Analyzing Large Sparse Markov Chains of Kronecker Products

Tuğrul Dayar
Department of Computer Engineering
Bilkent University
tugrul@cs.bilkent.edu.tr

13 September 2009

Outline

- Background
  - ▲ Kronecker representation of Q
  - ▲ An example
- Preprocessing
  - ▲ Reordering and grouping
  - ▲ Lumping
- Block iterative methods
  - ▲ Splitting the smaller matrices
  - ▲ Example (continued)
  - ▲ Block iterative methods for Kronecker products
- Multilevel methods
  - ▲ The simple multilevel method for Kronecker products
  - ▲ Example (continued)
  - ▲ A class of multilevel methods for Kronecker products
- Preconditioned projection methods
- Conclusion

Background

- *Continuous-time Markov chain* (CTMC) having n states is represented by $(n \times n)$ square matrix $Q \in \mathbb{R}^{n \times n}$ having

$$Q(i,j) \geq 0 \ \forall i \neq j \quad \text{and} \quad Q(i,i) = -\sum_{j \neq i} Q(i,j) \ \forall i.$$

- *Initial* distribution (row) vector: $\boxed{\pi_0 \in \mathbb{R}^{1 \times n}}$, where

  $\pi_0 \geq 0$, $\pi_0 e = 1$, and e is column vector of ones.

- *Transient* vector at time $t \geq 0$:

$$\pi_t = \pi_0 e^{Qt} = \pi_0 \sum_{i=0}^{\infty} \frac{(Qt)^i}{i!}.$$

- *Steady-state* (or *limiting*, *long-run*) vector

$$\pi = \lim_{t \to \infty} \pi_t \quad \text{satisfies} \quad \pi Q = 0, \quad \pi e = 1$$

  whenever it exists; it is also *stationary* distribution.

Background (continued)

- In Kronecker based approach [Dayar'06], Q is:
  - ▲ represented using Kronecker products of *smaller* matrices
  - ▲ *never* explicitly generated.

- Implementation of transient and steady-state solvers can rest on this compact representation, thanks to existence of:

  efficient *vector-Kronecker product multiplication* algorithm known as *shuffle* algorithm [Davio'81].

- Various algorithms for vector-Kronecker product multiplication based on shuffle algorithm are devised:

  [Benoit-Fernandes-Plateau-Stewart'04b]
  [Buchholz-Ciardo-Donatelli-Kemper'00]
  [Fernandes-Plateau-Stewart'98ab]
  [Plateau-Fourneau'91], [Plateau-Fourneau-Lee'88]

  and used as *kernels* in solvers proposed for different modeling formalisms.

Background (continued)

- In practice, Kronecker based representation of Q is obtained using various modeling formalisms:
  - ▲ *Compositional Markovian models* such as stochastic automata networks (SANs) [Plateau'85, Plateau-Atif'91, Plateau-Fourneau'91, Stewart'94] and different classes of superposed stochastic Petri nets [Donatelli'93, Kemper'96]
  - ▲ *Hierarchical Markovian models* (HMMs) of queueing networks [Buchholz'94a], generalized stochastic Petri nets (GSPNs) [Buchholz-Kemper'98], or systems of asynchronously communicating stochastic modules [Campos-Donatelli-Silva'99]
  - ▲ *Stochastic process algebras*, such as the performance evaluation process algebra (PEPA) [Hillston-Kloul'01].

- These modeling formalisms are integrated to various software packages:
  - ▲ *Abstract Petri Net Notation* toolbox [APNN'04, Bause-Buchholz-Kemper'98]
  - ▲ *Performance Evaluation of Parallel Systems* software tool [Benoit-Brenner-Fernandes-Plateau-Stewart'03, PEPS'03]
  - ▲ *PEPA* Workbench [Clark-Gilmore-Hillston-Thomas'99, PEPA'05]
  - ▲ *Stochastic Model checking Analyzer for Reliability and Timing* [Ciardo-Jones-Miner-Siminiceanu'03, SMART'04].

Background (continued)

- Transient distribution can be computed through *uniformization* using vector-Kronecker product multiplications [Buchholz'94a].
- Steady-state distribution also needs to be computed using vector-Kronecker product multiplications [Buchholz'99c,Stewart-Atif-Plateau'95], since direct methods based on complete factorizations, such as Gaussian elimination, normally introduce new nonzeros which cannot be accommodated.
- To deal with the *reachable state space* rather than the *product state space*, one can use:
  - ▲ HMMs [Buchholz'94a]
  - ▲ Matrix diagrams [Ciardo-Miner'99] and representations for specific models [Haddad-Moreaux'96]
  - ▲ Other approaches [Benoit-Fernandes-Plateau-Stewart'04b, Buchholz'99b, Buchholz-Ciardo-Donatelli-Kemper'00].
- Most work is on CTMCs; very few DTMCs based on Kronecker products discussed in literature.

Background (continued)

We take an algebraic view and make the assumption that MC at hand:

- does not have *unreachable* states
- is *irreducible*.

Kronecker (or *tensor*) *product* of two (rectangular) matrices $A \in \mathbb{R}^{n_A \times m_A}$ and $B \in \mathbb{R}^{n_B \times m_B}$ is written as $\boxed{A \otimes B}$ and yields the (rectangular) matrix $C \in \mathbb{R}^{n_A n_B \times m_A m_B}$ given by:

$$\boxed{c(i_C, j_C) = a(i_A, j_A)b(i_B, j_B)}$$

with

$$i_C = (i_A - 1)n_B + i_B \quad \text{and} \quad j_C = (j_A - 1)m_B + j_B$$

$$(i_A, j_A) \in \{1, 2, \ldots, n_A\} \times \{1, 2, \ldots, m_A\}$$

$$(i_B, j_B) \in \{1, 2, \ldots, n_B\} \times \{1, 2, \ldots, m_B\},$$

where $\times$ is the *Cartesian product* operator.

Background (continued)

■ In a 2-dimensional representation:

▲ *row indices* of $C \in \{1, 2, \ldots, n_A\} \times \{1, 2, \ldots, n_B\}$
▲ *column indices* of $C \in \{1, 2, \ldots, m_A\} \times \{1, 2, \ldots, m_B\}$.

■ Ordering of rows and columns of C is *lexicographical*, since

$$c(i_C, j_C) = c((i_A, i_B), (j_A, j_B)) = c((i_A - 1)n_B + i_B, (j_A - 1)m_B + j_B).$$

■ Kronecker product is associative; Kronecker product of H *square* matrices is:

$$\boxed{X = X^{(1)} \otimes X^{(2)} \otimes \cdots \otimes X^{(H)} = \otimes_{h=1}^{H} X^{(h)},}$$

where

▲ $X^{(h)} \in \mathbb{R}^{n_h \times n_h}$
▲ row/column indices of $X^{(h)} \in \mathcal{S}^{(h)} = \{1, 2, \ldots, n_h\}$ for $h = 1, 2, \ldots, H$
▲ $X \in \mathbb{R}^{n \times n}$ with $\boxed{n = \prod_{h=1}^{H} n_h}$.

Background (continued)

H-dimensional state space representation

■ Ordered H-dimensional tuples

$$\boxed{(i_1, i_2, \ldots, i_H) \in \times_{h=1}^{H} \mathcal{S}^{(h)} \quad \text{and} \quad (j_1, j_2, \ldots, j_H) \in \times_{h=1}^{H} \mathcal{S}^{(h)}}$$

used to represent *row* and *column* indices of X, respectively.

■ Kronecker product of H square matrices implies:

one-to-one onto mapping between an H-dimensional state space and a one-dimensional state space that are lexicographically ordered.

