INFOR, Vol. 46, No. 1, February 2008, pp. 3–14
ISSN 0315-5986 | EISSN 1916-0615

Four Canadian Contributions to Stochastic Modeling

Winfried K. Grassmann
Department of Computer Science, University of Saskatchewan, Saskatoon, SK S7N 5C9, email:grassman@cs.usask.ca,

Martin L. Puterman
Sauder School of Business, University of British Columbia, Vancouver, BC V6T 1Z2, email:marty@chcm.ubc.ca,

Pierre L'Ecuyer
Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, QC H3C 3J7, email:lecuyer@iro.umontreal.ca

Armann Ingolfsson
School of Business, University of Alberta, Edmonton AB T6G 2R6, email:armann.ingolfsson@ualberta.ca

Abstract — We outline the history, significance, and impact of four important contributions by Canadian researchers to stochastic modeling for operational research: the use of the uniformization method to compute transient probabilities for Markov chains, pioneered by Winfried K. Grassmann, contributions to Markov decision processes by Martin L. Puterman, contributions to the development of random number generators by Pierre L'Ecuyer, and contributions to queueing theory software by Armann Ingolfsson.

Keywords Stochastic modeling, uniformization method, Markov decision processes, random number generation, queueing theory software.

1. INTRODUCTION

Operational research employs a variety of mathematical tools, the primary ones being optimization methods and stochastic models. Our focus in this paper is on stochastic modeling for operational research. We do not attempt an exhaustive survey; instead we describe three major bodies of work that have had a major and sustained impact: (1) the use of the uniformization (or randomization) method to compute transient probabilities for Markov chains (Grassmann 1977a), pioneered by Winfried K. Grassmann, (2) contributions to Markov Decision Processes (MDP) (Puterman 1994), by Martin L. Puterman, and (3) contributions to the development of pseudo-random number generators (RNGs) (L'Ecuyer 1988), by Pierre L'Ecuyer. In addition, we describe recent contributions to the development of easy-to-use software for queueing calculations (Ingolfsson and Gallop 2003) by Armann Ingolfsson.

All four of these contributions have had a significant impact on theory, practice, or both. The uniformization method is mentioned in many text and reference books on stochastic processes and queueing theory (de Souza e Silva and Gail 2000, Kao 1997, Tijms 2003, Gross and Harris 1998, Stewart 1994, Daigle 1992, Trivedi 2002, Jain *et al*., 2007) as the method of choice for computing transient probabilities for Markov chains and it is implemented in popular software packages for performance evaluation of computer systems (Ciardo 2007, Sanders *et al*., 2007). Puterman's book on Markov Decision Processes (Puterman 1994) and a previous book chapter (Puterman 1990) are standard references for researchers in that field and have been cited over 2,000 times. The book was recently cited (Powell 2007) as "the current high-water mark for textbooks in [...] Markov decision processes." L'Ecuyer's papers on random number generation have been cited hundreds of times, and his random number generators are implemented in popular commercial simulation software packages such as SAS, Arena, Witness, Simul8, and Automod. His first paper on random number generation (L'Ecuyer 1986) was recently chosen as one of ten landmark papers published in the proceedings of the Winter Simulation Conference. Ingolfsson's *Queueing ToolPak* (QTP) spreadsheet add-in (Ingolfsson and Gallop 2003) to simplify

Received January 2008, Revised February 2008, Accepted February 2008.

computations with commonly used queueing models is used for teaching at many universities. A survey of spreadsheet add-ins for operational research (Grossman 2002) described QTP as making "queueing theory computations effortless."

In the remainder of this paper, we devote one section to each of the four contributions. Parts of the paper describe personal opinions and experiences by individual authors. For ease of reading, we use the first-person pronoun, but we include the initials of the relevant author whenever we switch from one author to another.

2. THE UNIFORMIZATION METHOD

Uniformization is a method to calculate transient probabilities for continuous-time Markov chains. It was first suggested by Jensen (1953), and it is often called Jensen's method. Another name for this method is randomization. Before we discuss uniformization, we show why Markov chains are important, and where transient solutions of Markov chains are required.

Most readers are familiar with Markovian queueing systems, such as the $M/M/1$ and the $M/M/c$ queue. Markov chains, both of the discrete and the continuous variety, are essential tools for queueing theorists. However, Markov chains are also used in maintenance, reliability and other applications. To deal with such a variety of applications, it is convenient to introduce what we are calling *Markovian event systems* or MESs (see Grassmann 1991). MESs are identical to SERTs, a term coined by Gross and Miller (1984), which stands for State variable, Event, Rate and Transition. In a Markovian event system, one has a number of variables, called state variables, which jointly determine the future of the system. A state is a particular assignment of values to all state variables. At certain rates, the state changes, and such changes are brought about by *events*. They correspond to the events used when simulating discrete event systems. Markovian event systems are special cases of discrete event systems, with the restriction that they are Markovian. Specifically, the rate of an event depends only on the present state, and not on the past of the system. This means that there is no need to schedule events, which makes the logic easier. It is relatively easy to convert Markovian event systems into continuous-time Markov chains. The simplest way, and a method that is quite efficient when dealing with queueing systems, is to enumerate all states, and find, for each state and each event, the rate of the event and the state it leads to, the so called target state. The rate of the event is then entered into the transition matrix with the row being the original state, and the column the target state.

Practically any discrete event system of interest can be approximated with arbitrary precision by an MES by using phase-type distributions, because such distributions are dense on the positive axis, which allows one to deal with arbitrary distributions, be they interarrival or service time distributions, or any distribution used for scheduling. Unfortunately, phase-type distributions require additional state variables, and, as we will see later, the computational complexity, both in time and space, increases exponentially with the number of state variables.

The uniformization method is extensively used in Generalized Stochastic Petri-Nets (GSPNs) and Stochastic Automata Networks (SANs), which, like MESs, can be converted to continuous-time Markov chains. GSPNs are graphs with two types of nodes: places and transitions. Generally speaking, places represent state variables, and transitions represent events. There is a third entity in a GSPN, the token, and each place can contain a number of tokens. In this way, the tokens represent the value of the state variable corresponding to the place in question. SANs work in a similar fashion. In this sense, SANs and GSPNs are examples of MESs. What makes GSPNs and SANs important is the fact that they are implemented in software tools (Ciardo 2007, Sanders *et al*., 2007) that allow users to formulate systems and find transient probabilities through uniformization. Another package to calculate transient probabilities is MARCA (Stewart 2007).

In queueing theory, one typically concentrates on steady-state solutions, yet real-life systems are almost never in steady-state. In this sense, transient solutions are more realistic. One reason for using steady state results is that in many cases, they are easier to obtain than transient results. Also, the initial probabilities of being in the different states are often unknown. On the other hand, if these initial probabilities are known, and if the transient probabilities are not too difficult to obtain, one should use them rather than equilibrium probabilities. In particular, if rates are piecewise constant when considered as functions of time, and this is a reasonable approximation (Ingolfsson *et al*., 2007), then transient solutions should be considered. The distribution of the state variables after the rate change then provides the initial distribution for the new period. Other applications of transient solutions involve waiting-time problems and embedded Markov chains, as we will show later.

