Queueing Theory in a World where most Queueing Problems are Solved by Simulation

Winfried Grassmann
University of Saskatchewan

- Monte Carlo simulation is one of the most successful applications in Operations Research and beyond.
- The bulk of queueing theory uses deterministic methods, that is, methods not subject to randomnes.
- Outside queueing theory, many queueing models are solved by Monte Carlo simulation, and the results of queueing theory are often ignored
- How can classical queueing theory be made more successful?

Why is simulations is successful

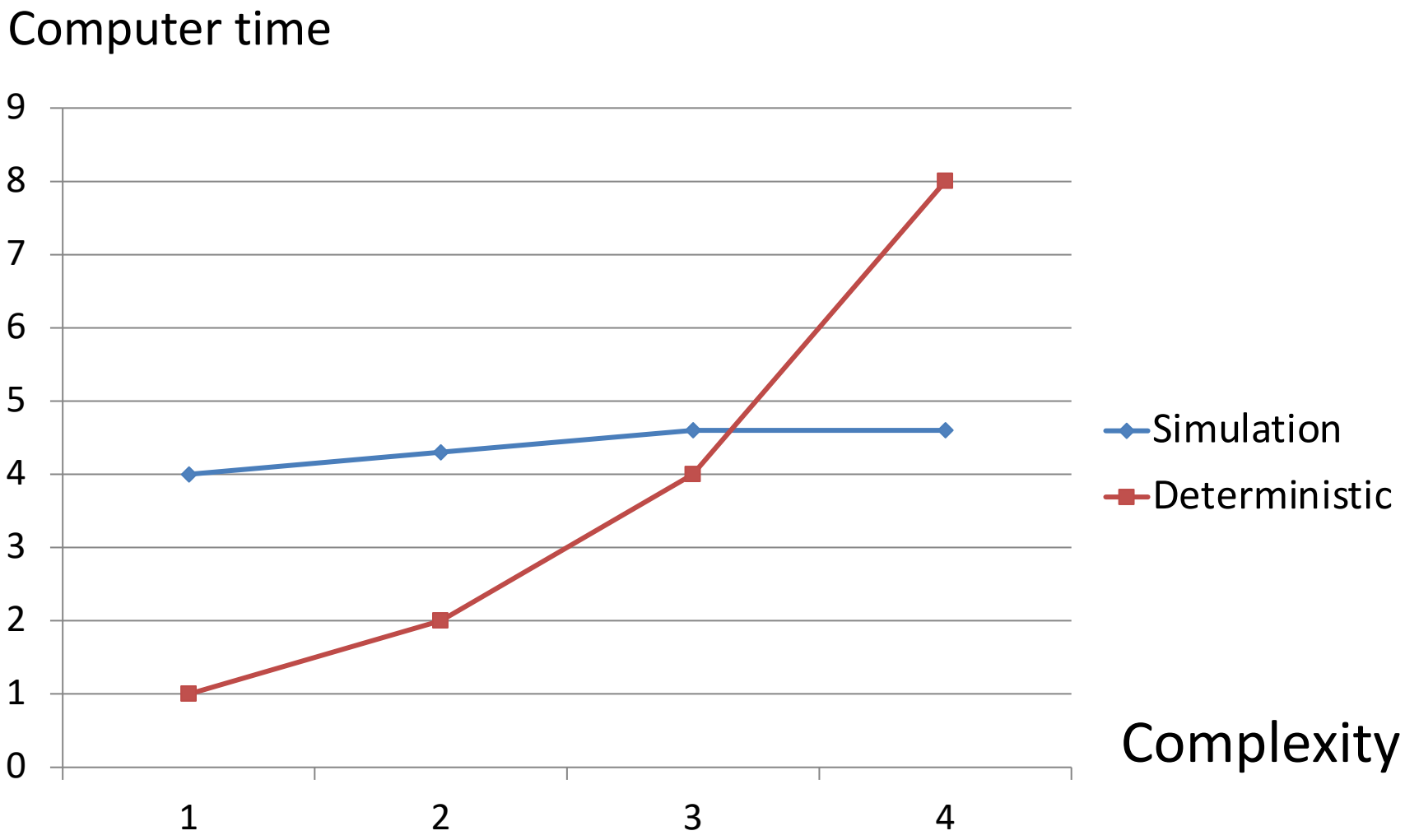
1. For large problems, Monte Carlo simulations are faster.
2. Simulation is mathematically easier. Most people have no difficulty doing simulation.
3. Simulation is much more flexible.