■ Kronecker product can be used to define:

MCs having *multi-dimensional state spaces*.

Kronecker representation of Q

Assume that H-dimensional CTMC at hand is represented as:

$$Q = Q_O + Q_D, \quad Q_O = \sum_{k=1}^{K} \bigotimes_{h=1}^{H} Q_k^{(h)}, \quad Q_D = \mathsf{diag}(-Q_O e),$$

where

- Q_O: *off-diagonal* part of Q ($Q_O \geq 0$)
- Q_D: *diagonal* part of Q ($Q_D \leq 0$)
- K: *# of Kronecker products* (or *terms*) forming Q_O
- H: *# of factors* in each Kronecker product
- $Q_k^{(h)} \in \mathbb{R}^{n_h \times n_h}$
  $Q_k^{(h)} \geq 0$ for $k = 1, 2, \ldots, K$ and $h = 1, 2, \ldots, H$
- diag: *diagonal matrix* which has its vector argument along its diagonal.

Kronecker representation of Q (continued)

- If row/column indices of $Q_k^{(h)} \in \mathcal{S}^{(h)} = \{1, \ldots, n_h\}$ for $k = 1, 2, \ldots, K$ and $h = 1, 2, \ldots, H$, then *H-dimensional state space* of Q is given by:

$$\mathcal{S} = \times_{h=1}^{H} \mathcal{S}^{(h)}.$$

- $|\mathcal{S}| = \prod_{h=1}^{H} |\mathcal{S}^{(h)}| = \prod_{h=1}^{H} n_h = n$.
- *One-dimensional value* of state $s \in \mathcal{S}$ corresponding to $(s_1, s_2, \ldots, s_H)$, where $s_h \in \mathcal{S}^{(h)}$ for $h = 1, 2, \ldots, H$, is given by:

$$s = 1 + \sum_{h=1}^{H} (s_h - 1) \prod_{i=h+1}^{H} n_i.$$

- We will be using one-dimensional and multi-dimensional representations of states interchangeably.

Kronecker representation of Q (continued)

<u>Space complexity</u>

■ One needs space for:

   ▲ diagonal matrix Q_D

   ▲ matrices in the Kronecker representation of Q_O,

   meaning a floating-point vector of length $\prod_{h=1}^{H} n_h$ and at most K (sparse) floating-point matrices of order n_h are stored for $h = 1, 2, \ldots, H$.

■ In the worst case, this amounts to a storage space of $\boxed{n + \sum_{h=1}^{H} nz_{Q^{(h)}}}$ floating-point values, where

$$nz_{Q^{(h)}}: \text{sum of } \# \text{ of nonzeros in } Q_k^{(h)} \text{ for } k = 1, 2, \ldots, K.$$

■ Q_D can also be expressed as a sum of Kronecker products:

$$\boxed{Q_D = - \sum_{k=1}^{K} \bigotimes_{h=1}^{H} \text{diag}(Q_k^{(h)} e).}$$

However, most of the time Q_D is precomputed and stored explicitly.

Kronecker representation of Q (continued)

<u>Time complexity</u>

Complexity of a vector multiplication with Q_O, which consists of K Kronecker product terms, is given by:

$$
\begin{aligned}
K \prod_{h=1}^{H} n_h + 2 \sum_{k=1}^{K} \prod_{h=1}^{H} n_h \sum_{l=1}^{H} nz_{Q_k^{(l)}} / n_l &= K \prod_{h=1}^{H} n_h + 2 \prod_{h=1}^{H} n_h \sum_{l=1}^{H} \left(\sum_{k=1}^{K} nz_{Q_k^{(l)}} \right) / n_l \\
&= n \left(K + 2 \sum_{h=1}^{H} nz_{Q^{(h)}} / n_h \right)
\end{aligned}
$$

floating-point arithmetic operations [Fernandes-Plateau-Stewart'98a], where:

$$nz_{Q_k^{(l)}}: \# \text{ of nonzeros in } Q_k^{(l)} \text{ for } k = 1, 2, \ldots, K \text{ and } l = 1, 2, \ldots, H.$$

Kronecker representation of Q (continued)

- Each nonzero element of the matrix $Q_k^{(h)}$ in Q is located by its row and column indices, which are members of $\mathcal{S}^{(h)}$.
- More generally, a nonzero element of $Q_k^{(h)}$ may be a function of states in state spaces other than $\mathcal{S}^{(h)}$, thus a function of non-local states.
- This phenomenon is a by-product of the modeling process and has been utilized in the SAN modeling formalism:
  - ▲ These nonzero elements are referred to as *functional transitions*.
  - ▲ Corresponding Kronecker products are said to be *generalized* [Plateau'85].
- It is possible to remove functional transitions from a sum of generalized Kronecker products by introducing new terms [Plateau-Fourneau'91] and/or factors [Benoit-Fernandes-Plateau-Stewart'04b], but functional transitions enable a more compact Kronecker representation with fuller factor matrices [Chung-Ciardo-Donatelli-He-Plateau-Stewart-Sulaiman-Yu'04].
- We do not consider functional transitions, but indicate that results extend to generalized Kronecker products whereever appropriate.

Kronecker representation of Q (continued)

- Let level 0 denote highest level at which Q is perceived as single block of order $n = \prod_{h=1}^{H} n_h$.
- For $l = 0, 1, \ldots, H$, we have:

$$\boxed{b_l = \prod_{h=1}^{l} n_h^2 \quad \text{and} \quad o_l = \prod_{h=l+1}^{H} n_h,}$$

  where

  > b_l: # of blocks at level l
  > o_l: order of blocks at level l.

- There are $\sqrt{b_l}$ blocks each of order o_l along the diagonal of Q.
- Block $((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l))$ of Q at level l:

$$
Q((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l)) = \sum_{k=1}^{K} \left(\prod_{h=1}^{l} q_k^{(h)}(i_h, j_h) \right) \left(\bigotimes_{h=l+1}^{H} Q_k^{(h)} \right) \\
+ Q_D((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l))
$$

  for $l = 0, 1, \ldots, H$.

Kronecker representation of Q (continued)

- $Q_D((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l))$ is block $((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l))$ of Q_D.

  $Q_D((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l)) = 0$ if $(i_1, i_2, \ldots, i_l) \neq (j_1, j_2, \ldots, j_l)$,
  meaning it is off-diagonal block at level l.

- $l = 0$ yields (block) Q and $l = H$ yields scalar (block):

$$q((i_1, i_2, \ldots, i_H), (j_1, j_2, \ldots, j_H)) = \sum_{k=1}^{K} \prod_{h=1}^{H} q_k^{(h)}(i_h, j_h)$$
$$+ q_D((i_1, i_2, \ldots, i_H), (j_1, j_2, \ldots, j_H)).$$

- *Nested* and *recursive* structure associated with Q is also valid in the presence of *functional transitions*.