2.1 The Principles of Uniformization

To find transient probabilities of a Markov chain with state space S, one needs the initial probability vector $\pi(0) = [\pi_i\ (0),\ i \in S]$ as well as the transition rates q_{ij}, $i, j \in S$, $i \neq j$. Also, q_{ii} is obtained by multiplying therate q_i of leaving state i by -1, that is, $q_{ii} = -q_i$. Let $Q = [q_{ij},\ i, j \in S]$. The first step of uniformization is to find the maximum leaving rate $q = \max(q_i)$, and *uniformize* Q according to the formula

$$P = Q/q + I. \tag{1}$$

Note that P has no negative entries, and each row of P has a sum of 1, at least provided the rows of Q all have a sum of 0. Hence, P can be considered as the transition matrix of a discrete-time Markov chain. Let $\pi^n = [\pi_i^n]$ be the probabilities

ISSN 0315-5986 | EISSN 1916-0615 DOI 10.3138/infor.46.1.3

of being in state i after n steps of this discrete Markov chain. We can find π^n recursively as

$$\begin{aligned} \pi^0 &= \pi(0) \\ \pi^{n+1} &= \pi^n P, n \geq 0. \end{aligned} \tag{2}$$

To find $\pi(t)$, we need the Poisson distribution with parameter qt, given by $p(n; qt) = e^{-qt}\,(qt)^n/n!$ The transient solutions for time t are now:

$$\pi(t) = \sum_{n=0}^{\infty} \pi^n p(n; qt). \tag{3}$$

For calculations, the sum in (3) cannot be extended to infinity. However, its terms are majorized by $p(n;qt)$, and one can extend the sum until $p(n;qt)$ is below a certain precision ϵ. Alternatively, one can use the fact that $p(n;fq)$ is approximately normal if qt is large. This suggests extending the sum until $n = M$, with

$$M = qt + z_\alpha \sqrt{qt}, \tag{4}$$

where z_α, the α-fractile of a standard normal distribution, is, say, 4, which means that $\alpha = 0.000032$.

It is noteworthy that P has the same equilibrium probabilities as Q, that is, $0 = \pi Q$ has the same equilibrium vector π as $\pi = \pi P$, as one can easily verify.

Before discussing (3) further, let me discuss how I (WKG) arrived at it. I was interested in obtaining equilibrium solutions for large Markov chains, and at this stage, Rolf Schassberger, then at the University of Calgary, suggested I iterate (2) until π^n reaches steady-state. Since the matrices I considered were very sparse, this is faster than Gaussian elimination. I tried this method, and it worked well. After joining the University of Saskatchewan in 1969, I studied volume 2 of Feller (1971), who uses what he calls randomization extensively to convert discrete-time processes into continuous-time processes. For instance, in the case of the discrete Markov chain given by P, instead of choosing a step size of 1, he chooses an exponential random variable as his step-size. If the expectation of this random variable is $1/q$, the result is equation (3). Indeed, a generalization of (3) is given by equation (9.1) of page 353 in Feller (1971). Since the recursion given by (2) worked so well, I concluded that (3) should also work well, which indeed it did. I wrote a program in 1971, which was submitted to the SHARE program library (Grassmann and Ngai 1971). Very soon, I added an MES-based matrix generator to this program. I had some difficulties publishing my results, and my first paper on uniformization did not appear until 1977 (Grassmann 1977a). Hence, except for one early use (Love 1977), the method did not receive much initial attention. This only changed after I met Don Gross during an ORSA/TIMS meeting where he gave a presentation involving transient solutions. I suggested he use the unifomization method, then called randomization, and he became an early convert. Together with Doug Miller, he wrote the famous paper "The Randomization Technique as a Modeling Tool and Solution Procedure for Transient Markov Processes" (Gross and Miller 1984), which became an instant hit. He gave me full credit by stating that randomization was "first introduced as a computational method by Grassmann." Some people might think that introducing randomization as a computational method is easy, but this was not the case. The issue is that there were a number of standard methods, such as the Runge-Kutta method, and people claimed that these methods were superior. As Don Gross told me, this is not true. He stated that randomization is often an order of magnitude faster than Runge-Kutta (Gross 1980). More recent studies have confirmed this assessment (Ingolfsson *et al*., 2007).

2.2 *Extensions*

If Markov chains are used to model computer systems or production systems, the total revenue within a certain time interval, say from 0 to T, is often of great interest. To handle this situation, one typically introduces rewards, which accumulate at a rate r_i while the system is in state i. This implies that the unconditional reward rate at time t is

$$r(t) = \sum_{i \in S} r_i \pi_i(t).$$

If $\bar{r}(T)$ is the total expected reward from 0 to T, one finds using (3)

$$\bar{r}(T) = \int_0^T r(t)dt = \sum_{n=0}^{\infty} \sum_{i \in S} r_i \pi_i^n \int_0^T p(n; qt)dt. \tag{5}$$

Note that

$$\int_0^T p(n; qt)dt = \frac{1}{q} \sum_{m=n+1}^{\infty} p(m; qt) = \frac{1}{q}\left(1 - \sum_{m=0}^{n} p(m; qt)\right).$$

Since the π_i^n and $p(n;qt)$ must be calculated in any event, we obtain $\bar{r}(T)$ with little extra effort.

The method just mentioned can be extended to find the variance of the reward accumulating from 0 to T (Grassmann 1987). This is important when dealing with production (Tan 1999), or performability (de Souza e Silva and Gail 1989, 1986). Similar considerations are also important to assess the efficiency of simulation (Grassmann and Luo 2005, Lock 1988, Oni 2003, Sethi 2007). It is more difficult to obtain the entire distribution of the actual total reward from 0 to T, but this can be done as well (de Souza e Silva and Gail 2000, 1989, 1986).

Uniformization can also be used to deal with embedded Markov chains (Ferreira and Pacheco 2006, Grassmann

DOI 10.3138/infor.46.1.3

1982). In this case, one needs integrals of the form

$$\int_0^\infty \pi_i(t)dG(t),$$

where $G(t)$ is a given distribution. Using arguments similar to the ones used to derive (5), one can determine this integral by replacing $p(n;qt)$ in (3) by

$$\int_0^\infty p(n;qt)dG(t).$$

For efficient methods to find this integral, see Grassmann (1982).

