FFT as $3/2N \log N$ (the exact number depends on the FFT-variant selected and requires at least $N \log N$ multiplications). Thus, the ratio between two such numbers indicates how much faster/slower the computation will be.

When we compare the Bright and Ramaswami dual without shifting, we find that the Bright dual performs up to 3 times faster. The lower the load or correlation gets, the bigger the gain. This is as expected, as higher loads and stronger correlation implies that the dominant eigenvalue η of R approaches one, which determines the speed at which the number of required roots decreases for the Bright dual. For the high load and correlation case in Figure 1(d) we find $\eta = 0.9998$, meaning it takes many iterations before a gain is achieved by the Bright dual. For the medium load and low correlation scenario in Figure 1(a) the dominant eigenvalue η still equals 0.942, which explains why we only gain a factor 3. For instance, setting $y = -1$ would result in a load of 0.4434 and for $x = 1$ the dominant eigenvalue η would decrease to 0.8536, resulting in an improvement of a factor 4.68. For even smaller values of η improvements of a factor 40 or more can be achieved (see [14]).

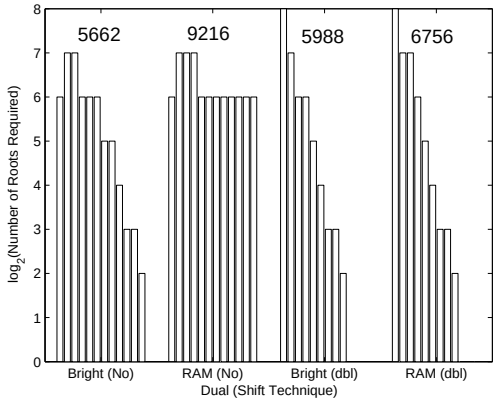
Figure 1 indicates that the shift operation reduces the number of iterations most notably for the high load scenarios (in Figure 1(c) from 12 to 9 iterations, in Figure 1(d) from 21 to 14). However, more roots are sometimes required during the first few iterations, meaning the overall performance is only better in the high load scenario. For the double shift, the gain achieved by the Bright dual is far less pronounced, as the double shift typically also induces a reduction in the number of roots required per iteration for the Ramaswami dual (though the rate is $1/\eta$ times as slow, see [14]). For the high load and correlation scenario, we find an identical performance, which is not that remarkable as $\eta = 0.9998$, meaning the rate at which the number of roots decreases is nearly identical. Although not shown in Figure 1, the two other types of shifting are inferior to the no shift even for the high load scenario, due to the (much) higher number of roots required during each iteration.



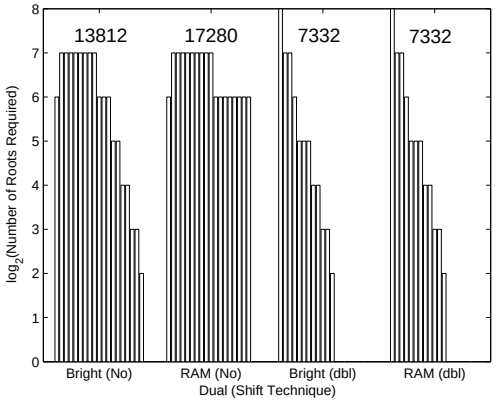
(a) Low correlation ($x = 1$), medium load ($y = 0$)



(b) High correlation ($x = 100$), medium load ($y = 0$)



(c) Low correlation ($x = 1$), high load ($y = 1$)



(d) High correlation ($x = 100$), high load ($y = 1$)

Figure 1: Number of roots required per iteration during the computation of the R matrix using the cyclic reduction algorithm

7. CONCLUSIONS

In this paper we presented an extension of the software package SMCSolver both in the Matlab and in the Fortran 90 versions. This extension includes a preprocessing of the original problem based on a more general shift technique and on some dual transformations.

The Matlab version of the software is available at `ftp://ftp.win.ua.ac.be/pub/pats/tools`; the Fortran 90 version with the GUI is available at `http://bezout.dm.unipi.it/SMCSolver`.

8. REFERENCES

[1] D. A. Bini, L. Gemignani, and B. Meini. Solving certain matrix equations by means of Toeplitz computations: algorithms and applications. In *Fast algorithms for structured matrices: theory and applications (South Hadley, MA, 2001)*, volume 323 of *Contemp. Math.*, pages 151–167. Amer. Math. Soc., Providence, RI, 2003.

[2] D. A. Bini, G. Latouche, and B. Meini. *Numerical methods for structured Markov chains*. Numerical Mathematics and Scientific Computation. Oxford University Press, New York, 2005. Oxford Science Publications.

[3] D. A. Bini and B. Meini. The cyclic reduction algorithm: from Poisson equation to stochastic processes and beyond. *Numerical Algorithms*, 51:23–61, 2009.

[4] D. A. Bini, B. Meini, and V. Ramaswami. A probabilistic interpretation of cyclic reduction and its relationships with logarithmic reduction. *Calcolo*, 45(3):207–216, 2008.

[5] D. A. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chains solver: algorithms. *Proceedings of SMCTOOLS, Pisa 2006*, 2006.

[6] D. A. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chains solver: software tools. *Proceedings of SMCTOOLS, Pisa 2006*, 2006.

[7] L. Bright. *Matrix-Analytic Methods in Applied Probability*. PhD thesis, University of Adelaide, 1996.