How can queuing theory compete?

Comparison of Execution Times

- In Monte Carlo simulation, execution times start at a high level, but they increase linearly with model complexity.
- In deterministic methods, execution times start at a low level, but they increase exponentially with model complexity. Queueing theory is at an advantage for small models, but not for large models

Simulation versus Deterministic



Can we simplify the mathematics?

- Stress numerical methods. People want results rather than a formula.
- Choice: standard numerical methods versus methods tailored to deal with probabilities?
- Standard methods:
  - Software available
- Methods tailored to deal with probabilities:
  - Probabilities are ≥ 0: Subtractions can be avoided, reducing rounding errors.
  - Algorithms often have probabilistic interpretation, providing thus additional insights.

Can we Increase Flexibility?

- One simulation paradigm: Discrete Event Simulation
- Copy this paradigm: Discrete Event Systems (Cassandras 2008).
- Components of discrete event systems
  - State: set of variables, e. g. queue lengths
  - All changes occur in discrete steps.
  - Every change is caused by events. Event: defined as something that changes state.
- Queues are discrete event systems. Why not use this paradigm in queueing theory?

State

- Physical state variables $X_1, X_2, \ldots, X_d$ (e. g. queues) , $0 \leq X_i \leq N_i$.
- Supplementary state variables:
  - Keep track of the schedule for future events, or
  - record when past events have happened.
- Enough supplementary variables must be added such that the distributions of future state variables can be calculated.
- System becomes a Markov chain, with state space ***S***, consisting of physical and supplementary state variables.

Events

- Events come of different types, such as arrivals to a waiting line, switch from line 1 to line 2, and so on.
- Each event k is characterized by
  - An event function $f_k(X)$: If the state before event k is X, the state after event k is $f_k(X)$.
    Example: state: $X = [X_1, X_2]$.
    $$f_k(X_1, X_2) = [X_1+1, X_2].$$
  - An event condition $C_k(X)$: Event prevented if event condition is not met.
  - An event process for each event:

    Poisson process , renewal process, PH process, ...

Example

There are two bins, 1 and 2.
Arrivals to bin 1 (2) : λ_1, (λ_2)
Repair: always 1 unit from bin 1, 2 from bin 2: μ
State: X_1 , # in bin 1, X_2: # in bin 2.

Event type	Event function	Event condition	Rate
Replenish 1	$X_1 + 1, X_2$	$X_1 < N_1$	λ_1
Replenish 2	$X_1, X_2 + 1$	$X_2 < N_2$	λ_2
Repair	$X_1 - 1, X_2 - 2$	$X_1 > 0$, $X_2 > 1$	μ

Create Transition Matrix Q, Poisson Events

Find Q = [q(i,j)] as follows

for all XϵS,

for all events k

If C_k(X) holds then Xnew = f_k (X)

q(X,Xnew) = rate of event k

next k

next X

Finding state numbers

- To store transition matrix, each state must be given a state number n, which is in a one-to-one relation with the state.
- This number is used to find row of the state, as well as the column.
- State Space:
  $[X_1, X_2, \dots X_d], 0 \leq X_i \leq N_i, i = 1,2, \dots, d$
- For example, use alphanumerical order:

$$n = \sum_{i=1}^{d} u_i X_i$$

$$u_i = \prod_{j=i+1}^{d} (N_j + 1), i = 1,2, \dots, d$$

Events Generated by Renewal Processes

- Times between events often iid random variables.
- We must add supplementary variables
  - Time since previous event of the same type
  - Time the next event of the same type is scheduled.
- Need to discretize, or use phase-type variables.
- Discretization forces us to deal with multiple events occurring simultaneously, increasing complexity.
- For equilibrium solutions, one can save computer time by embedding the system at the points where events occur.