Randomization is used extensively for finding waiting times (Grassmann 1977b, Melamed and Yadin 1984a,b). To do this, one tags a customer, and follows this customer through the system. To the system, one now adds an absorbing state, say ø, which represents the fact that the tagged customer has left. Then it follows that

$$P\{\text{Wait} < t\} = \pi_\text{ø}(t).$$

2.3 Algorithms Involving No Subtractions

It is noteworthy that the uniformization method does not involve any subtractions, a property is shares with the Grassmann/Taksar/Heyman (GTH) algorithm (Grassmann *et al*., 1985). In Grassmann (1993), it is shown that algorithms not containing subtractions are numerically very stable. The reason is that such algorithms avoid *subtractive cancellation*, a term that refers to the often catastrophic loss of digits when forming the difference of two floating point numbers of approximately equal size. It can be shown that when there are no subtractions, one needs only consider the relative as opposed to the absolute errors (Grassmann 1993). However, the classical way to deal with rounding involves adding error terms, and this is inappropriate when dealing with relative errors. Numerical analysts trained in classical methods have some difficulty with this, as I learned both as an author and as an associate editor. In fact, my first attempt to publish the randomization method met with a rejection. The referee claimed that the randomization method is unstable, and that the Runge-Kutta would be a far better method. His argument was as follows: it is known that the Taylor expansion of $\pi(t)$, given by

$$\pi(t) = \pi(0)e^{Qt} = \pi(0)\sum_{n=0}^{\infty}(Qt)^n/n! \qquad (6)$$

is numerically unstable. The reason is that for high enough t, the term $(Qt)^n/n!$ increases rapidly with t and initially even with n. The sum of all terms must be a probability, that is, it must be between 0 and 1, which implies that there is rampant subtractive cancellation. The referee then noted that (3) is also a Taylor expansion, which can be shown as follows:

$$\pi(0)e^{(P-I)qt} = \sum_{n=0}^{\infty}\pi(0)P^n e^{-qt}(qt)^n/n! = \sum_{n=0}^{\infty}\pi(0)P^n p(n;qt).$$

This, however, immediately leads to (3). Hence, for the referee, the case was clear: Taylor expansions are numerically unstable, and what my submission proposed was a Taylor expansion. What he missed was that P has no negative elements, and consequently no subtractive cancellation. In fact, an upper and a lower bound for the relative rounding error of (3) is given in equation (19) of Grassmann (1993), and these bounds are very narrow and sufficient for most practical purposes.

2.4 Other Issues

When applying uniformization, one has to calculate a probability for each state $i \in S$. Unfortunately, N, the number of states in S, increases exponentially with the number of state variables, and even for small systems, N can reach a thousand or even a million. Even though systems with N as high as one million can be solved by uniformization, we have to face the fact that systems with hundreds or more state variables cannot be solved that way. In fact, aside from simulation, there is no generally applicable method to find transient or even equilibrium solutions for such systems. On the other hand, for theoretical investigations, small systems are preferable, because tracking more than three or four state variables challenges the ability of most investigators.

The fact that uniformization can be applied to systems with a large state space stems from the fact that the transition matrices corresponding to MESs are typically sparse. This is true because if n_E is the number of events, then each row of Q can have at most n_E non-zero entries, and the number of all entries of Q is Nn_E , as opposed to N^2 when Q is dense. Typically, n_E is related to the number of state variables, and it seldom exceeds the square of the number of state variables. The challenge is therefore to preserve sparsity during all calculations. This excludes, for instance, methods that rely on matrix squaring.

Jensen's method cannot be used for systems that change continuously over time, say when arrival rates change during the day. To overcome this, one can slice the time into intervals, which Q remaining constant throughout the interval. The probabilities found at the end of each interval can then be used as the starting probabilities of the next interval. This method has been used by Ingolfsson *et al*. (2007), who provide a comparison of several methods that can deal with this situation.

A third problem arising in Jensen's method is the issue of stiffness (Reibman *et al*., 1989), which expresses itself by having diagonal elements varying over several orders of

DOI 10.3138/infor.46.1.3

magnitude. There are several methods to overcome this problem. For instance, van Moorsel and Sanders (1994) introduced the idea of adaptive randomization. In Carrasco and Calderon (1995), the idea was to convert the system into one that is less stiff. Another idea is to partition such systems (Miller 1983).

If some state variables have no upper bound, the state space is infinite, and the transition matrix has to be constructed on the fly, say by some breadth-first algorithm. Also, in this case, rates can approach infinity as a particular state variable increases. Both problems can be overcome by using adaptive uniformization (van Moorsel and Sanders 1994).

2.5 Conclusions

As a summary of our discussion, let me quote Sidje *et al*. (2007), who write:

> "Markovian analysts have traditionally used the uniformization method, which still remains very popular. Evidence suggests that it can work very well, especially on nonstiff problems, and this is the reason why it remains one of the most widely used method for computing transient solutions."

Of course, I fully agree with this assessment, but I would like to add two further comments: (1) For queueing systems involving only 1 or 2 queues, the computational complexity of uniformization does not challenge today's computers, and the main cost is really the time of the analyst formulating and programming his model. (2) Uniformization is probably one of the fastest methods to program, and since one need not worry about rounding errors, the programmer does not need to be a specialist in numerical analysis.

3. MARKOV DECISION PROCESSES

From performance evaluation for Markov chains, via uniformization, we now turn to optimization of Markov chains, via Markov Decision Processes.

I (MLP) first heard the term "dynamic programming" in 1965 while an undergraduate math major at Cornell. I was sitting outside the student union building one warm Spring day, having survived a brutal Ithaca winter, chatting with an industrial engineering student I knew about operations research and its subfields linear programming, dynamic programming and simulation. Without even knowing what it was, I found the term *dynamic programming* curiously engaging. Little did I know at that time that it would be the focus of the next 42 (and still counting) years of my career.

Shift forward to 1968, when I was an Operations Research Ph.D student at Stanford and taking a course in dynamic programming from Don Iglehart. I was intrigued by the wide applicability and elegant theory of dynamic programming (a term that I view as synonymous with Markov decision processes). I was especially drawn to the rich theory of value iteration and contraction mappings (Shapley 1953, Denardo 1967), policy iteration (Howard 1960), and sensitive optimality (Blackwell 1962, Veinott, Jr. 1969). While walking across campus following our class, fellow Ph.D. student Robert W. (Bob) Rosenthal (Radner and Ray 2003) raised the question "Why does policy iteration work so well?" This intriguing question proved to be the main focus of my research on MDP theory and algorithms throughout the 70s.

While at Stanford I also took courses by Chernoff and Siegmund on sequential analysis and by Karlin and Chung on stochastic processes in addition to math courses in real analysis, functional analysis and partial differential equations. This rigorous foundation would prove invaluable to my subsequent research on diffusion process control and MDPs.