An example

Consider following matrices for a 4-dimensional problem (each dimension with 2 states) having 7 terms of Kronecker products:

$$Q_1^{(1)} = \begin{pmatrix} & 1 \\ & \end{pmatrix}, \; Q_2^{(1)} = Q_3^{(1)} = Q_4^{(1)} = Q_5^{(1)} = I, \; Q_6^{(1)} = \begin{pmatrix} & 1 \\ & \end{pmatrix}, \; Q_7^{(1)} = \begin{pmatrix} & \\ 10 & \end{pmatrix}$$

$$Q_1^{(2)} = I, \; Q_2^{(2)} = \begin{pmatrix} & \\ 1 & \end{pmatrix}, \; Q_3^{(2)} = Q_4^{(2)} = Q_5^{(2)} = Q_6^{(2)} = I, \; Q_7^{(2)} = \begin{pmatrix} & 1 \\ & \end{pmatrix}$$

$$Q_1^{(3)} = Q_2^{(3)} = I, \; Q_3^{(3)} = \begin{pmatrix} & 1 \\ & \end{pmatrix}, \; Q_4^{(3)} = I, \; Q_5^{(3)} = \begin{pmatrix} & 1 \\ & \end{pmatrix}, \; Q_6^{(3)} = \begin{pmatrix} & \\ 10 & \end{pmatrix}, \; Q_7^{(3)} = I$$

$$Q_1^{(4)} = Q_2^{(4)} = Q_3^{(4)} = I, \; Q_4^{(4)} = \begin{pmatrix} & 1 \\ & \end{pmatrix}, \; Q_5^{(4)} = \begin{pmatrix} & \\ 10 & \end{pmatrix}, \; Q_6^{(4)} = I, \; Q_7^{(4)} = I$$

Then,

$$\boxed{Q = \sum_{k=1}^{7} \bigotimes_{h=1}^{4} Q_k^{(h)} + Q_D.}$$

An example (continued)

$$
Q =
\begin{array}{cccc|cccc|cccc|cccc}
 & & & & \text{1} & \text{1} & \text{1} & \text{1} & \text{1} & \text{1} & \text{1} & \text{1} & \text{2} & \text{2} & \text{2} & \text{2} & \text{2} & \text{2} & \text{2} & \text{2} \\
\end{array}
$$

Column labels (top, read top-to-bottom per column):

1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2
1	1	1	1	2	2	2	2	1	1	1	1	2	2	2	2
1	1	2	2	1	1	2	2	1	1	2	2	1	1	2	2
1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2

$$
Q = \begin{pmatrix}
-3 & 1 & 1 & & & & & & & 1 & & & & & & \\
 & -12 & 10 & 1 & & & & & & & 1 & & & & & \\
 & & -12 & 1 & & & & & 10 & & 1 & & & & & \\
 & & & -11 & & & & & & 10 & & 1 & & & & \\
1 & & & & -4 & 1 & 1 & & & & & & 1 & & & \\
 & 1 & & & & -13 & 10 & 1 & & & & & & 1 & & \\
 & & 1 & & & & -13 & 1 & & & & & 10 & & 1 & \\
 & & & 1 & & & & -12 & & & & & & 10 & & 1 \\
 & & & & 10 & & & & -12 & 1 & 1 & & & & & \\
 & & & & & 10 & & & & -21 & 10 & 1 & & & & \\
 & & & & & & 10 & & & & -11 & 1 & & & & \\
 & & & & & & & 10 & & & & -10 & & & & \\
 & & & & & & & & 1 & & & & -3 & 1 & 1 & \\
 & & & & & & & & & 1 & & & & -12 & 10 & 1 \\
 & & & & & & & & & & & & & & -2 & 1 \\
 & & & & & & & & & & & & & 1 & & -1 \\
\end{pmatrix}
$$

- ■ # of floating-point values stored in Kronecker representation is 10 for matrices and 16 for diagonal, thus totaling 26; whereas, it is 60 for flat representation.
- ■ Discrepancy between Kronecker and flat representations becomes substantial for larger values of the state space size, n.

Preprocessing

Preprocessing

- ■ Objective is to expedite analysis of MCs based on Kronecker products.
- ■ There are three techniques that can be used to put Kronecker representation into more favorable form before solvers take over.
- ■ These are:
  - ▲ Reordering
  - ▲ Grouping
  - ▲ Lumping

Reordering and grouping

$(K, H) = (1, 1)$ corresponds to a *flat* representation.

- As $H \searrow 1$, Kronecker representation becomes flatter,
  implying increased storage requirements.
- If K were 1, then Q could be analyzed along each dimension independently
  $\Rightarrow$ K is normally assumed to be larger than 1.

$$\boxed{\text{Make } K \text{ as small as possible without changing } H}$$

  $\Rightarrow$ # of terms in Q_O decreases, $Q_k^{(h)}$ become fuller.

Effects of reordering and grouping of factors of Kronecker products on convergence and space requirements of iterative methods have been investigated [Buchholz-Dayar'04a, Buchholz-Dayar'05, Dayar'00, Gusak-Dayar'01, Uysal-Dayar'98], but a broad, systematic study seems to be lacking.

Reordering and grouping (continued)

Reordering in MCs based on Kronecker products refers to:

1. either permuting factors of Kronecker products
2. or renumbering states in state spaces of factors.

Latter corresponds to symmetric permutation of factor matrices $Q_k^{(h)}$
for $k = 1, 2, \ldots, K$ associated with renumbered state space $\mathcal{S}^{(h)}$.

- Reordering of first kind may be used to reduce overhead associated with generalized vector-Kronecker product multiplication [Benoit-Fernandes-Plateau-Stewart'04b, Fernandes-Plateau-Stewart'98b].
- Reordering of both kinds can change nonzero structure of underlying MC
  $\Rightarrow$ can affect convergence of iterative methods [Dayar'98].

Symmetrically permute nonzero structure of underlying MC to more favorable form for iterative method of choice:

  use nonzero structure of $\sum_{k=1}^{K} Q_k^{(h)}$, which indicates how factor h contributes to nonzero structure of Q_O for $h = 1, 2, \ldots, H$.

Reordering and grouping (continued)

Grouping in MCs based on Kronecker products refers to collapsing same adjacent factors in each Kronecker product. Hence:

- factors in each Kronecker product are reduced by same number
- state space sizes of factors are increased.

Results [Benoit-Fernandes-Plateau-Stewart'04b, Fernandes-Plateau-Stewart'98ab] show that in some cases grouping may:

- reduce state space if it had unreachable states
- decrease overhead associated with functional transitions
- decrease number of terms in the Kronecker representation.

Group factors with functional dependencies among each other.

In the absence of functional transitions, group as many factors as possible given available memory starting from highest indexed factor
$\quad\Rightarrow\quad$ flatter representation for diagonal blocks at a particular level, which is useful in certain iterative methods.

Lumping

Lumpability is a property possessed by some MCs which, if conditions are met, may be used to reduce a large state space $\mathcal{S}$ to a smaller state space $\mathcal{S}_{lumped}$.

Find a partitioning of $\mathcal{S}$ such that, when states in each partition are lumped (or *aggregated*) to form a single state, the resulting MC described by $\mathcal{S}_{lumped}$ has *equivalent* behavior to original chain.

We refer to two kinds of lumpability:

1. ordinary lumpability
2. exact lumpability.

Here we give definitions for CTMCs.
Equivalent definitions can be stated for DTMCs.

Lumping (continued)

Q is said to be *ordinarily lumpable* with respect to a partitioning $\mathcal{S} = \cup_i \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for all $i \neq j$ if for all $\mathcal{S}_i \subset \mathcal{S}$ and all $s_i, s_i' \in \mathcal{S}_i$

$$\sum_{s_j \in \mathcal{S}_j} q(s_i, s_j) = \sum_{s_j \in \mathcal{S}_j} q(s_i', s_j) \text{ for all } \mathcal{S}_j \subset \mathcal{S}.$$

Q is said to be *exactly lumpable* with respect to a partitioning $\mathcal{S} = \cup_i \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for all $i \neq j$ if for all $\mathcal{S}_i \subset \mathcal{S}$ and all $s_i, s_i' \in \mathcal{S}_i$

$$\sum_{s_j \in \mathcal{S}_j} q(s_j, s_i) = \sum_{s_j \in \mathcal{S}_j} q(s_j, s_i') \text{ for all } \mathcal{S}_j \subset \mathcal{S}.$$

- Ordinary lumpability refers to a partitioning of S in which sums of transition rates from each state in a partition to a(nother) partition are the same.
- Exact lumpability refers to a partitioning of S in which sums of transition rates from all states in a partition into each state of a(nother) partition are the same.

