The Vectorized and Parallelized Solving of Markovian Models for Optical Networks

Beata Bylina and Jarosław Bylina

Department of Computer Science, Marie Curie-Skłodowska University
Pl. M. Curie-Skłodowskiej 1, 20-031 Lublin, Poland
{beatas, jmbylina}@hektor.umcs.lublin.pl

Abstract. The article presents two approaches to the WZ factorization – specific ones for solving Markov chains – and the results of their vectorization and parallelization.

1 Introduction

The probabilistic methods – especially Markov models – are the most useful ones to describe queueing models. A homogeneous continuous-time Markov chain can be described with one singular matrix $\mathbf{Q} = (q_{ij})_{i=1,\ldots,n}^{j=1,\ldots,n}$ called the transition rate matrix given by $q_{ij} = \lim_{\Delta t \to 0} \frac{p_{ij}(\Delta t)}{\Delta t}$ for $i \neq j$ and by $q_{ii} = -\sum_{j \neq i} q_{ij}$.

We are to find $\mathbf{x} = \pi^T$ – the vector of the stationary probabilities π_i that the system is in the state i at the time t – from:

$$\mathbf{Q}^T\mathbf{x} = 0, \qquad \mathbf{x} \geq 0, \qquad \mathbf{e}^T\mathbf{x} = 1, \quad \text{where } \mathbf{e} = (1, 1, \ldots, 1)^T. \tag{1}$$

2 The WZ Factorization

The WZ factorization is described in [3]. $\mathbf{A} = \mathbf{WZ}$, where (for an even n) $\mathbf{W} = (w_{ij})_{i=1,\ldots,n}^{j=1,\ldots,n}$ is shaped like a butterfly ($w_{ij} = 0$ for $i < j < n - i + 1$ and for $n - i - 1 < j < i$, $w_{ii} = 1$) and $\mathbf{Z} = (z_{ij})_{i=1,\ldots,n}^{j=1,\ldots,n}$ is shaped like a transposed butterfly ($w_{ij} = 0$ for $j < i < n - j$ and for $n - j < i < j$). After the factorization we can solve two linear systems: $\mathbf{Wc} = \mathbf{b}$ and $\mathbf{Zx} = \mathbf{c}$ instead of one $\mathbf{Ax} = \mathbf{b}$.

The sequential algorithm for solving the linear system with the WZ factorization is presented on the figure 1. On the figure 2 we present its vectorized counterpart (from [1]).

3 Replacing an Equation (RWZ)

The most intuitive approach to solving a homogenous linear system (1) is to replace an arbitrary equation of that system with the normalization equation $\mathbf{e}^T\mathbf{x} = 1$. Let $\widetilde{\mathbf{Q}}_p$ be the matrix $\mathbf{Q}$ with the pth column replaced with the vector $\mathbf{e}$. Our modified system can be written $\widetilde{\mathbf{Q}}_p^T\mathbf{x} = \mathbf{e}_p$, where $\mathbf{e}_p = (\delta_{ip})_{i=1,\ldots,n}$.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3037, pp. 578–581, 2004.

```
% elimination loop – steps of reduction from A to Z:
for k = 0 : m
  k2 = n-k-1;
  det = A(k,k)*A(k2,k2)-A(k2,k)*A(k,k2);
% finding elements of W
  for i = k+1 : k2-1
    wk1 = (A(k2,k)*A(i,k2)-A(k2,k2)*A(i,k))/det;
    wk2 = (A(k,k2)*A(i,k)-A(k,k)*A(i,k2))/det;
% updating A
    for j = k+1 : k2-1
      A(i,j) = A(i,j)+wk1*A(k,j)+wk2*A(k2,j);
% updating b
    b(i) = b(i)+wk1*b(k)+wk2*b(k2);
% finding x
for j = m : 0
% solving a 2 × 2 linear system
  j2 = n-j-1;
  det = A(j,j)*A(j2,j2)-A(j2,j)*A(j,j2);
  x(j) = (b(j)*A(j2,j2)-b(j2)*A(j,j2))/det;
  x(j2) = (b(j2)*A(j,j)-b(j)*A(j2,j))/det;
% updating b
  for i = j-1 : 0
    i2 = n-i-1
    b(i) = b(i)-x(j)*A(i,j)-x(j2)*A(i,j2);
    b(i2) = b(i2)-x(j)*A(i2,j)-x(j2)*A(i2,j2);
```