When it was time to choose a dissertation topic, my advisor Arthur F. (Pete) Veinott Jr., suggested I look at the recent book on diffusion processes by Petr Mandl (Mandl 1968). In particular he wanted me to investigate whether the sensitive optimality theory for Markov Decision Processes (Veinott 1969) applied to controlled one-dimensional diffusion processes. I found Mandl's book extremely challenging and had to do considerable background reading in Markov processes, differential equations, and functional analysis to understand it. Among other results in my thesis, I developed a theory of sensitive optimality for controlled one-dimensional diffusion processes (Puterman 1974) and established the convergence of policy iteration for controlled multi-dimensional diffusions (Puterman 1977). The latter research combined methods from stochastic processes, partial differential equations and dynamic programming and would provide a lead in to my fundamental work on policy iteration.

3.1 Research on Policy Iteration

I completed my dissertation in 1972 while an assistant professor at the University of Massachusetts. But the lure of the west coast was too strong, so my wife, another Stanford alum, and I moved in 1974 to British Columbia where I became an Assistant Professor in the Faculty of Commerce at the University of British Columbia (UBC). During my first year at UBC, I presented my dissertation research in a study group in optimal control theory with Ulrich Haussman, Ray Rishel and Armand Makowski (the latter two were visiting UBC from the University of Kentucky). One of them noted that my work appeared related to quasilinearization (Kalaba 1959).

Investigating this line of research led me to recognize the link between policy iteration and Newton's method. This was formalized in Puterman and Brumelle (1979) in which we established in considerable generality that policy iteration was equivalent to Newton's method applied to the discounted dynamic programming optimality equation (OE) expressed

ISSN 0315-5986 | EISSN 1916-0615 DOI 10.3138/infor.46.1.3

in the form

$$Bv(s) \equiv \max_{a \in A} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a)v(j) - v(s) \right\} = 0 \quad (7)$$

(the notation follows Puterman, 1994). Prior to our research it was customary to express the OE as the fixed point equation

$$Tv(s) \equiv \max_{a \in A} \left\{ r(s,a) + \lambda \sum_{j \in S} p(j|s,a)v(j) \right\} = v(s) \quad (8)$$

so that it was natural to solve it with value iteration.

Newton's method in its simplest form solves an equation $f(x) = 0$, by iteratively computing $f'(x_n)$ and then solving the linearized form of $f(x) = 0$ given by

$$f(x_n) - f'(x_n)(x - x_n) = 0 \quad (9)$$

To establish the relationship between policy iteration and Newton's method, we found it convenient to re-express the OE in the form $Bv = 0$ so that solving the OE became equivalent to finding the zero (or root) of the operator B in a linear space. We then showed that determining the maximizing policy in policy iteration corresponded to the differentiation step in Newton's method and evaluating a policy corresponded to solving an equation of the form (9).

Next we addressed Bob Rosenthal's question of why policy iteration converged in so few iterations. It was known in great generality that value iteration converged linearly at rate λ. The numerical analysis literature showed that under certain smoothness properties on $f(x)$, Newton's method converged quadratically. Our work built on this theory to establish conditions under which policy iteration converged quadratically and to provide error bounds. The culmination of this work was an analytic theory for policy iteration which showed its equivalence to Newton's method and established that policy iteration converged in significantly fewer iterations than value iteration. Hence we had an answer to the question Bob Rosenthal raised many years earlier as well as a formal theory to build on.

In this research we expressed the policy iteration algorithm in terms of the recursion

$$v_{n+1} = v_n + (I - \lambda P_n)^{-1} B v_n$$

where B is defined in (7) the OE above and P_n is the transition matrix corresponding to the maximizing decision rule at iteration n. Of course, when evaluating a policy in policy iteration, it was not necessary to compute the above inverse but instead solve a linear equation which required $O(N^3)$ operations where N is the number of states. We noted that for $\lambda < 1$

$$(I - \lambda P_n)^{-1} = \sum_{m=0}^{\infty} (\lambda P_n)^m$$

and therefore we could obtain a class of algorithms by truncating the above series at M_n. The consequence of this was that when $M_n = 0$ for all n, the new algorithm was equivalent to value iteration, when $M_n = \infty$ for all n the algorithm was equivalent to policy iteration and in between we had a wide range of algorithms corresponding to different choices for the sequence M_n. We referred to this class of algorithms as modified policy iteration (MPI) and analyzed its theory and computational properties in Puterman and Shin (1978, 1982). In my opinion, MPI remains the most efficient approach for solving moderate size MDPs.

3.2 The Book

At UBC I taught a Ph.D. course on MDPs. I received encouragement from several colleagues, most notably Nico van Dijk of the University of Amsterdam, to formally write up my course notes. The final impetus came from Dan Heyman and Matt Sobel whom I met with while attending an Operations Research Society Meeting in Atlanta in 1985. They were putting together the *Stochastic Models* (Heyman and Sobel 1990) volume of the *Handbooks in Operations Research and Management Science* series and asked me to write a chapter on MDPs. This was the encouragement I needed and began this work during my sabbatical in 1987. Unfortunately, my article was far too long for a handbook chapter and had to be condensed significantly. But I had a lot of written material and a framework to build on for my book *Markov Decision Processes* (Puterman 1994). Writing this book became my passion over the next six years.

I had two main objectives for the book; to create a state of the art monograph that brought all the beautiful and deep research on MDPs together in one place using a common notation, and to provide an introduction to MDPs that would be accessible to graduate students and researchers. In particular, I wanted to provide a complete treatment of discounted models, average reward models, sensitive optimality, and continuous time models. I had hoped to also include a chapter on partially observed MDPs but due to other pressing commitments and my wife threatening to leave me if I didn't finish the book, I chose to omit this material.

I feel that my book was timely, useful and filled a significant gap; it has motivated students and researchers, it has been used as a text at many universities and it has been a source for many favorable comments and reviews. I am deeply touched when I visit other universities and find it on my colleagues' desks or bookshelves.

The book was published at a time when there was a lull in MDP research; perhaps because its theory was mature or perhaps because the main challenges at that point of time were computational. Shortly thereafter, several different groups, most notably computer scientists and electrical engineers, became actively interested in solving very large MDPs. In collaboration with operations researchers, these

DOI 10.3138/infor.46.1.3

communities created a broad range of techniques under the headings reinforcement learning, neuro-dynamic programming and approximate dynamic programming (ADP) that perhaps have built on the foundations laid out in my book.

3.3 Conclusions

Since completing the book, I have continued my research on MDP theory and application. With my extensive background in statistical modelling and application, I focused on problems which explored the relationship between optimization and estimation. Ding, Puterman, and Bisi (2002) investigated the trade-off between dynamic parameter estimation and control in a dynamic newsvendor setting while Carvalho and Puterman (2005) studied this trade-off in a dynamic pricing environment. Recently I have begun investigating using MDP methods to address health care management problems. In Patrick, Puterman, and Queyranne (2007), we use ADP to determine an approximately optimal policy for dynamic stochastic multi-priority patient scheduling. My current research focuses on extending this work as well as using MDPs methods to enhance cancer care delivery.