Lumping (continued)

- On ordinarily lumped MC, one can compute:
  - ▲ performance measures defined over $\mathcal{S}_{lumped}$.
- On exactly lumped MC, one can compute:
  - ▲ steady-state performance measures defined over $\mathcal{S}$
  - ▲ transient performance measures defined over $\mathcal{S}_{lumped}$
  - ▲ transient performance measures defined over $\mathcal{S}$ if states in exactly lumpable partitions have same initial probabilities.

  Since MCs satisfy row sum property rather than column sum property, exact lumpability is more difficult to be satisfied than ordinary lumpability.

Lumping (continued)

- Lumpability can be investigated within each state space $\mathcal{S}^{(h)}$ that defines the Kronecker representation of Q_O for $h = 1, 2, \ldots, H$ independently:

  - ▲ For $\mathcal{S}^{(h)}$, detection of ordinary and exact lumpability through partition refinement [Buchholz'00b] requires a time complexity of $O(nz_{Q^{(h)}} \log n_h)$ and a space complexity of $O(nz_{Q^{(h)}})$.
  - ▲ Lumped Kronecker representation may be obtained by replacing each of $\mathcal{S}^{(h)}$ and its corresponding matrices $Q_k^{(h)}$ for $k = 1, 2, \ldots, K$ with equivalent lumped ones.

- Lumpability can be investigated among $\mathcal{S}^{(h)}$ that are *replicated* (or identical) with respect to Kronecker representation of Q_O [Brenner-Benoit-Fernandes-Plateau'04a]:

  - ▲ Replication is very specific symmetry in Kronecker representation.
  - ▲ Ordinary lumpability of replicated state spaces is shown in the presence of functional transitions.
  - ▲ Performance measures of interest over $\mathcal{S}_{lumped}$ can be computed.

Lumping (continued)

- Lumpability can be investigated among $\mathcal{S}^{(h)}$ by considering dependencies and matrix properties in Kronecker representation [Gusak-Dayar-Fourneau'03ab]:

  - ▲ Sufficient conditions that satisfy ordinary lumpability are specified by identifying ordinarily lumpable partitionings induced by nested block structure of generalized Kronecker representation.
  - ▲ Enables detection of ordinarily lumpable partitionings in which blocks are composed of multiple (non-identical) state spaces but individual state spaces cannot be lumped by themselves.
  - ▲ An iterative steady-state solution method which is able to compute performance measures over $\mathcal{S}$ is given for CTMCs and DTMCs in the presence of functional transitions.

Neither of the last two approaches:

- are completely automated
- use a Kronecker representation for the lumped MC
- possess a proper complexity analysis.

Splitting the smaller matrices

Consider *splitting* smaller matrices that form Kronecker products as in [Uysal-Dayar'98]:

$$Q_k^{(h)} = D_k^{(h)} + U_k^{(h)} + L_k^{(h)}$$

for $k = 1, 2, \ldots, K$ and $h = 1, 2, \ldots, H$,

where

$D_k^{(h)}$: *diagonal* part of $Q_k^{(h)}$
$U_k^{(h)}$: *strictly upper-triangular* part of $Q_k^{(h)}$
$L_k^{(h)}$: *strictly lower-triangular* part of $Q_k^{(h)}$.

Observe that:

$$D_k^{(h)} \geq 0, \ U_k^{(h)} \geq 0, \ L_k^{(h)} \geq 0$$

since $Q_k^{(h)} \geq 0$.

QEST 2009 13 September 2009 − 28 / 57

Splitting the smaller matrices (continued)

Then using Lemma A.8 in [Uysal-Dayar'98], which rests on:

■ *associativity* of Kronecker product
■ *distributivity* of Kronecker product over matrix addition,

it is possible to express Q_O of Q at level $l = 0, 1, \ldots, H$ as

$$Q_O = Q_{U(l)} + Q_{L(l)} + Q_{DU(l)} + Q_{DL(l)},$$

where

$$Q_{U(l)} = \sum_{k=1}^{K} \sum_{h=1}^{l} \left(\bigotimes_{f=1}^{h-1} D_k^{(f)} \right) \otimes U_k^{(h)} \otimes \left(\bigotimes_{f=h+1}^{H} Q_k^{(f)} \right)$$

$$Q_{L(l)} = \sum_{k=1}^{K} \sum_{h=1}^{l} \left(\bigotimes_{f=1}^{h-1} D_k^{(f)} \right) \otimes L_k^{(h)} \otimes \left(\bigotimes_{f=h+1}^{H} Q_k^{(f)} \right)$$

correspond respectively to *strictly block upper- and lower-triangular* parts of Q_O at level l.

QEST 2009 13 September 2009 − 29 / 57

Splitting the smaller matrices (continued)

$$Q_{DU(l)} = \sum_{k=1}^{K} \sum_{h=l+1}^{H} \left(\bigotimes_{f=1}^{h-1} D_k^{(f)} \right) \otimes U_k^{(h)} \otimes \left(\bigotimes_{f=h+1}^{H} Q_k^{(f)} \right)$$

$$Q_{DL(l)} = \sum_{k=1}^{K} \sum_{h=l+1}^{H} \left(\bigotimes_{f=1}^{h-1} D_k^{(f)} \right) \otimes L_k^{(h)} \otimes \left(\bigotimes_{f=h+1}^{H} Q_k^{(f)} \right)$$

correspond respectively to *strictly upper- and lower-triangular parts of block diagonal* of Q_O at level l.
Observe that:
$$Q_{U(l)} \geq 0, \; Q_{L(l)} \geq 0, \; Q_{DU(l)} \geq 0, \; Q_{DL(l)} \geq 0.$$

$$l = 0 \quad \Rightarrow \quad Q_O \text{ is a single block with } Q_{U(0)} = Q_{L(0)} = 0$$
$$l = H \quad \Rightarrow \quad \text{a point-wise partitioning of } Q_O$$
$$\text{with } Q_{DU(H)} = Q_{DL(H)} = 0.$$

Hence, for iterative methods based on block partitionings $l = 1, 2, \ldots, H-1$ should be used.

