Analyzing Markov Chains using Kronecker Products

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

9 June 2014

Outline

- Background
  - Kronecker representation of Q
  - An example
  - Vector-Kronecker product multiplication
- Preprocessing
  - Reordering and grouping
  - Lumping
- Iterative methods
  - Splitting the smaller matrices
  - Example (continued)
  - Block iterative methods for Kronecker products
- Multilevel methods
  - A simple multilevel method for Kronecker products
  - Example (continued)
  - A class of multilevel methods for Kronecker products
- 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

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

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

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

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

$$\boxed{\pi_t = \pi_0 e^{Qt} = \pi_0 e^{-\Gamma t} \sum_{k=0}^{\infty} \frac{(\Gamma t)^k}{k!} \left(I + \frac{1}{\Gamma} Q\right)^k, \Gamma = \max_i |q(i,i)|}$$

[Grassmann'77, Gross-Miller'84]

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

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

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

Background (continued)

- In Kronecker based approach [Dayar'12], 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].
- π_t can be computed through *uniformization* using vector-Kronecker product multiplications [Buchholz'94a].
- π 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.

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 with $A = [a(i_A, j_A)]$ is

$$\boxed{A \otimes B = [a(i_A, j_A)B].}$$

Or more formally, given $A \in \mathbb{R}^{n_A \times m_A}$ and $B \in \mathbb{R}^{n_B \times m_B}$, $A \otimes B$ yields the (rectangular) matrix $C \in \mathbb{R}^{n_A n_B \times m_A m_B}$ whose entries satisfy

$$\boxed{c(i_C, j_C) = a(i_A, j_A)b(i_B, j_B) \quad \text{with} \quad i_C = i_A n_B + i_B \text{ and } j_C = j_A m_B + j_B,}$$

$$\boxed{(i_A, j_A) \in \{0, \ldots, n_A - 1\} \times \{0, \ldots, m_A - 1\},}$$

$$\boxed{(i_B, j_B) \in \{0, \ldots, n_B - 1\} \times \{0, \ldots, m_B - 1\},}$$

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

Background (continued)

- In a 2-dimensional representation:
  - *row indices* of $C \in \{0, \ldots, n_A - 1\} \times \{0, \ldots, n_B - 1\}$
  - *column indices* of $C \in \{0, \ldots, m_A - 1\} \times \{0, \ldots, m_B - 1\}$.
- Ordering of rows and columns of C is *lexicographical*, since

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

- 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 $X^{(h)} \in \mathcal{S}^{(h)} = \{0, \ldots, n_h - 1\}$ for $h = 1, \ldots, H$
  - $X \in \mathbb{R}^{n \times n}$ with $\boxed{n = \prod_{h=1}^{H} n_h.}$

Background (continued)

H-dimensional state space representation

- Ordered H-dimensional tuples

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

  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:

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

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 $k = 1, \ldots, K$ and $h = 1, \ldots, H$
- diag: *diagonal matrix* which has its vector argument along its diagonal.

Kronecker representation of Q (continued)

- If row/column indices of $Q_k^{(h)} \in \mathcal{S}^{(h)} = \{0, \ldots, n_h - 1\}$ for $k = 1, \ldots, K$ and $h = 1, \ldots, H$, then *H-dimensional state space* of Q is given by:

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

- $|\mathcal{S}| = \prod_{h=1}^{H} |\mathcal{S}^{(h)}| = \prod_{h=1}^{H} n_h = n.$
- *One-dimensional value* of state $\mathbf{i} \in \mathcal{S}$ corresponding to $(i_1, \ldots, i_H)$, where $i_h \in \mathcal{S}^{(h)}$ for $h = 1, \ldots, H$, is given by:

$$\boxed{i = \sum_{h=1}^{H} i_h \prod_{l=h+1}^{H} n_l.}$$

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

Kronecker representation of Q (continued)

Space complexity

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

  $nz_{Q^{(h)}}$: sum of # of nonzeros in $Q_k^{(h)}$ for $k = 1, \ldots, K$.
- Q_D can also be expressed as a sum of Kronecker products:

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

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

Kronecker representation of Q (continued)

- At level $l = 0, \ldots, H$, 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 is # of blocks at level l, o_l is order of blocks at level l.
- There are $\sqrt{b_l}$ blocks each of order o_l along the diagonal of Q.
- Block $((i_1, \ldots, i_l), (j_1, \ldots, j_l))$ of Q at level $l = 0, \ldots, H$:

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

- $Q_D((i_1, \ldots, i_l), (j_1, \ldots, j_l))$ is block $((i_1, \ldots, i_l), (j_1, \ldots, j_l))$ of Q_D.
  $Q_D((i_1, \ldots, i_l), (j_1, \ldots, j_l)) = 0$
  if $(i_1, \ldots, i_l) \neq (j_1, \ldots, j_l)$, meaning it is off-diagonal block at level l.
- $l = 0$ yields (block) Q and $l = H$ yields scalar (block):

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

An example

Consider the following matrices for a 3-dimensional problem (with 2, 3, and 2 states, respectively) having 4 terms of Kronecker products:

$$Q_2^{(1)} = Q_3^{(1)} = I_2\,, \quad Q_1^{(1)} = \begin{pmatrix} & \lambda_1 \\ \mu_1 & \end{pmatrix}, \quad Q_4^{(1)} = \begin{pmatrix} & \\ \mu & \end{pmatrix},$$

$$Q_1^{(2)} = Q_3^{(2)} = I_3\,, \quad Q_2^{(2)} = \begin{pmatrix} & \lambda_2 & \\ \mu_2 & & \lambda_2 \\ & \mu_2 & \end{pmatrix}, \quad Q_4^{(2)} = \begin{pmatrix} & & \\ & & \\ 1 & & \end{pmatrix},$$

$$Q_1^{(3)} = Q_2^{(3)} = I_2\,, \quad Q_3^{(3)} = \begin{pmatrix} & \lambda_3 \\ \mu_3 & \end{pmatrix}, \quad Q_4^{(3)} = \begin{pmatrix} & \\ 1 & \end{pmatrix}.$$

Then,

$$\boxed{Q = \sum_{k=1}^{4} \bigotimes_{h=1}^{3} Q_k^{(h)} + Q_D.}$$

An example (continued)

$$
Q = \begin{array}{c}
\\ 0\,0\,0 \\ 0\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 0\,2\,0 \\ 0\,2\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \\ 1\,2\,0 \\ 1\,2\,1
\end{array}
\begin{array}{c}
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 2 & 2 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1
\end{array} \\
\begin{pmatrix}
* & \lambda_3 & \lambda_2 & & & & \lambda_1 & & & & & \\
\mu_3 & * & & \lambda_2 & & & & \lambda_1 & & & & \\
\mu_2 & & * & \lambda_3 & \lambda_2 & & & & \lambda_1 & & & \\
 & \mu_2 & \mu_3 & * & & \lambda_2 & & & & \lambda_1 & & \\
 & & \mu_2 & & * & \lambda_3 & & & & & \lambda_1 & \\
 & & & \mu_2 & \mu_3 & * & & & & & & \lambda_1 \\
\mu_1 & & & & & & * & \lambda_3 & \lambda_2 & & & \\
 & \mu_1 & & & & & \mu_3 & * & & \lambda_2 & & \\
 & & \mu_1 & & & & \mu_2 & & * & \lambda_3 & \lambda_2 & \\
 & & & \mu_1 & & & & \mu_2 & \mu_3 & * & & \lambda_2 \\
 & & & & \mu_1 & & & & \mu_2 & & * & \lambda_3 \\
\mu & & & & & \mu_1 & & & & \mu_2 & \mu_3 & *
\end{pmatrix}
\end{array}
$$

- # of floating-point values stored in Kronecker representation is 11 for matrices and 12 for diagonal, thus totaling 23; whereas, it is 53 for flat representation.
- Discrepancy between Kronecker and flat representations becomes substantial for larger values of the state space size, n.

Vector-Kronecker product multiplication

At heart of all iterative solvers for sums of Kronecker products.