I anticipate that the MDP field will continue to expand, and that MDP models will become more broadly applied. Applications in telecommunications, information search and health care management as well as new theoretical challenges in approximate dynamic programming will move the field forward and attract many new researchers. I feel that I entered the field when many basic research challenges presented themselves. Good advice from my Stanford advisor Pete Veinott and our mutual love for mathematical elegance and rigor inspired my research. I am grateful for this foundation as well as the opportunity to have worked on MDP problems with several outstanding Ph.D. students.

4. RANDOM NUMBER GENERATION

In this section, we move away from Markov modeling to a discussion of the basic building blocks needed to analyze general stochastic systems via simulation, namely, random number generators.

When I (PL) received my Ph.D. in 1983, I had never taken a course on simulation and I had no clue about how computers were generating random numbers. My thesis was on approximate dynamic programming for Markov decision processes with continuous-state spaces, a topic that regained much attention recently, especially in the area of machine learning. Back then, I was reading the books of Bertsekas and the papers of Puterman and others, under the guidance of Alain Haurie. I was studying DP in general abstract settings on the one hand, and was developing solution methods based on finite element approximations of the value function or the policy on the other hand.

About a year into my first job as a computer science professor at Université Laval, in Québec City, I was asked to teach simulation. This is when I started learning about it, one chapter ahead of the students. The Random Number Generation (RNG) chapter was still far ahead when a colleague who was developing educational computer games came to me for help. The default *Microsoft Basic* RNG was giving him great trouble and I was supposed to be the local expert on simulation and RNGs. In one of the games, a character on the screen repeatedly moved in one of four possible directions at random. The program used a random 16-bit integer from the Basic generator to choose the direction at each step, making use of the two least significant bits, using a "modulo 4" operation. The problem was that the successive pairs of bits followed an easily predictable pattern, making the computer game meaningless. Instead of admitting my zero knowledge about RNGs, I rushed to read the book chapter and other relevant references, and to find out what algorithm was used by the Basic compiler. As it turned out, it was a simple linear congruential generator (LCG), based on a recurrence of the form $x_i = ax_{i-1} \bmod m$, whose modulus m was a power of 2. It is well known that for such generators, the least significant bits of the successive values of x_i are periodic with a very short period, and this was obviously the source of the problem. I was astonished that such a simplistic and grossly defective RNG was employed in such highly-popular commercial software, and viewed it as an easy opportunity to make a useful contribution.

My initial goal was just to cook up and implement a better generator for 16-bit computers. But that led me deeper into number theory, geometry of numbers, statistical testing, and efficient implementation tricks for RNGs. I found these fields quite interesting and I got hooked. About a year later, I had a Pascal code for a reasonably quick and portable RNG for 32-bit computers, with period length near 2^{61} . This is one billion times larger than the period of LCGs (with modulus $2^{31} - 2$) used in most simulation software at the time. The new generator combined two LCGs with distinct prime moduli m_1 and m_2 , each selected based on a spectral analysis of its lattice structure, and subtracted their states modulo m_1. It had a much more robust behavior in empirical statistical tests than the commonly used LCGs.

I showed this to Bennett Fox, a true expert on simulation, who had chaired my Ph.D. thesis committee at the Université de Montréal. By an extraordinary coincidence, that same month, Ben was organizing an invited session on RNGs for the 1986 Winter Simulation Conference (WSC), the world's primary yearly event on discrete-event simulation. He liked my generator and this became my first WSC paper (L'Ecuyer 1986). There, I received enthusiastic encouragement from RNG experts such as Luc Devroye, George Fishman, Ulrich Dieter, and Richard Nance, for example, and that was the beginning of my long and exciting journey in the world of simulation research.

INFOR, Vol. 46, No. 1, February 2008, pp. 3–14
ISSN 0315-5986 | EISSN 1916-0615
DOI 10.3138/infor.46.1.3

In 1988, a polished and expanded version of this work appeared in the widely-distributed *Communications of the ACM* (L'Ecuyer 1988), which published research articles at the time. The generator made its way to several books, software, and even pocket calculators. I went on to work on RNGs based on higher-order linear recurrences, of the form $x_n = (a_1 x_{n-1} + \cdots + a_k x_{n-k}) \bmod m$, with output $u_n = x_n/m$ (or a variant of this to avoid returning 0), and presented that at the 1988 WSC. These RNGs, known as *multiple recursive generators* (MRG), can have much longer periods and stronger statistical behavior than the LCGs.

The next year, Shu Tezuka, from IBM Tokyo, noticed that the pairs of successive numbers produced by my 1986 combined generator were very close to the pairs produced by a particular (large) LCG with modulus $m_1\ m_2$. This approximation property also turned out to be true for vectors of successive output values in an arbitrary number of dimensions. Tezuka's observation was important for the following reason. A key aspect in the quality of a uniform RNG is the uniformity of the set Ψ_t of all vectors of t successive output values produced by the RNG, from all possible initial states, for $t = 1,2,3, \ldots$. For a random initial state (or seed) the RNG returns a vector uniformly distributed over Ψ_t as an approximation of a uniformly distributed vector over $[0,1)^t$. For the approximation to be good, Ψ_t must cover the unit hypercube very evenly. For this, a large cardinality of Ψ_t (which usually means a long period for the RNG) is necessary, but not sufficient. For certain types of RNGs based on linear recurrences, including the LCG and MRG, the uniformity is assessed via quantitative measures that can be computed without generating the points, by exploiting the mathematical structure of Ψ_t (L'Ecuyer 1994, 2006). In the case of LCGs and MRGs, Ψ_t turns out to be the intersection of a lattice with the unit hypercube (Knuth 1998, L'Ecuyer 2006). Tezuka's remark meant that the uniformity of the combined LCGs could be measured by analyzing the lattice structure of the approximating LCG. We published a joint paper that studied the basic theory for two classes of combined LCGs and their approximation by a single LCG, with explicit bounds on the differences in their output (L'Ecuyer and Tezuka 1991).

In L'Ecuyer (1996a), I generalized this theory to combined MRGs, and went on to construct specific high-quality combined MRG in L'Ecuyer (1999a). One of them, named `MRG32k3a`, is now used in a large variety of software tools for simulation, statistics, finance and risk analysis, gaming machines, etc. Combined MRGs can be motivated as efficient ways of implementing (approximately) MRGs with large moduli and robust statistical behavior, as I now explain. Fast implementations of simple MRGs are available when several of the coefficients a_j are zero and the others are small, but in that case, the MRG has too much structure and poor statistical behavior. As an illustration, one type of widely-used MRG (even today) is based on a recurrence of the form $x_n = (x_{n-r} \pm x_{n-k}) \bmod m$ for some integers $k > r > 0$. It turns out that all triples of output values of the form (u_{n-k}, u_{n-r}, u_n) produced by such a generator lie in only two planes in the three-dimensional unit cube (L'Ecuyer 1997). This is definitely very far from uniform! Another class of long-period generators called add-with-carry and subtract-with-borrow, also widely-used, have essentially the same property. This, and many other less obvious structural deficiencies of popular generators, were uncovered in Couture and L'Ecuyer (1994, 1996, 1997) and Tezuka *et al*. (1994). Combination is one effective way of obtaining fast and robust generators. The idea is to select the components so that a fast implementation is available for them, while the MRG that approximates the combination has the best possible uniformity in terms of the lattice structure of its sets Ψ_t, as measured by the theoretical figures of merit. The parameters are selected by extensive computer searches.