Transient and equilibrium probabilities

- Wanted: $\pi_j(t)$, prob. to be in j at t
- Discrete: $P = [p_{i,j}]$

$$\pi_j(t+1) = \sum_{i=1}^{N} \pi_i(t) p_{i,j}, j = 1{:}N$$

$$\pi_j = \sum_{i=1}^{N} \pi_i p_{i,j}$$

- Continuous: $Q = [q_{i,j}]$

$$\pi_j{}'(t) = \sum_{i=1}^{N} \pi_i(t) q_{i,j}, j = 1{:}N$$

$$0 = \sum_{i=1}^{N} \pi_i q_{i,j}$$

Properties of Matrices

- Matrices huge, but sparse
  - All event processes Poisson
    - Matrix size: $N = \prod_{i=1}^{d}(N_i + 1)$
    - Entries per row = number of events (e)
    - Density: $\frac{e}{N}$
  - Event processes renewal, $0 \leq Y_i < M_i$, $i = 1,2, \ldots, e$
    - Matrix size: $N = \prod_{i=1}^{d}(N_i + 1) \prod_{i=1}^{e} M_i$
    - Density: $2^e/N$ for M_i not too small
- Matrix banded (alphanumerical order, Poisson)
  - if X_1 changes by at most 1, the state number changes by $u_1 = \prod_{i=2}^{d}(N_i + 1)$.

Examples

- System state: $X_1, X_2, X_3\ X_4,\ 0 \leq X_i \leq 9.$

  (e. g. queues) , all events Poisson

  $$10^4 = 10{,}000 \text{ states}$$

  – Transition matrix: $(10^4)^2$= 100,000,000 entries.

  – Assume: 5 events (tandem queue): Only 50,000 non-zero entries.

- Small densities for events generated by renewal processes.
- Store only non-zero entries

Storing only non-zero Elements

3 arrays: row, col, prob. Apply to finding $\pi_j(t+1)$

$$\pi_j(t+1) = \sum_{i=1}^{N} \pi_i(t) p_{i,j}$$

Assume: mtot non-zero entries.

```
pit(i) = 0, i = 1:N; pit(1)=1; pinew(i) =0, i=1:N
for t = 1:ttot
        for m = 1:mtot
                i = row(m)
                j = col(m)
                pinew(j) = pinew(j) + pit(i) * prob(m)
        next m
        pit(i) = pinew(i), i = 1:N;  pinew(i) =0, i=2:N
next t
```

Transient Solutions, Continuous-time

$$P = \frac{Q}{f} + I, f \geq -q_{i,i} \text{ for all } i$$

$$P = \left[p_{i,j}\right] \text{ is a stochastic matrix}$$

$$\pi_j{}^{n+1} = \sum_{i=1}^{N} \pi_i{}^{n} p_{i,j}$$

$$\pi_j(t) = \sum_{n=0}^{\infty} \pi_j{}^{n}\, p(n; ft)$$

$$p(n; ft) = e^{-ft}(ft)^n \frac{1}{n!}$$

Time between changes not 1, but exponential with rate f.

Randomization or Uniformization

Objection from Mathematicians

- This is essentially a Taylor expansion of the matrix exponential.
- Taylor expansions tend to be numerically unstable.
- Therefore, this method is potentially unstable.

Not true

- Reason: no subtractions! If there are no subtractions, rounding errors increase very slowly. If there are subtraction, they magnify earlier errors.
- Randomization/Uniformization works for millions of states.
- Performs much better than the standard algorithm for solving differential equations.

Steady State Solutions

- Only discrete-time Markov chains discussed.

$$\pi_j = \sum_{i=1}^{N} \pi_i p_{i,j}$$

- Gaussian elimination. Mathematical Opinion: unstable if number of states large.
- Reformulate such that subtractions are avoided.
- The resulting algorithm has a probabilistic interpretation.
- Rule in numerical mathematics: start with smallest entities $\rightarrow$ $\mathrm{Eliminate}\ \pi_N$ first!