Left–oriented version: $\boxed{\mathbf{x}' = \mathbf{x} \bigotimes_{h=1}^{H} X^{(h)},}$

where $X^{(h)} \in \mathbb{R}^{n_h \times m_h}$ for $h = 1, \ldots, H$, is based on

$$\boxed{\bigotimes_{h=1}^{H} X^{(h)} = \prod_{h=1}^{H} I_{m_1} \otimes \cdots \otimes I_{m_{h-1}} \otimes X^{(h)} \otimes I_{n_{h+1}} \otimes \cdots \otimes I_{n_H},}$$

or more simply

$$\boxed{\bigotimes_{h=1}^{H} X^{(h)} = \prod_{h=1}^{H} \left(I_{\prod_{f=1}^{h-1} m_f} \otimes X^{(h)} \otimes I_{\prod_{f=h+1}^{H} n_f} \right)}$$

due to *compatibility* of Kronecker product with matrix multiplication [Fernandes-Plateau-Stewart'98a].

Vector-Kronecker product multiplication (continued)

- Left–multiplication of $\mathbf{x} \in \mathbb{R}^{1 \times \prod_{h=1}^{H} n_h}$ with $\bigotimes_{h=1}^{H} X^{(h)}$ yields a product vector whose length ranges from $\boxed{m_1 \prod_{h=2}^{H} n_h}$ to $\boxed{\prod_{h=1}^{H} m_h}$ during the course of the multiplication.
- The hth factor of the form:

$$\boxed{I_{\prod_{f=1}^{h-1} m_f} \otimes X^{(h)} \otimes I_{\prod_{f=h+1}^{H} n_f}}$$

  is a rectangular $(\prod_{f=1}^{h-1} m_f \prod_{f=h}^{H} n_f \times \prod_{f=1}^{h} m_f \prod_{f=h+1}^{H} n_f)$ block diagonal matrix having $\prod_{f=1}^{h-1} m_f$ diagonal blocks each of size $(n_h \prod_{f=h+1}^{H} n_f \times m_h \prod_{f=h+1}^{H} n_f)$.
- Furthermore, each of the diagonal blocks is an $(n_h \times m_h)$ block matrix, where each subblock is a diagonal matrix of order $\prod_{f=h+1}^{H} n_f$ with a particular entry of $X^{(h)}$ appearing along its diagonal $\prod_{f=h+1}^{H} n_f$ many times. It is this feature that is used in devising the algorithm.

Vector-Kronecker product multiplication (continued)

Algorithm for $\mathbf{x}' = \mathbf{x} \bigotimes_{h=1}^{H} X^{(h)}$

Copy $\mathbf{x}$ to $\mathbf{x}'$; $i_{left} = 1$; $i_{right} = \prod_{h=2}^{H} n_h$; $n_{H+1} = 1$;
For $h = 1$ to H,
    $base_i = 0$; $base_j = 0$;
    For $i_l = 0, \ldots, i_{left} - 1$,
        For $i_r = 0, \ldots, i_{right} - 1$,
            $index_i = base_i + i_r$;
            For $row = 0, \ldots, n_h - 1$,
                $\mathbf{z}(row) = \mathbf{x}'(index_i)$; $index_i = index_i + i_{right}$;
            $\mathbf{z}' = \mathbf{z} X^{(h)}$;
            $index_j = base_j + i_r$;
            For $col = 0, \ldots, m_h - 1$,
                $\mathbf{x}''(index_j) = \mathbf{z}'(col)$; $index_j = index_j + i_{right}$;
        $base_i = base_i + n_h i_{right}$; $base_j = base_j + m_h i_{right}$;
    $i_{left} = i_{left} m_h$; $i_{right} = i_{right} / n_{h+1}$;
    Copy $\mathbf{x}''$ to $\mathbf{x}'$.

Vector-Kronecker product multiplication (continued)

Two temporary floating-point vectors, $\mathbf{z}$ and $\mathbf{z}'$,
with lengths $\max_h(n_h)$ and $\max_h(m_h)$, respectively.