[8] C. He, B. Meini, and N. H. Rhee. A shifted cyclic reduction algorithm for quasi-birth-death problems. *SIAM J. Matrix Anal. Appl.*, 23(3):673–691 (electronic), 2001/02.

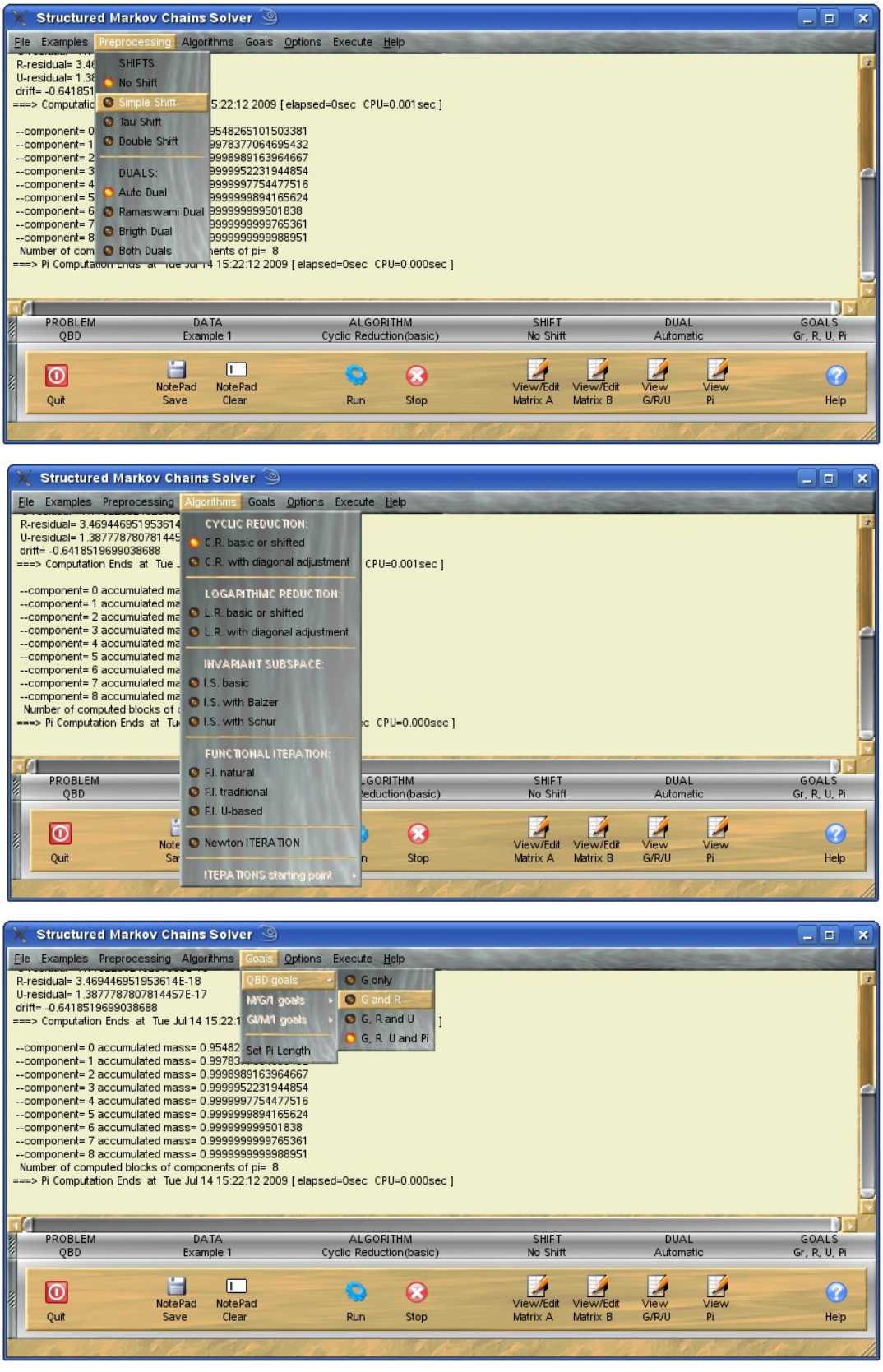


Figure 2: Examples from the GUI: main menu with submenus "Preprocessing", "Algorithms" and "Goals"

[9] Q. He. Age process, workload process, sojourn times, and waiting times in a discrete-time SM[K]/PH[K]/1/FCFS queue. *Queueing Systems*, 49:363–403, 2005.

[10] G. Latouche and V. Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*. ASA-SIAM Series on Statistics and Applied Probability. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[11] M. F. Neuts. *Matrix-geometric solutions in stochastic models*, volume 2 of *Johns Hopkins Series in the Mathematical Sciences*. Johns Hopkins University Press, Baltimore, Md., 1981. An algorithmic approach.

[12] M. F. Neuts. *Structured stochastic matrices of $M/G/1$ type and their applications*, volume 5 of *Probability: Pure and Applied*. Marcel Dekker Inc., New York, 1989.

[13] V. Ramaswami. A duality theorem for the matrix paradigms in queueing theory. *Stochastic Models*, 6:151–161, 1990.

[14] P. G. Taylor and B. Van Houdt. On the dual relationship between Markov chains of GI/M/1 and M/G/1 type. 2009. Submitted.