Gaussian Elimination

$$\pi_j = \sum_{i=1}^{N} \pi_i p_{i,j},\ j = 1{:}\,N$$

Set $j = N$, solve for π_N

$$\pi_N = \sum_{i=1}^{N-1} \pi_i \frac{p_{i,N}}{1-p_{N,N}}$$

Substitute:

$$\pi_j = \sum_{i=1}^{N-1} \pi_i p_{i,j} + \pi_N p_{N,j}$$

$$= \sum_{i=1}^{N-1} \pi_i p_{i,j} + \sum_{i=1}^{N-1} \pi_i \frac{p_{i,N}}{1-p_{N,N}} p_{N,j}$$

$$\pi_j = \sum_{i=1}^{N-1} \pi_i \left(p_{i,j} + \frac{p_{i,N} p_{N,j}}{1-p_{N,N}} \right),\ j = 1{:}\,N-1$$

Let ${p_{i,j}}^{N-1} = p_{i,j} + \frac{p_{i,N}p_{N,j}}{1-p_{N,N}}, i,j < N$

We now have a new system of equations, from which we can eliminate π_{N-1}, which yields a system of equations involving only π_{N-2}, π_{N-3}, ... π_1. The coefficients of the resulting systems will be denoted by ${p_{i,j}}^{n}$. One has

$$ {p_{i,j}}^{n-1} = {p_{i,j}}^{n} + \frac{{p_{i,n}}^{n}{p_{n,j}}^{n}}{1-{p_{n,n}}^{n}} $$

$$ \pi_n = \sum_{i=1}^{n-1} \pi_i \, {p_{i,n}}^{n} / \left(1-{p_{n,n}}^{n}\right) $$

Elimination = Embedding

Given: sequence $X(1), X(2), X(3), \cdots$, set C: Create subsequence: $X(t)$ included only if $X(t) \in C$.

Example $C = \{1,2,3\}$

$X(t)$: 1,3,5,2,4,2

$X^C(t)$: 1,3,2,2

Embedding into an embedded sequence:

Embed $X^C(t)$ into $D = \{1,2\}$

$X^D(t)$: 1,2,2

Equivalent to embedding original sequence into D!

Reduce state space **S** successively by 1 state, until only one state left. State reduction.

Embedded Markov Chain

- If $X_1, X_2, \ldots$ is a DTMC, so is any subsequence obtained by using embedding set C.
- Problem: find transition matrix ${p_{i,j}}^{\{C\}}$
- Solution: add the probabilities of all paths that start in i, end in j, and avoid any state of C in between.

Elimination = Embedding

Original sequence includes states 1,2, ... N.

Embed into $C = \{1,2,\cdots,N-1\}$

Find ${p_{i,j}}^{\{C\}}$. Add probabilities of possible sequences:

$$i,j\colon p_{i,j}$$

$$i,N,j\colon\ p_{i,N}\ p_{N,j}$$

$$i,N,N,j\colon p_{i,N}p_{N,N}p_{N,j}$$

$$i,N,N,N,j\colon\ p_{i,N}\ \ {p_{N,N}}^{2}\ p_{N,j}$$

${p_{i,j}}^{\{C\}} = p_{i,j} + \sum_{k=0}^{\infty} p_{i,N}\ \ {p_{N,N}}^{k}\ p_{N,j} = p_{i,j} + \frac{p_{i,N}p_{N,j}}{1-p_{N,N}}$

This, however, is ${p_{i,j}}^{N-1}$. The rest follows.

Reduce state space 1 by 1: **State reduction**

Remove Subtraction from State Reduction

$$\pi_j = \sum_{i=1}^{N-1} \pi_i \left(p_{i,j} + \frac{p_{i,N} p_{N,j}}{1 - p_{N,N}} \right)$$

$$1 - p_{N,N} = \sum_{j=1}^{N-1} p_{N,j}$$

It follows that

$$p_{i,j}^{\,n-1} = p_{i,j}^{\,n} + \frac{p_{i,n}^{\,n} p_{n,j}^{\,n}}{\sum_{j=1}^{N-1} p_{n,j}}$$

GTH method: Grassmann/Taksar/Heyman.