Two floating-point vectors, $\mathbf{x}'$ and $\mathbf{x}''$,
of length $\max_{h=0,\ldots,H}(\prod_{f=1}^{h} m_f \prod_{f=h+1}^{H} n_f)$ to compute and return the result.

Time complexity

Complexity of a vector multiplication with Q_O consisting of K Kronecker product terms when $m_h = n_h$ for $h = 1, \ldots, H$ is given by:

$$K \prod_{h=1}^{H} n_h + 2 \sum_{k=1}^{K} \prod_{h=1}^{H} n_h \sum_{f=1}^{H} nz_{Q_k^{(f)}}/n_f = K \prod_{h=1}^{H} n_h + 2 \prod_{h=1}^{H} n_h \sum_{f=1}^{H} \left(\sum_{k=1}^{K} nz_{Q_k^{(f)}} \right) / n_f$$

$$= n(K + 2 \sum_{h=1}^{H} nz_{Q^{(h)}}/n_h)$$

floating-point arithmetic operations [Fernandes-Plateau-Stewart'98a], where:
$nz_{Q_k^{(l)}}$: # of nonzeros in $Q_k^{(l)}$ for $k = 1, \ldots, K$ and $l = 1, \ldots, H$.

Preprocessing

To expedite the analysis of MCs based on Kronecker products, 3 techniques can be used to put Kronecker representation into more favorable form before solvers take over:

- 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 $\Rightarrow$ normally assume $K > 1$.

Make K as small as possible without changing H

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

Reordering and grouping

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 $Q_k^{(h)}$ for $k = 1, \ldots, K$ associated with renumbered state space $\mathcal{S}^{(h)}$.

- 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 $h = 1, \ldots, H$.

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 show that in some cases grouping may:

- decrease number of terms in the Kronecker representation.

Group as many factors as possible given available memory starting from highest indexed factor

$\Rightarrow$ flatter representation for diagonal blocks at a particular level, which is useful in certain iterative methods.

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.

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 $\mathcal{S} = \cup_l \mathcal{S}_l$ and $\mathcal{S}_l \cap \mathcal{S}_u = \emptyset$ for all $l \neq u$ if for all $\mathcal{S}_l \subset \mathcal{S}$ and all $\mathbf{i}, \mathbf{i}' \in \mathcal{S}_l$

$$\boxed{\sum_{\mathbf{j} \in \mathcal{S}_u} q(\mathbf{i}, \mathbf{j}) = \sum_{\mathbf{j} \in \mathcal{S}_u} q(\mathbf{i}', \mathbf{j}) \text{ for all } \mathcal{S}_u \subset \mathcal{S}.}$$

Q is said to be *exactly lumpable* with respect to a partitioning $\mathcal{S} = \cup_l \mathcal{S}_l$ and $\mathcal{S}_l \cap \mathcal{S}_u = \emptyset$ for all $l \neq u$ if for all $\mathcal{S}_l \subset \mathcal{S}$ and all $\mathbf{i}, \mathbf{i}' \in \mathcal{S}_l$

$$\boxed{\sum_{\mathbf{j} \in \mathcal{S}_u} q(\mathbf{j}, \mathbf{i}) = \sum_{\mathbf{j} \in \mathcal{S}_u} q(\mathbf{j}, \mathbf{i}') \text{ for all } \mathcal{S}_u \subset \mathcal{S}.}$$

- Ordinary lumpability refers to a partitioning of $\mathcal{S}$ 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 $\mathcal{S}$ 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 $h = 1, \ldots, H$ 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.
  - Performance measures of interest over $\mathcal{S}_{lumped}$ can be computed.

Lumping (continued)

- Lumpability can be investigated among $\mathcal{S}^{(h)}$ by considering 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 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.

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]:

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

$$\text{for} \quad k = 1, \ldots, K \quad \text{and} \quad h = 1, \ldots, H,$$

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

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 $l = 0, \ldots, H$ as

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

where

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

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

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

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:

$$Q_{U(l)} \geq 0,\ Q_{L(l)} \geq 0,\ Q_{DU(l)} \geq 0,\ Q_{DL(l)} \geq 0.$$

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

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

Example (continued)

Consider block partitioning of the 3-dimensional problem at level 1 for which:

- $l = 1$
- $b_1 = 4$
- Q is viewed as (2×2) block matrix with blocks of order $o_1 = 6$.