Similar ideas were developed for RNGs based on linear recurrences modulo 2, which can run very fast on binary computers because they can be implemented with a few simple binary operations such as shift, rotation, bitwise or, and exclusive-or, on blocks of bits (L'Ecuyer and Panneton 2007). Again, one can combine simple components that run fast, in a way that the combination has very good uniformity and (of course) a long period. The theory underlying this was studied in Tezuka and L'Ecuyer (1991), L'Ecuyer (1996b), and tables of good parameters together with specific implementations were given in L'Ecuyer (1999b). More recently, in Panneton *et al*. (2006), we studied and constructed generators with extremely long periods (up to $2^{44497} - 1$) and excellent uniformity, also based on linear recurrences modulo 2.

A important concept that I have kept advocating for many years in stochastic simulation is the need for RNGs with multiple streams and substreams. One should be able to declare and create RNGs just like any other type of variable or object in a programming language. These virtual RNGs, often called *streams*, produce a very long sequence of random numbers, and can themselves be split into *substreams*, which are long enough to preclude any potential overlap. For any stream, one should be able to generate the next number, to rewind the stream to its starting point, or to the beginning of the current substream, or to the beginning of the next substream. An early version of this was proposed by L'Ecuyer and Côté (1991), with a restricted number of streams. A more recent implementation that provides a practically unlimited number of streams and substreams (L'Ecuyer *et al*., 2002) has been adopted recently in high-profile software such as SAS, Arena, Witness, Simul8, and Automod, among others. The streams and substreams are implemented from a single backbone generator, by cutting its sequence into several long disjoint pieces. This requires efficient algorithms to jump ahead quickly by an arbitrary number of steps in the sequence (L'Ecuyer *et al*., 2002, L'Ecuyer 2006, Haramoto *et al*., 2007).

The usefulness of these streams and substreams can be illustrated with a small example. Suppose we want to estimate $\nu = \mathbb{E}[X_2 - X_1]$, where X_1 and X_2 are the performances of two

ISSN 0315-5986 | EISSN 1916-0615 DOI 10.3138/infor.46.1.3

similar systems; for example, a telephone call center with two slightly different numbers of agents to answer the calls, or a manufacturing system in which some conveyors or machines have slightly different speeds. This type of estimation problem often occurs in sensitivity analysis and in simulation-based optimization. We would like to estimate ν by simulating the two similar systems with *common random numbers* (the same uniform random numbers used for the same purpose for both systems, whenever possible, to minimize the noise in the difference), repeat this n times independently, and compute a confidence interval on ν from the n independent copies of $X_2 - X_1$. Sometimes, a random number used for one system is not needed in the other system (for example, if a customer receives service in one case and abandons in the other case). To make sure that the same random numbers are used for the same purpose, we would devote one stream to each type of random number used in the simulation (e.g., one for service times and one for interarrival times in a queueing system, one for the size of orders in an inventory system, etc.). For each stream, we use n different substreams for the n replications. At the beginning of a replication, each stream is placed to the beginning of a new substream and the model is simulated to compute X_1. Then each stream is reset to the beginning of its current substream and we simulate the model again to compute X_2. This ensures that exactly the same sequences of random numbers, for each type, are used for both X_1 and X_2. Each stream is then moved to the beginning of the next substream for the next pair of runs.

4.1 Conclusions

My involvement with RNGs started with a very lucky coincidence. By yet another coincidence, I am writing these lines while attending the 2007 WSC, 21 years later, in exactly the same city (Washington D.C.) as the 1986 WSC. To celebrate the 40th anniversary of this conference, a special committee has selected 10 *landmark papers* that had an important impact on the field, among the nearly 10,000 papers published in the conference proceedings during those 40 years, and a special celebration was organized to honor their authors. Amazingly, my 1986 paper is among the ten! That paper was actually very simple and I had no idea at that time that it could lead me that far. But it was the seed of all my subsequent work and contributions to RNGs. In that sense, it has certainly been a landmark of my scientific life.

In retrospective, my journey was successful mainly because it started as an attempt to satisfy a concrete need for a better algorithm and to provide an efficient software implementation. The user's viewpoint defined the target. At the same time, I was ready to devote the required efforts to understand the underlying mathematics, especially after my 1986 paper. Most researchers on RNGs can be classified as either theoreticians who study mathematical properties without paying much attention to concrete implementations, or hackers who pay minimal attention to the theory (they do not take time to study it) and concentrate on clever programming tricks. Over the past two decades, it was very important to me to stay at the leading edge in both the theory and implementation aspects, and I have been one of the very few researchers on RNGs to do so. This alliance of theory with practice has been the key to my success.

This story only gives a partial and oversimplified view of my work on RNGs. It skips several other exciting topics: randomized quasi-Monte Carlo methods (for which the tools are actually very similar to those used for RNG analysis), variance reduction techniques, rare-event simulation, gradient estimation and stochastic optimization by simulation, and applications in various areas such as manufacturing, finance, telecommunications, and call center management. To learn more about this, the reader is invited to browse through the research publications available at http://www.iro.umontreal.ca/∼lecuyer/.

5. QUEUEING THEORY SOFTWARE

In this final section, we move back to Markov modeling, and discuss barriers to the widespread use of Markovian queueing models in practice and how software might lower some of those barriers.

Markovian queues in steady state (notable the $M/M/c$ and $M/M/c/n$ models) are among the most widely used stochastic models in operational research. These models have been used by telephone engineers since they were developed by Erlang in the early part of the twentieth century, and later for the analysis of telecommunication, computer, manufacturing, and service systems. They are beginning to see use in health care systems modeling. Many engineering and computer science students and most business students learn about these basic models.

And yet, despite the long list of areas where these results have been found useful and the large number of students that have learned about them, I (AI) often see or hear about situations where simple queueing analysis could have been valuable but was not used. I am sure that other operational researchers have had similar experiences, and not just with queueing analysis. However, I think the barriers against use are more extreme for queueing theory than other OR methods, such as linear programming or simulation, for the following reasons:

Higher level of abstraction: It is more difficult to explain how a Markov chain works and how it can model reality than, say, a linear program or a simulation. The concept of "steady state" is especially difficult to explain.