Very stable!

Problems with State Reduction

- If matrix dense:
  $\frac{2N^3}{3}$ floating point operations (flops)
- In the example, we had $N = 10^4$ states:
  $\frac{2}{3}10^{12}$ flops, which requires around 100,000 seconds computer time $\approx$ 1 day on a laptop!
- Bandedness helps somewhat. $\approx \frac{N^3}{(N_1+1)^2}$
- Note: Computer time to calculate transient solutions increases, $\mathrm{per\ iteration}$, with Ne. Linear with N. Faster!

Iterative Methods

- Calculate transient solutions
- Improve convergence by using the jump matrix.
- Gauss-Seidel. Order of states is important for convergence. Typically, order in the opposite direction of the flow (e.g. tandem queues: last queue first, first queue last). (Mitra and Tsoucas)

Theoretical Uses of State Reduction

- Dealing with infinite state spaces.
- Dealing with matrices having repeated structure.
- Reducing state space through embedding. Time does not permit to deal with this issue.

Infinite state spaces

If states below n recurrent, then the state space can be cut without changing the ${p_{i,j}}^n$ by more than ε. Consider the paths

$$X(t) = i, X(t+1) > n, X(t+2) > n, \ldots X(t+R-1) > n, X(t+R) \leq n$$

${p_{i,j}}^n$ can be found by adding all the probabilities of all these paths.

If system recurrent, then for each $\varepsilon > 0$, there exits a v such that Prob$\{R > v\} < \varepsilon$.

If one considers only paths with $R \leq v$, then you cannot reach states above vd , where d is the largest possible value $X(t+1) - X(t)$ can assume, that is, the largest jump up.

Consequently, cutting all states > vd will not change ${p_{i,n}}^n$ by more than ε.

Transition Matrices with Repeating Columns

For models with $p_{i,j} = p_{i+k,j+k}$, $i, j > c$, we have

$$p_{i,j}{}^{n} = p_{i+k,j+k}{}^{n+k}.$$

Proof: Any path

$X(t) = i, X(t+1) > n, X(t+2) > n, \ldots X(t+m-1) > n, X(t+m) \leq n$

can be matched with

$X(t) = i+k, X(t+1) > n+k, X(t+2) > n+k, \ldots X(t+m-1) > n+k, X(t+m) \leq n$+k

This path obviously has the same stochastic structure as the original path, completing the proof.

Allows to reduce MAM methods to state reduction.

Conclusions

To increase the flexibility

use event-based approaches.

To reduce computer time

try to reduce number of state variables.

To make the mathematics easier

use numerical methods, modified to deal with probabilities.

Literature

- C.G. Cassandras and S. Lafortune, 2008, Introduction to Discrete Event Systems, Springer Verlag.
- W. K. Grassmann, 2000, Computational Probability, Kluwer Academic Publishers.
- W. K. Grassmann, M. Taksar and D. P. Heyman, 1993, Regenerative Analysis and Steady State Distributions for Markov Chains, Operations Research 33, 1107—1117.
- W. K. Grassmann and D. P. Heyman", 1990, Equilibrium Distribution of Block-Structured Markov Chains with Repeating Rows, Journal of Applied Probability 27, 557—576.
- W. K. Grassmann, 1985, The Factorization of Queueing Equations and Their Interpretation, J. Opl. Res. Soc. 36, 1041—1050.
- W. K. Grassmann, 1977, Transient Solutions in Markovian Queueing Systems, Computers and Operations Research 4, 47—56.
- D. Mitra and P. Tsoucas, 1987, Convergence of Relaxations for Numerical Solutions of Stochastic Problems}, in “Computer Performance and Reliability”.
- W. J. Stewart, 1994, An Introduction to the Numerical Solution of Markov Chains, Princeton University Press.