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

$$Q_{U(1)} + Q_{L(1)} = \left(\begin{array}{cccccc|cccccc} & & & & & & \lambda_1 & & & & & \\ & & & & & & & \lambda_1 & & & & \\ & & & & & & & & \lambda_1 & & & \\ & & & & & & & & & \lambda_1 & & \\ & & & & & & & & & & \lambda_1 & \\ & & & & & & & & & & & \lambda_1 \\ \hline \mu_1 & & & & & & & & & & & \\ & \mu_1 & & & & & & & & & & \\ & & \mu_1 & & & & & & & & & \\ & & & \mu_1 & & & & & & & & \\ & & & & \mu_1 & & & & & & & \\ \mu & & & & & \mu_1 & & & & & & \end{array}\right).$$

Example (continued)

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

$$Q_{DU(1)} = \sum_{k=1}^{4} D_k^{(1)} \otimes U_k^{(2)} \otimes Q_k^{(3)} + \sum_{k=1}^{4} D_k^{(1)} \otimes D_k^{(2)} \otimes U_k^{(3)},$$

$$Q_{DL(1)} = \sum_{k=1}^{4} D_k^{(1)} \otimes L_k^{(2)} \otimes Q_k^{(3)} + \sum_{k=1}^{4} D_k^{(1)} \otimes D_k^{(2)} \otimes L_k^{(3)}.$$

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

$$Q_{DU(1)} + Q_{DL(1)} = \left(\begin{array}{cccccc|cccccc} & \lambda_3 & \lambda_2 & & & & & & & & & \\ \mu_3 & & & \lambda_2 & & & & & & & & \\ \mu_2 & & & \lambda_3 & \lambda_2 & & & & & & & \\ & \mu_2 & \mu_3 & & & \lambda_2 & & & & & & \\ & & \mu_2 & & & \lambda_3 & & & & & & \\ & & & \mu_2 & \mu_3 & & & & & & & \\ \hline & & & & & & & \lambda_3 & \lambda_2 & & & \\ & & & & & & \mu_3 & & & \lambda_2 & & \\ & & & & & & \mu_2 & & & \lambda_3 & \lambda_2 & \\ & & & & & & & \mu_2 & \mu_3 & & & \lambda_2 \\ & & & & & & & & \mu_2 & & & \lambda_3 \\ & & & & & & & & & \mu_2 & \mu_3 & \end{array}\right).$$

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

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

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

- $\pi_{(0)} > 0$ is initial approximation such that $\pi_{(0)}\mathbf{e} = 1$
- $T = NM^{-1}$ 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:
    $Q((i_1,\ldots,i_l),(i_1,\ldots,i_l)) = \sum_{k=1}^{K}\left(\prod_{h=1}^{l} q_k^{(h)}(i_h,i_h)\right)\left(\bigotimes_{h=l+1}^{H} Q_k^{(h)}\right)$
    $+ Q_D((i_1,\ldots,i_l),(i_1,\ldots,i_l))$ for $(i_1,\ldots,i_l) \in \times_{h=1}^{l}\mathcal{S}^{(h)}$
    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.
- Nonrecursive block upper-triangular solution algorithm for BSOR is also possible [Buchholz-Dayar'04a] and block row-oriented version is preferable.