Example (continued)

Consider block partitioning of the 4-dimensional problem at level 1 for which:

- $l = 1$
- $b_1 = 4$
- Q is viewed as (2×2) block matrix with blocks of order $o_1 = 8$.

$$Q_{U(1)} = \sum_{k=1}^{7} U_k^{(1)} \otimes Q_k^{(2)} \otimes Q_k^{(3)} \otimes Q_k^{(4)} \quad \text{and} \quad Q_{L(1)} = \sum_{k=1}^{7} L_k^{(1)} \otimes Q_k^{(2)} \otimes Q_k^{(3)} \otimes Q_k^{(4)},$$

$$Q_{U(1)} + Q_{L(1)} = \begin{pmatrix} & & & & 1 & & & \\ & & & & 10 & 1 & 1 & \\ & & & & & 10 & & 1 \\ & & & & & & 1 & \\ & & & & & & & 1 \\ & & & & & & 10 & 1 & 1 \\ \hline & & & & & & & 10 & 1 \\ & 10 & & & & & & & \\ & & 10 & & & & & & \\ & & & 10 & & & & & \\ & & & & 10 & & & & \\ & & & & & & & & \end{pmatrix}$$

16

Example (continued)

The strictly upper- and lower-triangular parts of the block diagonal are given by:

$$
\begin{aligned}
Q_{DU(1)} &= \sum_{k=1}^{7} D_k^{(1)} \otimes U_k^{(2)} \otimes Q_k^{(3)} \otimes Q_k^{(4)} + \sum_{k=1}^{7} D_k^{(1)} \otimes D_k^{(2)} \otimes U_k^{(3)} \otimes Q_k^{(4)} \\
&\quad + \sum_{k=1}^{7} D_k^{(1)} \otimes D_k^{(2)} \otimes D_k^{(3)} \otimes U_k^{(4)}, \\
Q_{DL(1)} &= \sum_{k=1}^{7} D_k^{(1)} \otimes L_k^{(2)} \otimes Q_k^{(3)} \otimes Q_k^{(4)} + \sum_{k=1}^{7} D_k^{(1)} \otimes D_k^{(2)} \otimes L_k^{(3)} \otimes Q_k^{(4)} \\
&\quad + \sum_{k=1}^{7} D_k^{(1)} \otimes D_k^{(2)} \otimes D_k^{(3)} \otimes L_k^{(4)},
\end{aligned}
$$

Example (continued)

There are $\sqrt{b_1} = 2$ blocks along the diagonal:

$$
Q_{DU(1)} + Q_{DL(1)} = \left(
\begin{array}{ccccccc|ccccccc}
1 & 1 & & & & & & & & & & & & \\
 & 10 & 1 & & & & & & & & & & & \\
 & & 1 & & & & & & & & & & & \\
1 & & & & 1 & 1 & & & & & & & & \\
 & 1 & & & & 10 & 1 & & & & & & & \\
 & & 1 & & & & 1 & & & & & & & \\
 & & & 1 & & & & & & & & & & \\
\hline
 & & & & & & & 1 & 1 & & & & & \\
 & & & & & & & & 10 & 1 & & & & \\
 & & & & & & & & & 1 & & & & \\
 & & & & & & & 1 & & & & 1 & 1 & \\
 & & & & & & & & 1 & & & & 10 & 1 \\
 & & & & & & & & & 1 & & & & 1 \\
 & & & & & & & & & & 1 & & & \\
\end{array}
\right)
$$

17

Block iterative methods for Kronecker products

Let Q be *irreducible* and split at level l as:

$$\boxed{Q = Q_O + Q_D = Q_{U(l)} + Q_{L(l)} + Q_{DU(l)} + Q_{DL(l)} + Q_D = M - N,}$$

where M is nonsingular (i.e., M^{-1} exists).
Then:

- power
- block Jacobi over-relaxation (BJOR)
- block successive over-relaxation (BSOR)

methods are based on different splittings of Q, and each satisfies

$$\boxed{\pi_{(m+1)}M = \pi_{(m)}N \quad \text{for} \quad m = 0, 1, \dots}$$

with sequence of approximations $\pi_{(m+1)}$ to π, where

- $\pi_{(0)} > 0$ is initial approximation such that $\pi_{(0)}e = 1$
- $T = NM^{-1}$ is iteration matrix.

Block iterative methods for Kronecker products (continued)

Splittings corresponding to power, BJOR, and (forward) BSOR methods are:

$$
\begin{aligned}
M^{Power} &= -\alpha I \\
&\quad N^{Power} = -\alpha(I + Q/\alpha) \\
M^{BJOR} &= (Q_D + Q_{DU(l)} + Q_{DL(l)})/\omega \\
&\quad N^{BJOR} = (1-\omega)(Q_D + Q_{DU(l)} + Q_{DL(l)})/\omega - Q_{U(l)} - Q_{L(l)} \\
M^{BSOR} &= (Q_D + Q_{DU(l)} + Q_{DL(l)})/\omega + Q_{U(l)} \\
&\quad N^{BSOR} = (1-\omega)(Q_D + Q_{DU(l)} + Q_{DL(l)})/\omega - Q_{L(l)},
\end{aligned}
$$

where

> $\alpha \in [\max_{s \in \mathcal{S}} |q_D(s,s)|, \infty)$: uniformization parameter of Power
> $\omega \in (0, 2)$: relaxation parameter of BJOR and BSOR.

Block iterative methods for Kronecker products (continued)

Point versus block methods

- Power works at level $l = H$ since it is point method
- BJOR and BSOR reduce to block Jacobi (BJacobi) and block Gauss-Seidel (BGS) for $\omega = 1$
- BJOR and BSOR become (point) JOR and (point) SOR for $l = H$.

Convergence

- Since Q is singular and assumed to be irreducible, $\rho(T) = 1$.
- In order to ensure convergence,
  T should not have other eigenvalues with magnitude one.
- For converging approximations, magnitude of eigenvalue of T closest to one determines rate of convergence.

Power

$$\pi_{(m+1)} = \pi_{(m)} + \pi_{(m)} Q_D / \alpha + \pi_{(m)} Q_O / \alpha.$$

Block iterative methods for Kronecker products (continued)

BJOR

$$\pi_{(m+1)}(Q_D + Q_{DU(l)} + Q_{DL(l)}) =$$
$$(1 - \omega)\pi_{(m)} Q_D + (1 - \omega)\pi_{(m)} Q_{DU(l)} + (1 - \omega)\pi_{(m)} Q_{DL(l)} - \omega \pi_{(m)} Q_{U(l)} - \omega \pi_{(m)} Q_{L(l)}.$$

$\sqrt{b_l}$ independent, ns linear systems each of order o_l and nonzero right-hand side

- If there is space:
  - ▲ Generate and factorize in sparse storage ns blocks:
    $Q((i_1, i_2, \ldots, i_l), (i_1, i_2, \ldots, i_l)) = \sum_{k=1}^{K} \left(\prod_{h=1}^{l} q_k^{(h)}(i_h, i_h) \right) \left(\bigotimes_{h=l+1}^{H} Q_k^{(h)} \right)$
    $+ Q_D((i_1, i_2, \ldots, i_l), (i_1, i_2, \ldots, i_l))$ for $(i_1, i_2, \ldots, i_l) \in \times_{h=1}^{l} \mathcal{S}^{(h)}$
    along the diagonal of $(Q_D + Q_{DU(l)} + Q_{DL(l)})$ at outset.
  - ▲ Solve the $|\times_{h=1}^{l} \mathcal{S}^{(h)}| = \sqrt{b_l}$ systems directly at each iteration.
- Otherwise, use (block) iterative method, such as BJOR,
  since off-diagonal parts of diagonal blocks given by $\sum_{k=1}^{K} \left(\prod_{h=1}^{l} q_k^{(h)}(i_h, i_h) \right) \left(\bigotimes_{h=l+1}^{H} Q_k^{(h)} \right)$ are
  sums of Kronecker products.

Block iterative methods for Kronecker products (continued)

BSOR

$$\pi_{(m+1)}(Q_D + Q_{DU(l)} + Q_{DL(l)} + \omega Q_{U(l)}) =$$
$$(1-\omega)\pi_{(m)}Q_D + (1-\omega)\pi_{(m)}Q_{DU(l)} + (1-\omega)\pi_{(m)}Q_{DL(l)} - \omega\pi_{(m)}Q_{L(l)}.$$

Block upper-triangular linear system with $\sqrt{b_l}$ blocks of order o_l along diagonal of ns coefficient matrix $(Q_D + Q_{DU(l)} + Q_{DL(l)} + \omega Q_{U(l)})$ and nonzero right-hand side.