Assumptions that are perceived as limiting: Those who do grasp the concepts of memoryless holding times and limiting state probabilities often dismiss queueing results as irrelevant because the assumptions appear unrealistic. It is also common to assume that queueing theory is limited to steady state results. Many practitioners and academics are not aware of the relative ease with which transient probabilities can be computed for a variety of realistic systems, as discussed in Section 2.

ISSN 0315-5986 | EISSN 1916-0615 DOI 10.3138/infor.46.1.3

Difficulty in performing calculations: Even when practitioners do see the potential for using queueing analysis, they may run into difficulties in computing the numerical answers they need, if they lack training in programming and numerical analysis, or the time needed to write their own code.

I developed the Queueing ToolPak (QTP, Ingolfsson and Gallop 2003) in an attempt to lower at least the last of these barriers. QTP is a free spreadsheet add-in for Microsoft Excel that provides a set of functions to compute such quantities as average waiting times or the minimum number of servers needed to achieve a given service level. I chose to develop the software as a spreadsheet add-in because with current technology, electronic spreadsheets are probably the computing platform that the largest number of people have easy access to and are familiar with. With familiarity come expectations, for example, that computations in spreadsheets are updated automatically when inputs are changed, and the QTP functions conform with that expectation. This is somewhat different from the way that academics often think of computations, as requiring one to "run a program," which in software design often gets translated into requiring the user to "click a button" to perform calculations. By designing the QTP functions to work the same way as other spreadsheet functions work, my hope was that users would find them easier to use and easier to incorporate as building blocks in larger models. Initially, I developed QTP for educational purposes and it is used for teaching at many universities. However, over time I found that practitioners would use QTP as a component to develop customized spreadsheets for their clients and even academics would use it for research purposes.

One thing I discovered when developing QTP was that although formulas for $M/M/c/n$ were developed almost a century ago and they appear in countless introductory textbooks, it is not obvious how to implement computational algorithms that work reliably across a wide range of input parameter values, particularly system size, as measured by the number of servers. The primary reason that I continued to develop QTP beyond version 1.0 was that users contacted me about situations where a particular set of input parameters caused the functions to return errors. Readers who are evaluating queueing software or developing their own code for queueing calculations may find the following test case useful: an $M/M/c$ queue with a service rate μ of 1, number of servers $s = x$, and an arrival rate $\lambda = 0.9x$. Many software packages fail when x exceeds 750, including the first version of QTP. Researchers have recognized these numerical difficulties and partly because of them have developed asymptotic approximations for relatively simple Markovian queueing models such as the $M/M/c$ queue (Halfin and Whitt 1981) and the $M/M/c + M$ queue (Garnett *et al*., 2002), where customers renege after an exponentially distributed amount of time. In recent work, I have focused on developing a general algorithm to reliably compute performance measures for birth-death processes with limiting state probabilities that decay geometrically or faster—a class that includes the $M/M/c$, $M/M/c + M$, and many other models. The basic ideas are to compute the probabilities recursively, to start the recursion at a state that one expects a priori to have relatively high probability, to rescale the probabilities so that they add up to one after every iteration, and to use error bounds to truncate the upper and lower tails of the state space.

In conclusion, to echo comments from the previous section, I consider it important to understand the needs of users and potential users of stochastic models and to take advantage of that understanding to increase the impact of our field, whether it is through advances in theory, computation, or software design.

REFERENCES

Blackwell, D. 1962. Discrete dynamic programming. *Annals Mathematical Statistics* 3, 719–726.

Carrasco, J. A. and Calderon, A. 1995. Regenerative randomization: Theory and application examples. *Performance Evaluation Review* 23, 23–241.

Carvalho, A. and Puterman, M. L. 2005. Learning and pricing in an internet environment with binomial demands. *Journal of Revenue and Pricing Management* 3, 320–336.

Ciardo, G. 2007. SMART, a stochastic model checking analyser for reliability and timing. http://www.cs.urs.edu/~ciardo/SMART. Accessed June 11 2007.

Couture, R. and L'Ecuyer, P. 1994. On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. *Mathematics of Computation* 62, 798–808.

Couture, R. and L'Ecuyer, P. 1996. Orbits and lattices for linear random number generators with composite moduli. *Mathematics of Computation* 65, 189–201.

Couture, R. and L'Ecuyer, P. 1997. Distribution properties of multiply-with-carry random number generators. *Mathematics of Computation* 66, 591–607.

Daigle, J. N. 1992. *Queueing Theory for Telecommunication*. Addison-Wesley, Reading, MA.

de Souza e Silva, E. and Gail, H. R. 1986. Calculating cumulative operational time distributions of repairable computer systems. *IEEE Transactions on Computers* 35, 322–332.

de Souza e Silva, E. and Gail, H. R. 1989. Calculating availability and performability measures of repairable computer systems using randomization. *Journal of the ACM* 36, 171–193.

de Souza e Silva, E. and Gail, H. R. 2000. *Transient solutions for Markov chains*. Grassmann, W. K., ed., Computational Probability. International Series in Operations Research and Management Science, Kluwer, Boston, MA, 43–79.

Denardo, E. V. 1967. Contraction mappings in the theory underlying dynamic programming. *SIAM Review* 9, 165–177.

Ding, X., Puterman, M. L., and Bisi, A. 2002. The censored newsvendor and the optimal acquisition of information. *Operations Research* 50, 517–527.

Feller, William. 1971. *An Introduction to Probability Theory and its Applications*, vol. 2. 2nd ed. Wiley, New York.

Ferreira, F. and Pacheco, A. 2006. Analysis of the $GI/M/s/c$ queues using uniformization. *Computers and Mathematics with Applications* 51, 291–304.

DOI 10.3138/infor.46.1.3

Garnett, O., Mandelbaum, A., and Reiman, M. I. 2002. Designing a call center with impatient customers. *Manufacturing & Service Operations Management* 4, 208–227.

Grassmann, W. K. 1977a. Transient solutions in Markovian queueing systems. *Computers & Operations Research* 4, 47–53.

Grassmann, W. K. 1977b. Transient solutions in Markovian queues. *European Journal of Operational Research* 1, 396–402.

Grassmann, W. K. 1982. The *GI/PH/1* queue: A method to find the transition matrix. *INFOR* 20, 144–156.

Grassmann, W. K. 1987. Means and variances of time averages in Markovian environments. *European Journal ofOperational Research* 31, 132–139.

Grassmann, W. K. 1991. Finding transient solutions in Markovian event systems through randomization. Stewart, W. J., ed., Numerical Solutions of Markov Chains. Marcel Dekker, New York.

Grassmann, W. K. 1993. Rounding errors in certain algorithms involving Markov chains. *ACM Transactions on Mathematical Software* 19, 496–508.