Block iterative methods for Kronecker products (continued)

Algorithm for nonrecursive block upper-triangular solution at level l

$\mathbf{b} = (1-\omega)\pi_{(m)}Q_D + (1-\omega)\pi_{(m)}Q_{DU(l)} + (1-\omega)\pi_{(m)}Q_{DL(l)} - \omega\pi_{(m)}Q_{L(l)}$;
For row of blocks $(i_1,\ldots,i_l) = (0,\ldots,0)$ to $(n_1-1,\ldots,n_l-1)$ lexicographically,
  Solve $\pi_{(m+1)}((i_1,\ldots,i_l))Q((i_1,\ldots,i_l),(i_1,\ldots,i_l)) = \mathbf{b}((i_1,\ldots,i_l))$;
  For column of blocks $(j_1,\ldots,j_l) > (i_1,\ldots,i_l)$,
    $\mathbf{b}((j_1,\ldots,j_l)) = \mathbf{b}((j_1,\ldots,j_l))$
      $-\omega\pi_{(m+1)}((i_1,\ldots,i_l))Q_{U(l)}((i_1,\ldots,i_l),(j_1,\ldots,j_l))$.

- In BSOR, ns diagonal blocks $Q((i_1,\ldots,i_l),(i_1,\ldots,i_l))$ must be solved in lexicographical order.
- After each block is solved for unknown subvector $\pi_{(m+1)}((i_1,\ldots,i_l))$, $\mathbf{b}$ 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

- 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].

A simple multilevel method for Kronecker products

Let:

- $\mathcal{S}_{(l)} = \times_{h=l+1}^{H} \mathcal{S}^{(h)}$ for $l = 0, \ldots, H$
- 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 $l = 1, \ldots, H$ 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 $\alpha_{(m,l)} \in [\max_{\mathbf{i}_{(l)} \in \mathcal{S}_{(l)}} |\tilde{q}_{(m,l)}(\mathbf{i}_{(l)}, \mathbf{i}_{(l)})|, \infty)$ at level l for iteration m.

A simple multilevel method for Kronecker products (continued)

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

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

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

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

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

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

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

The simple multilevel method for Kronecker products (continued)

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

At iteration m, recursion ends and backtracking starts when:

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

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

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

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.

A 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, $P_{\mathbf{x}_{(m,l)}}$:
  - depends on $\mathbf{x}_{(m,l)}$
  - has the nonzero structure of $R_{(l)}^T$.

A simple multilevel method for Kronecker products (continued)

- $P_{\mathbf{x}_{(m,l)}}$ 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 $\mathbf{i}_{(l+1)}$ of vector corresponding to kth term in Kronecker representation at level $(l+1)$ for iteration m is:

$$\mathbf{a}_{(m,l+1),k}(\mathbf{i}_{(l+1)}) = \frac{\left(\sum_{\mathbf{j}_{(l)} \in \mathcal{S}_{(l)}, f_{(l)}(\mathbf{j}_{(l)}) = \mathbf{i}_{(l+1)}} \mathbf{x}_{(m,l)}(\mathbf{j}_{(l)})\ \mathbf{a}_{(m,l),k}(\mathbf{j}_{(l)})\ (\mathbf{e}^T_{\mathbf{j}_{(l)}(l+1)} Q_k^{(l+1)} \mathbf{e})\right)}{\pi_{(m,l+1)}(\mathbf{i}_{(l+1)})}$$

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

where $\mathbf{a}_{(m,0),k} = \mathbf{e}$, $\mathbf{j}_{(l)}(l+1) \in \mathcal{S}^{(l+1)}$, and $\mathbf{e}_{\mathbf{j}_{(l)}(l+1)}$ is $\mathbf{j}_{(l)}(l+1)$st column of I.

A simple multilevel method for Kronecker products (continued)

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

- Second summation returns diagonal matrix which sums rows of $\tilde{Q}_{(m,l+1)}$ to 0.
- No need to store $\mathbf{a}_{(m,0),k} = \mathbf{e}$ for $k = 1, \ldots, K$ 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 $\mathbf{a}_{(m,l+1),k} = \mathbf{e}$ for those k which:

- either have single $Q_k^{(h)} \neq I$ for $h = 1, \ldots, H$,
- 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 3-dimensional problem with the parameter set $(\lambda_1, \lambda_2, \lambda_3, \mu_1, \mu_2, \mu_3, \mu) = (1, 2, 3, 2, 4, 6, 10)$, $\pi_{(0,0)} = \mathbf{e}/12$, $\alpha_{(0,0)} = 22$, and $\eta_{(0,0)} = \nu_{(0,0)} = 1$.