- Recursive algorithm is given for ns linear system with lower-triangular coefficient matrix in the form of sum of Kronecker products and nonzero right-hand side [Uysal-Dayar'98]. Such a system arises in backward SOR. A version of the same algorithm for backward BSOR is also discussed.
- Iterative block upper-triangular solution algorithm for BSOR is also possible [Buchholz-Dayar'04a] and block row-oriented version is preferable in the presence of functional transitions.

Block iterative methods for Kronecker products (continued)

ALGORITHM 1. *Iterative block upper-triangular solution at level l for MCs based on Kronecker products*

$b = (1-\omega)\pi_{(m)}Q_D + (1-\omega)\pi_{(m)}Q_{DU(l)} + (1-\omega)\pi_{(m)}Q_{DL(l)} - \omega\pi_{(m)}Q_{L(l)};$
For row of blocks $(i_1, i_2, \ldots, i_l) = (1, 1, \ldots, 1)$ to $(n_1, n_2, \ldots, n_l)$ lexicographically,
    Solve $\pi_{(m+1)}((i_1, i_2, \ldots, i_l))Q((i_1, i_2, \ldots, i_l), (i_1, i_2, \ldots, i_l)) = b((i_1, i_2, \ldots, i_l));$
    For column of blocks $(j_1, j_2, \ldots, j_l) > (i_1, i_2, \ldots, i_l),$
      $b((j_1, j_2, \ldots, j_l)) = b((j_1, j_2, \ldots, j_l))$
        $-\omega\pi_{(m+1)}((i_1, i_2, \ldots, i_l))Q_{U(l)}((i_1, i_2, \ldots, i_l), (j_1, j_2, \ldots, j_l)).$

- In BSOR, ns diagonal blocks $Q((i_1, i_2, \ldots, i_l), (i_1, i_2, \ldots, i_l))$ must be solved in lexicographical order.
- After each block is solved for unknown subvector $\pi_{(m+1)}((i_1, i_2, \ldots, i_l))$,
  b is updated by multiplying computed subvector with corresponding row of blocks above diagonal.
- BSOR at level l reduces to SOR if $Q_{DL(l)} = 0$.

Block iterative methods for Kronecker products (continued)

■ Block iterative solvers, sometimes called two-level iterative solvers, have still not been incorporated into most analysis packages based on Kronecker representations although they are shown to be more effective than point solvers on many test cases [Buchholz-Dayar'04a, Uysal-Dayar'98].
■ To the contrary of block partitionings considered for sparse MCs [Dayar-Stewart'00], block partitionings of Kronecker products are nested and recursive due to lexicographical ordering of states. Hence, there tends to be more common structure among diagonal blocks of a MC expressed as sum of Kronecker products.

   ▲ Diagonal blocks having identical off-diagonal parts and diagonals which differ by multiple of identity can share and work with factorization of only one diagonal block [Buchholz-Dayar'04a]. This saves not only from time spent for factorization of diagonal blocks at the outset, but also from space.
   ▲ Three-level version of BSOR can be considered for MCs based on Kronecker products in which diagonal blocks that are too large to be factorized are solved using BSOR [Buchholz-Dayar'04a, Gusak-Dayar'01].

■ One can alter nonzero structure of underlying MC of Kronecker representation by reordering factors and states of factors so as to make it more suitable for block iterative methods. Power and JOR methods will not benefit from such reordering.

Multilevel methods

Multilevel methods

■ Aggregation-disaggregation steps are coupled with various iterative methods for MCs based on Kronecker products to accelerate convergence [Buchholz'94a, Buchholz'99bce].
■ Iterative aggregation-disaggregation (IAD) method for MCs based on Kronecker products and its adaptive version,
which analyzes aggregated systems for those parts where error is estimated to be high, are proposed [Buchholz'97, Buchholz'99a].
■ Adaptive IAD method is improved through recursive definition and called multilevel (ML) [Buchholz'00a].

The simple multilevel method for Kronecker products

Let:

- $\mathcal{S}_{(l)} = \times_{h=l+1}^{H} \mathcal{S}^{(h)}$ for $l = 0, 1, \ldots, H$
- mapping $f_{(l)} : \mathcal{S}_{(l)} \longrightarrow \mathcal{S}_{(l+1)}$ represent aggregation of dimension $(l+1)$ (i.e., the state space $S^{(l+1)}$) so that
  states in $\mathcal{S}_{(l)}$ are mapped to states in $\mathcal{S}_{(l+1)}$; note:
    - ▲ $\mathcal{S}_{(0)} = \mathcal{S}$
    - ▲ $\mathcal{S}_{(H)} = \{1\}$.
- aggregated CTMCs $\tilde{Q}_{(m,l)}$ with state spaces $\mathcal{S}_{(l)}$ be defined at levels $l = 1, 2, \ldots, H$ with $\tilde{Q}_{(m,0)} = Q$ for iteration m
- Power be used as *smoother* (or *accelerator*):
    - ▲ $\eta_{(m,l)}$ times before aggregation
    - ▲ $\nu_{(m,l)}$ times after disaggregation
    with $\alpha_{(m,l)} \in [\max_{s_{(l)} \in \mathcal{S}_{(l)}} |\tilde{q}_{(m,l)}(s_{(l)}, s_{(l)})|, \infty)$ at level l for iteration m.

The simple multilevel method for Kronecker products (continued)

Then ML iteration matrix at level l for iteration m is given by:

$$T_{(m,l)}^{ML} = (I + \tilde{Q}_{(m,l)}/\alpha_{(m,l)})^{\eta_{(m,l)}} R_{(l)} T_{(m,l+1)}^{ML} P_{x_{(m,l)}} (I + \tilde{Q}_{(m,l)}/\alpha_{(m,l)})^{\nu_{(m,l)}}$$

and satisfies $\boxed{\pi_{(m+1,l)} = \pi_{(m,l)} T_{(m,l)}^{ML} \quad \text{for} \quad m = 0, 1, \ldots,}$ where

$$x_{(m,l)} = \pi_{(m,l)} (I + \tilde{Q}_{(m,l)}/\alpha_{(m,l)})^{\eta_{(m,l)}}$$

$$r_{(l)}(s_{(l)}, s_{(l+1)}) = \begin{cases} 1 & \text{if } f_{(l)}(s_{(l)}) = s_{(l+1)} \\ 0 & \text{otherwise} \end{cases} \quad \text{for } s_{(l)} \in \mathcal{S}_{(l)} \text{ and } s_{(l+1)} \in \mathcal{S}_{(l+1)}$$

$$p_{x_{(m,l)}}(s_{(l+1)}, s_{(l)}) = \begin{cases} \dfrac{x_{(m,l)}(s_{(l)})}{\sum_{s_{(l)} \in \mathcal{S}_{(l)}, f_{(l)}(s_{(l)}) = s_{(l+1)}} x_{(m,l)}(s_{(l)})} & \text{if } f_{(l)}(s_{(l)}) = s_{(l+1)} \\ 0 & \text{otherwise} \end{cases}$$

for $s_{(l+1)} \in \mathcal{S}_{(l+1)}$ and $s_{(l)} \in \mathcal{S}_{(l)}$,

The simple multilevel method for Kronecker products (continued)

$$\pi_{(m,l+1)} = x_{(m,l)} R_{(l)} \quad \text{and} \quad \tilde{Q}_{(m,l+1)} = P_{x_{(m,l)}} \tilde{Q}_{(m,l)} R_{(l)}.$$

At iteration m, recursion ends and backtracking starts when:

- $\tilde{Q}_{(m,l+1)}$ is the last aggregated CTMC and solved exactly to give

$$\boxed{T_{(m,l+1)} = e\pi_{(m+1,l+1)},}$$

  where $\pi_{(m+1,l+1)} \tilde{Q}_{(m,l+1)} = 0$ and $\pi_{(m+1,l+1)} e = 1$.