Fig. 1. The sequential algorithm for the WZ factorization

```
% elimination loop – steps of reduction from A to Z:
for k = 0 : m
  k2 = n-k-1;
  det = A(k,k)*A(k2,k2)-A(k2,k)*A(k,k2);
  for i = k+1 : k2-1
% finding elements of W
    wk1 = (A(k2,k)*A(i,k2)-A(k2,k2)*A(i,k))/det;
    wk2 = (A(k,k2)*A(i,k)-A(k,k)*A(i,k2))/det;
% updating A
    A(i,k+1:k2-1) = A(i,k+1:k2-1)+wk1*A(k,k+1:k2-1)+wk2*A(k2,k+1:k2-1);
% updating b
    b(i) = b(i)+wk1*b(k)+wk2*b(k2);
% finding x
for j = m : 0
% solving a 2 × 2 linear system – as on figure 1
% updating the upper and the lower part of b
  b(0:j-1) = b(0:j-1)-x(j)*A(0:j-1,j)-x(j2)*A(0:j-1,j2);
  b(n-j:n-1) = b(n-j:n-1)-x(j)*A(n-j:n-1,j)-x(j2)*A(n-j:n-1,j2);
```

Fig. 2. The vectorized algorithm for the WZ factorization

Let $\widetilde{\mathbf{Q}}_p^T = \widetilde{\mathbf{W}}\widetilde{\mathbf{Z}}$. Setting $\widetilde{\mathbf{Z}}\mathbf{x} = \mathbf{y}$ in the system $\widetilde{\mathbf{W}}\widetilde{\mathbf{Z}}\mathbf{x} = \mathbf{e}_p$ we get $\widetilde{\mathbf{W}}\mathbf{y} = \mathbf{e}_p$ from which it is obvious that $\mathbf{y} = \mathbf{e}_p$. So now we are to solve the system $\widetilde{\mathbf{Z}}\mathbf{x} = \mathbf{e}_p$.

This approach is likely to yield a less accurate result than the next one (section 4). When we compute x_p and x_q they will be contaminated with the round-off errors from all of the previous elimination steps. Moreover, this will propagate throughout the next backsubstitution steps.

4 Removing an Equation (DWZ)

Another approach is to remove an equation. We know that the rank of $\mathbf{Q}$ is $(n-1)$ – that is one of the equations can be written as a linear combinantion of other equations. If we drop an equation we get a linear system of $(n-1)$ equations with n unknowns (and the normalization equation).

In this approach we divide (after [4]) our matrix in blocks $\mathbf{Q}^T = \begin{bmatrix} \mathbf{B} & \mathbf{d} \\ \mathbf{c}^T & f \end{bmatrix}$ where $\mathbf{B}$ is a nonsingular matrix of the size $(n-1)$, $\mathbf{c}$ and $\mathbf{d}$ are $(n-1)$-element vectors and f is a real number.

Let us assign $x_n = 1$, now $\mathbf{x}^T = (\widehat{\mathbf{x}}^T, 1)$ and our equation (1) gives the equatoins: $\mathbf{B}\widehat{\mathbf{x}} + \mathbf{d} = 0$ and $\mathbf{c}^T\widehat{\mathbf{x}} + f = 0$.