Then, $\boxed{\mathbf{x}_{(0,0)} = \pi_{(0,0)}(I + \tilde{Q}_{(0,0)}/22)}$ yields

$$x_{(0,0)} = (19\ 11\ 13\ 10\ 12\ 9\ 13\ 10\ 12\ 9\ 11\ 3)/132$$

$$R_{(0)} = \begin{array}{c} \\ 0\,0\,0 \\ 0\,0\,1 \\ 0\,1\,0 \\ 0\,1\,1 \\ 0\,2\,0 \\ 0\,2\,1 \\ 1\,0\,0 \\ 1\,0\,1 \\ 1\,1\,0 \\ 1\,1\,1 \\ 1\,2\,0 \\ 1\,2\,1 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{array} \\ \left(\begin{array}{cccccc} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \\ 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{array}\right) \end{array}$$

Example (continued)

$$P_{\vec{x}_{(0,0)}} = \begin{array}{c} \\ 0\,0 \\ 0\,1 \\ 1\,0 \\ 1\,1 \\ 2\,0 \\ 2\,1 \end{array} \begin{array}{c} \begin{array}{cccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 2 & 2 & 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{array} \\ \left(\begin{array}{cccccccccccc} \frac{19}{32} & & & & & & \frac{13}{32} & & & & & \\ & \frac{11}{21} & & & & & & \frac{10}{21} & & & & \\ & & \frac{13}{25} & & & & & & \frac{12}{25} & & & \\ & & & \frac{10}{19} & & & & & & \frac{9}{19} & & \\ & & & & \frac{12}{23} & & & & & & \frac{11}{23} & \\ & & & & & \frac{3}{4} & & & & & & \frac{1}{4} \end{array}\right) \end{array}$$

- 12 states represented by 3-tuples in $\mathcal{S}_{(0)} = \mathcal{S}$ are mapped to 6 states represented by 2-tuples in $\mathcal{S}_{(1)}$.
- For instance, states $(0,0,0)$ and $(1,0,0)$ are mapped to $(0,0)$, whereas states $(0,0,1)$ and $(1,0,1)$ are mapped to $(0,1)$.

Example (continued)

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

$$\pi_{(0,1)} = (32\ 21\ 25\ 19\ 23\ 12)/132.$$

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

$$\begin{aligned}
\mathbf{a}_{(0,1),1} &= (45/32\ 31/21\ 37/25\ 28/19\ 34/23\ 5/4), \\
\mathbf{a}_{(0,1),2} &= \mathbf{a}_{(0,1),3} = \mathbf{e}, \\
a_{(0,1),4} &= (65/16\ 100/21\ 24/5\ 90/19\ 110/23\ 5/2).
\end{aligned}$$

and aggregated CTMC is expressed as

$$\begin{aligned}
\tilde{Q}_{0,1} &= P_{\mathbf{x}_{(0,0)}} \tilde{Q}_{(0,0)} R_{(0)} \\
&= \sum_{k=1}^{4} \mathsf{diag}(\mathbf{a}_{(0,1),k}) \bigotimes_{h=2}^{3} Q_k^{(h)} - \sum_{k=1}^{4} \mathsf{diag}(\mathbf{a}_{(0,1),k}) \bigotimes_{h=2}^{3} \mathsf{diag}(Q_k^{(h)} \mathbf{e}).
\end{aligned}$$

Example (continued)

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

$$\tilde{Q}_{(0,1)} = \begin{array}{c} \\ 0\,0 \\ 0\,1 \\ 1\,0 \\ 1\,1 \\ 2\,0 \\ 2\,1 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{array} \\ \left(\begin{array}{cccccc} -5 & 3 & 2 & & & \\ \hline 6 & -8 & & 2 & & \\ \hline 4 & & -9 & 3 & 2 & \\ \hline & 4 & 6 & -12 & & 2 \\ \hline & & 4 & & -7 & 3 \\ \hline \frac{5}{2} & & & 4 & 6 & -\frac{25}{2} \end{array}\right) \end{array}.$$

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.

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 $\mathcal{S}$ 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.

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