Level to end recursion depends on available memory since there must be space to store and factorize $\tilde{Q}_{(m,l+1)}$ at that level.

When $\pi_{(0,0)} > 0$:

- aggregated CTMCs $\tilde{Q}_{(m,l+1)}$ are *irreducible* [Buchholz'00a]
- ML method has been observed to converge if a sufficient number of smoothings are performed to improve $\pi_{(m,l)}$ at each level.

The simple multilevel method for Kronecker products (continued)

To the contrary of block iterative methods, ML iteration matrix changes from iteration to iteration $\Rightarrow$ method is *non-stationary*.

- $(|\mathcal{S}_{(l)}| \times |\mathcal{S}_{(l+1)}|)$ *aggregation* operator, $R_{(l)}$, is:
  - ▲ constant
  - ▲ need not be stored since it is defined by $f_{(l)}$.

- At level l, $|\mathcal{S}_{(l)}| = \prod_{h=l+1}^{H} n_l$ states represented by $(H-l)$-tuples are mapped to the $|\mathcal{S}_{(l+1)}| = \prod_{h=l+2}^{H} n_l$ states represented by $(H-l-1)$-tuples by aggregating the leading dimension $\mathcal{S}^{(l+1)}$ in $\mathcal{S}_{(l)}$.
  - ▲ Corresponds to aggregation based on contiguous and non-interleaved block partitioning if states in $\mathcal{S}_{(l)}$ were ordered anti-lexicographically.

- $(|\mathcal{S}_{(l+1)}| \times |\mathcal{S}_{(l)}|)$ *disaggregation* operator, $P_{x_{(m,l)}}$:
  - ▲ depends on $x_{(m,l)}$
  - ▲ has the nonzero structure of $R_{(l)}^T$.

The simple multilevel method for Kronecker products (continued)

- $P_{x_{(m,l)}}$ can be stored in a vector of length $|\mathcal{S}_{(l)}|$ since it has one nonzero per column by definition.
- These vectors amount to total storage of $\boxed{\sum_{l=0}^{H-1} \prod_{h=l+1}^{H} n_h}$ floating-point values if recursion terminates at level H.

$\tilde{Q}_{(m,l+1)}$ can be expressed as a sum of Kronecker products [Buchholz'00a] using:

- at most K vectors of length $|\mathcal{S}_{(l+1)}|$
- matrices corresponding to factors $(l+2)$ through H.

Element $s_{(l+1)}$ of vector corresponding to kth term in Kronecker representation at level $(l+1)$ for iteration m is:

$$a_{(m,l+1),k}(s_{(l+1)}) = \frac{\left(\sum_{s_{(l)} \in \mathcal{S}_{(l)}, f_{(l)}(s_{(l)}) = s_{(l+1)}} x_{(m,l)}(s_{(l)})\, a_{(m,l),k}(s_{(l)})\, (e_{s_{(l)}(l+1)}^T Q_k^{(l+1)} e) \right)}{\pi_{(m,l+1)}(s_{(l+1)})}$$

$$\text{for } s_{(l+1)} \in \mathcal{S}_{(l+1)} \text{ and } k = 1, 2, \ldots, K,$$

where $a_{(m,0),k} = e$, $s_{(l)}(l+1) \in \mathcal{S}^{(l+1)}$, and $e_{s_{(l)}(l+1)}$ is $s_{(l)}(l+1)$st column of I.

The simple multilevel method for Kronecker products (continued)

$$\boxed{\tilde{Q}_{(m,l+1)} = \sum_{k=1}^{K} \text{diag}(a_{(m,l+1),k}) \bigotimes_{h=l+2}^{H} Q_k^{(h)} - \sum_{k=1}^{K} \text{diag}(a_{(m,l+1),k}) \bigotimes_{h=l+2}^{H} \text{diag}(Q_k^{(h)} e)}$$

- Second summation returns diagonal matrix which sums rows of $\tilde{Q}_{(m,l+1)}$ to 0.
- No need to store $a_{(m,0),k} = e$ for $k = 1, 2, \ldots, K$ at level 0.
- If recursion ends at level H, then $\tilde{Q}_{(m,H)}$ is (1×1) CTMC equal to 0, and need not be stored since its steady-state vector is 1.

No need to store $a_{(m,l+1),k} = e$ for those k which:

- either have single $Q_k^{(h)} \neq I$ for $h = 1, 2, \ldots, H$,
- or have all $Q_k^{(h)} = I$ for $h = l + 2, \ldots, H$.

K vectors at particular level have same length,
but vary in length from $\prod_{h=2}^{H} n_h$ at level 1 to n_H at level $(H-1)$,
implying a storage requirement of at most $K \sum_{l=1}^{H-1} \prod_{h=l+1}^{H} n_h$ floating-point values to facilitate the Kronecker representation of the aggregated CTMCs.

Grouping of factors will further reduce storage requirement for vectors.

Example (continued)

Consider the 4-dimensional problem with

$$\pi_{(0,0)} = e/16, \quad \alpha_{(0,0)} = 21, \quad \text{and} \quad \eta_{(0,0)} = \nu_{(0,0)} = 1.$$

Then, $\boxed{x_{(0,0)} = \pi_{(0,0)}(I + \tilde{Q}_{(0,0)}/21)}$ yields

$$x_{(0,0)} = (19\ 11\ 21\ 13\ 27\ 19\ 29\ 21\ 21\ 13\ 23\ 15\ 29\ 21\ 31\ 23)/336$$

$$R_{(0)} = \begin{pmatrix}
1 & & & & & & & \\
& 1 & & & & & & \\
& & 1 & & & & & \\
& & & 1 & & & & \\
& & & & 1 & & & \\
& & & & & 1 & & \\
& & & & & & 1 & \\
1 & & & & & & & \\
& 1 & & & & & & \\
& & 1 & & & & & \\
& & & 1 & & & & \\
& & & & 1 & & & \\
& & & & & 1 & & \\
& & & & & & 1 & \\
& & & & & & & 1
\end{pmatrix}$$

Example (continued)

$$P_{x_{(0,0)}} = \begin{pmatrix}
\frac{19}{40} & & & & & & & & \frac{21}{40} & & & & & & & \\
& \frac{11}{24} & & & & & & & & \frac{13}{24} & & & & & & \\
& & \frac{21}{44} & & & & & & & & \frac{23}{44} & & & & & \\
& & & \frac{13}{28} & & & & & & & & \frac{15}{28} & & & & \\
& & & & \frac{27}{56} & & & & & & & & \frac{29}{56} & & & \\
& & & & & \frac{19}{40} & & & & & & & & \frac{21}{40} & & \\
& & & & & & \frac{29}{60} & & & & & & & & \frac{31}{60} & \\
& & & & & & & \frac{21}{44} & & & & & & & & \frac{23}{44}
\end{pmatrix}.$$

- 16 states represented by 4-tuples in $\mathcal{S}_{(0)} = \mathcal{S}$ are mapped to 8 states represented by 3-tuples in $\mathcal{S}_{(1)}$.
- For instance, states $(1,1,1,1)$ and $(2,1,1,1)$ are mapped to $(1,1,1)$, whereas states $(1,1,1,2)$ and $(2,1,1,2)$ are mapped to $(1,1,2)$.