Grassmann, W. K. and Luo, J. 2005. Simulating Markov-reward processes with rare events. *ACM Transactions on Modeling and Computer Simulation* 15, 138–154.

Grassmann, W. K. and Ngai, T. K. 1971. Program description for transient solutions in continuous Markov chains. SHARE Program Library Agency, Program 360D-15.0.005, Research Triangle Park, North Carolina.

Grassmann, W. K., Taksar, M. I., and Heyman, D. P. 1985. Regenerative analysis and steady state distributions for Markov chains. *Operations Research* 33, 1107–1116.

Gross, D. 1980. Personal communication.

Gross, D. and Harris, C. M. 1998. *Fundamentals of Queueing Theory*. 3rd ed. Wiley, New York.

Gross, D. and Miller, D. R. 1984. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research* 32, 343–361.

Grossman, T. A. 2002. Spreadsheet add-ins for OR/MS. *OR/MS Today* 29(4) 46–51.

Halfin, S. and Whitt, W. 1981. Heavy-traffic limits for queues with many exponential servers. *Operations Research* 29, 567–587.

Haramoto, H., Matsumoto, M., Nishimura, T., Panneton, F., and L'Ecuyer, P.. 2007. Efficient jump ahead for F_2-linear random number generators. *INFORMS Journal on Computing* (to appear).

Heyman, D. and Sobel, M. J. 1990. *Handbooks in Operations Research and Management Science, Volume 2: Stochastic Models*. North Holland, Amsterdam.

Howard, R. A. 1960. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA.

Ingolfsson, A. and Gallop, F. 2003. Queueing toolpak 4.0. http://www.business.ualberta.ca/aingolfsson/qtp/, accessed Jan. 21, 2008.

Ingolfsson, A. E., Akhmetshina, E., Budge, S., Li, Y., and Wu, X. 2007. A survey and experimental comparison of service level approximation methods for non-stationary $M/M/s$ queueing systems. *INFORMS Journal of Computing* 19, 201–214.

Jain, J. L., Mohanti, S. G., and Bohm, W. 2007. *A Course on Queueing Models*. Chapman and Hall, Boca Raton, Florida.

Jensen, A. 1953. Markoff chains as an aid in the study of Markoff processes. *Skandinavisk Aktuar-ietidskrift* 36, 87–91.

Kalaba, R. 1959. On the nonlinear differential equations, the maximum operator and monotone convergence. *Journal of Mathematics and Mechanics* 9, 519–574.

Kao, E. P. C. 1997. *An Introduction to Stochastic Processes*. Duxbury Press, New York.

Knuth, D. E. 1998. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. 3rd ed. Addison-Wesley, Reading, MA.

L'Ecuyer, P. 1986. Efficient and portable 32-bit random variate generators. *Proceedings of the 1986 Winter Simulation Conference*. 275–277.

L'Ecuyer, P. 1988. Efficient and portable combined random number generators. *Communications of the ACM* 31 742-749 and 774. See also the correspondence in the same journal, 32 (1989) 1019–1024.

L'Ecuyer, P. 1994. Uniform random number generation. *Annals of Operations Research* 53, 77–120.

L'Ecuyer, P. 1996a. Combined multiple recursive random number generators. *Operations Research* 44, 816–822.

L'Ecuyer, P. 1996b. Maximally equidistributed combined Tausworthe generators. *Mathematics of Computation* 65, 203–213.

L'Ecuyer, P. 1997. Bad lattice structures for vectors of non-successive values produced by some linear recurrences. *INFORMS Journal on Computing* 9, 57–60.

L'Ecuyer, P. 1999a. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research* 47, 159–164.

L'Ecuyer, P. 1999b. Tables of maximally equidistributed combined LFSR generators. *Mathematics of Computation* 68, 261–269.

L'Ecuyer, P. 2006. Uniform random number generation. Henderson, S. G. and Nelson, B. L., eds., *Simulation*. Handbooks in Operations Research and Management Science, Elsevier, Amsterdam, 55–81.

L'Ecuyer, P. and Côté, S. 1991. Implementing a random number package with splitting facilities. *ACM Transactions on Mathematical Software* 17, 98–111.

L'Ecuyer, P. and Panneton, F. 2007. F2-linear random number generators. GERAD Report 2007-21 (submitted for publication).

L'Ecuyer, P., Simard, R., Chen, E. J., and Kelton, W. D. 2002. An object-oriented random-number package with many long streams and substreams. *Operations Research* 50, 1073–1075.

L'Ecuyer, P. and Tezuka, S. 1991. Structural properties for two classes of combined random number generators. *Mathematics of Computation* 57, 735–746.

Lock, S. W. K. 1988. Some experimental designs for determining run-lengths in simulation. Master's thesis, Department of Computer Science, University of Saskatchewan, Saskatoon, Canada.

Love, C. E. 1977. Purchase/replacement rules for decaying service facilities. *Computers & Operations Research* 4, 111–118.

Mandl, P. 1968. *Analytic Treatment of One-Dimensional Markov Processes*. Springer-Verlag, New York.

Melamed, B and Yadin, M. 1984a. Numerical computations of sojourn-time distributions in queueing networks. *Journal of the ACM* 31, 839–854.

Melamed, B. and Yadin, M. 1984b. Randomization procedures in the computation of cumulative-time distributions over discrete state Markov processes. *Operations Research* 32, 926–944.

Miller, D. R. 1983. Reliability calculations using randomization for Markovian fault-tolerant computing systems. *13th Annual*

DOI 10.3138/infor.46.1.3

International Symposium on Fault-Tolerant Computing. IEEE Computer Society, Silver Springs, MD.

Oni, O. A. 2003. Initial bias in the simulation of Markovian event systems. Master's thesis, Department of Computer Science, University of Saskatchewan, Saskatoon, Canada.

Panneton, F., L'Ecuyer, P., and Matsumoto., M. 2006. Improved long-period generators based on linear recurrences modulo 2. *ACM Transactions on Mathematical Software* 32, 1–16.

Patrick, J., Puterman, M. L., and Queyranne, M. 2007. Dynamic multi-priority patient scheduling for a diagnostic resource. *Operations Research* (under review).

Powell, W. 2007. *Approximate Dynamic Programming*. Wiley, New York.

Puterman, M. L. 1974. Sensitive discount optimality in controlled one dimensional diffusions. *Annals of Probability* 2, 408–419.

Puterman, M. L. 1977. Optimal control of diffusion processes with reflection. *Journal of Optimization Theory and Its Applications* 22, 103–115.

Puterman, M. L. 1990. Markov decision processes. Heyman, D. P. and Sobel, M. J., eds., *Handbooks in Operations Research and Management Science*, *Volume 2*: *Stochastic Models*. North Holland, Amsterdam, 331–434.

ISSN 0315-5986 | EISSN 1916-0615 DOI 10.3138/infor.46.1.3