Now we can solve the linear system without the last equation, that is only $\mathbf{B}\widehat{\mathbf{x}} = -\mathbf{d}$. We solve it using the WZ factorization – the matrix $\mathbf{B}$ is factorized: $\mathbf{B} = \mathbf{WZ}$ and the equations $\mathbf{W}\mathbf{y} = -\mathbf{d}$ and $\mathbf{Z}\widehat{\mathbf{x}} = \mathbf{y}$ are solved. Now we must normalize the solution vector $\mathbf{x}^T = (\widehat{\mathbf{x}}^T, 1)$.

Of course, whichever equation can be dropped, not only the last.

5 The Example, the Implementation, and the Experiments

A queuing model of an optical network edge node is presented in [2]. Information (electronical) packages arrive in a buffer (of an optical switch) of capacity of $N = 250$ blocks. The packages are of different sizes (sizes are integer, from one block up to 20 blocks). When the buffor is filled, the bigger (optical) package is formed and sent. The buffer is then emptied. We don't want to divide information from one electronical package between two optical packages so when an arriving package is too big to fit into the buffer we send an uncompleted optical package and package just received starts a new optical package. An uncompleted optical package is also sent when a given timeout is over. Arriving packages stream has a Poisson distribution and the probability that the received packages have i $(i = 1, \ldots, 20)$ blocks is $p_i = 0.05$. In our model the timeout is approximated with an Erlang distribution consisting of 10 phases of exponential distribution because we want to preserve a Markovian form of this model.

In our experiments the transition rates matrix ($\mathbf{Q}$) was not very big (only 2491 states) so we decided to store it in a traditional, two-dimensional array

with a lot of zeroes. This storing scheme is rather space consuming, but is the best when we have enough space and the computation time matters.

The algorithms **RWZ** and **DWZ** were implemented for the single precision numbers with the use of the language C. The programs were compiled with the *icc* (*Intel C Compiler*). We denoted the algorithms **RWZ** and **DWZ** improved with the vectorized algorithm presented in the figure 2 by **VRWZ** and **VDWZ**, respectively. The algorithms **VRWZ** and **VDWZ** were implemented with the use of the *BLAS1* functions from the *mkl* (Intel's *Mathematics Kernel Library*). All the programs were tested on a Pentium III 733 MHz machine.

Table 1. The performance and the residual for the tested algorithms

algorithm	$\|\|\mathbf{Q}^T\mathbf{x}\|\|_2$	**time A [s]**	**time B [s]**
DWZ	3.84383e–07	239.98	228.65
VDWZ	5.69805e–07	9.84	231.82
RWZ	1.91620e–06	232.96	229.08
VRWZ	1.93863e–06	15.47	220.02

All the described algorithms were tested as not parallelized ones and as parallelized (with the use of the *OpenMP* standard) algorithms – but the parallelization did not give the significiant improvement in performance (no more than 5%), so in the table 1 we present the residual (here: $||\mathbf{Q}^T\mathbf{x}||_2$) and performance (as **time A**) of the not parallelized ones only.

To better understand such a big improvement in performance (with the use of *BLAS1* only) we made some tests with the same algorithms but with other matrices. Namely we prepared some random matrices with all non-zero elements. The performance for such dense matrices (**time B** in table 1) for not vectorized algorithms was similar to performance for our sparse (54782 non-zeroes for $2491^2 = 6205081$ elements) matrix. However, for vectorized ones the performance for the sparse matrix was much better than the performance for the dense ones – because of the special treatment of huge number of zeroes by the *mkl* and the processor.

References

1. Bylina, B.: Solving lineaar systems with vectorized WZ factorization. Annales UMCS Informatica **1** (2003) 5–13
2. Domańska, J., Czachórski, T.: Model formowania pakietów w węźle brzegowym sieci optycznej. Studia Informatica **2A(53)** (2003) 51–62 (in Polish)
3. Evans, D.J., Hatzopoulos, M.: The parallel solution of linear system. Int. J. Comp. Math. **7** (1979) 227–238
4. Stewart, W.: Introduction to the Numerical Solution of Markov Chains. Princeton University Press, Chichester, West Sussex (1994)