Example (continued)

Using $R_{(0)}$, we obtain starting approximation at level 1 as

$$\pi_{(0,1)} = (40\ 24\ 44\ 28\ 56\ 40\ 60\ 44)/336.$$

7 vectors used to represent aggregated CTMC at level 1 are computed as

$$a_{(0,1),1} = (19/40\ 11/24\ 21/44\ 13/28\ 27/56\ 19/40\ 29/60\ 21/44),$$
$$a_{(0,1),2} = a_{(0,1),3} = a_{(0,1),4} = a_{(0,1),5} = e,$$
$$a_{(0,1),6} = (19/40\ 11/24\ 21/44\ 13/28\ 27/56\ 19/40\ 29/60\ 21/44),$$
$$a_{(0,1),7} = (210/44\ 130/24\ 230/44\ 150/28\ 290/56\ 210/40\ 310/60\ 230/44).$$

and aggregated CTMC is expressed as

$$\begin{aligned}
\tilde{Q}_{0,1} &= P_{x_{(0,0)}} \tilde{Q}_{(0,0)} R_{(0)} \\
&= \sum_{k=1}^{7} \mathsf{diag}(a_{(0,1),k}) \bigotimes_{h=2}^{4} Q_k^{(h)} - \sum_{k=1}^{7} \mathsf{diag}(a_{(0,1),k}) \bigotimes_{h=2}^{4} \mathsf{diag}(Q_k^{(h)} e).
\end{aligned}$$

Example (continued)

We may very well set $a_{(0,1),1} = e$ as suggested before, because effect of $a_{(0,1),1}$
in first term of first summation will be to diagonal of $\tilde{Q}_{0,1}$
(since $Q_1^{(2)} = Q_1^{(3)} = Q_2^{(4)} = I$), but this effect will be cancelled by first term of second summation
(since $\mathsf{diag}(Q_1^{(2)} e) = \mathsf{diag}(Q_1^{(3)} e) = \mathsf{diag}(Q_2^{(4)} e) = I$). Hence:

1	1	1	1	2	2	2	2
1	1	2	2	1	1	2	2
1	2	1	2	1	2	1	2

$$\tilde{Q}_{(0,1)} = \begin{pmatrix}
-\frac{290}{40} & \frac{40}{40} & \frac{40}{40} & & \frac{210}{40} & & & \\
& -\frac{394}{24} & \frac{240}{24} & \frac{24}{24} & & \frac{130}{24} & & \\
\frac{210}{44} & & -\frac{484}{44} & \frac{44}{44} & & & \frac{230}{44} & \\
& \frac{130}{28} & & -\frac{280}{28} & & & & \frac{150}{28} \\
\frac{56}{56} & & & & -\frac{168}{56} & \frac{56}{56} & \frac{56}{56} & \\
& \frac{40}{40} & & & & -\frac{480}{40} & \frac{400}{40} & \frac{40}{40} \\
& & \frac{60}{60} & & \frac{290}{60} & & -\frac{410}{60} & \frac{60}{60} \\
& & & \frac{44}{44} & & \frac{210}{44} & & -\frac{254}{44}
\end{pmatrix}.$$

26

A class of multilevel methods for Kronecker products

ML method discussed follows a V-cycle at each iteration and uses Power as smoother.

- State spaces $\mathcal{S}^{(h)}$ are aggregated according to fixed ordering $h = 1, 2, \ldots, H$.
- To the contrary of ML method for sparse MCs [Horton-Leutenegger'94]:
  - ▲ definition of aggregated state spaces follows naturally from Kronecker representation
  - ▲ aggregated CTMCs can also be represented using Kronecker products.

Class of ML methods in [Buchholz-Dayar'04b] are:

- capable of using JOR and SOR as smoothers
- performing W- and F-cycles inspired by multigrid
- aggregating state spaces in cyclic and adaptive orderings.

QEST 2009   13 September 2009 − 52 / 57

A class of multilevel methods for Kronecker products (continued)

Numerical experiments proved ML methods to be very strong, robust, and scalable solvers for MCs based on Kronecker products.

- Convergence properties of ML methods are discussed in [Buchholz-Dayar'07].
- It is not clear how behavior would be affected if block iterative methods are used as smoothers.

  BJOR and BSOR should normally not use a direct method for the solution of diagonal blocks when employed as smoothers with ML method, since aggregated CTMC at each level changes from iteration to iteration and factorization may be too time consuming to offset.

Efficient algorithm that finds nearly completely decomposable (NCD) partitioning of $\mathcal{S}$ in the presence of functional transitions for user specified decomposability parameter is given [Gusak-Dayar-Fourneau'01]. Since IAD using NCD partitionings has certain rate of convergence guarantees, the algorithm may be useful in the context of ML methods to determine loosely coupled dimensions to be aggregated first in given iteration.

QEST 2009   13 September 2009 − 53 / 57

Preconditioned projection methods   54 / 57

Preconditioned projection methods

Projection (or *Krylov subspace*) methods are non-stationary iterative methods in which approximate solutions satisfying various constraints are extracted from small dimensional subspaces.

- Basic operation is vector-Kronecker product multiplication.
- Compared to block iterative methods, they require a larger number of supplementary vectors of length n.
- Should be used with preconditioners to result in effective solvers.

At each iteration of preconditioned projection method, row residual vector, r, is used as right-hand side of linear system:

$$zM = r$$

to compute preconditioned row residual vector, z, where M is preconditioning matrix.

QEST 2009   13 September 2009 − 54 / 57

Preconditioned projection methods (continued)

Objective of preconditioning is:

> to improve error in approximate solution vector at that iteration.

M should approximate Q in some sense,
yet solution of linear systems involving M should be cheap.

- To result as effective solvers, projection methods for sparse MCs should be used with preconditioners, such as those based on incomplete LU (ILU) factorizations [Dayar-Stewart'00].
- It is still not clear how one can devise ILU-type preconditioners for MCs that are based on Kronecker products.

Preconditioners for Kronecker structured MCs:

- truncated Neumann series [Stewart'94, Stewart-Atif-Plateau'95]
- cheap and separable preconditioner [Buchholz'99c]
- circulant preconditioners for a specific class of problems [CC00]
- Kronecker product approximate preconditioner [Langville-Stewart'04abc].

Preconditioned projection methods (continued)

[Buchholz'99ce, Langville-Stewart'04bc, Stewart-Atif-Plateau'95]: room for research regarding effective preconditioners for MCs based on Kronecker products.

BSOR preconditioner

Block iterative methods are *preconditioned power methods* for which preconditioning matrix is M.

- BSOR preconditioner using M is proposed [Buchholz-Dayar'05].
- To the contrary of BSOR preconditioner for sparse MCs, BSOR preconditioner based on Kronecker products has rich structure induced by lexicographical ordering of states.

Two-level BSOR preconditioned projection methods in which diagonal blocks are solved exactly are competitive with block iterative methods and ML methods.

- JOR, BJOR, and SOR preconditioners can be compared with existing preconditoners for MCs based on Kronecker products.
- ML methods are candidates for preconditioning projection methods.

Conclusion

- MCs based on Kronecker products have rich structure, which is *nested* and *recursive*.
- Preprocessing techniques that take advantage of this rich structure to expedite analysis are:

  - ▲ reordering
  - ▲ grouping
  - ▲ lumping.

- Software packages working with Kronecker products should include:

  - ▲ block iterative methods based on splittings
  - ▲ multilevel methods
  - ▲ projection methods preconditioned with block iterative methods.

- Implementation requires *intricate programming* with dynamically allocated, relatively complex data structures, needing time, careful testing, and tuning.