Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

INVERSE-TRANSFORMATION ALGORITHMS FOR SOME COMMON STOCHASTIC PROCESSES

Bruce Schmeiser
Whey-Ming Tina Song
School of Industrial Engineering
Purdue University
West Lafayette, IN 47907, U.S.A.

ABSTRACT

Realizations from common stochastic processes are often used by simulation-methodology researchers in Monte Carlo performance evaluation of new and existing methods for output analysis, variance reduction, and optimization. Typically realizations can be obtained easily from either the definition or simple properties of the process. We discuss using the inverse of the distribution function for generating realizations from some of these processes. The inverse transformation always possesses the advantage of correlation induction, useful for variance reduction. We consider the discrete-time processes ARMA, EAR, M/M/1-QT (time in queue), and M/M/1-ST (time in system, the sojourn time), and Markov chains. The inverse-transformation algorithms are sometimes slower (e.g., ARMA, M/M/1-ST), sometimes faster (e.g., M/M/1-QT), and often about the same speed as the usual algorithm. Some Fortran implementations are provided.

1. INTRODUCTION

Inverting the distribution function with an argument that is assumed to be a uniform (0,1) random number is a classic random-variate generation method. The inverse transformation is one-to-one and monotonically increasing, which makes it ideal for creating simulation estimators that are correlated by using the same random numbers for different simulation runs. Both positively and negatively correlated estimators are useful for reducing variance, as discussed in many textbooks.

The inverse transformation for scalar random variates is straightforward. Given a random number u, the inverse transformation is $F_X^{-1}(u) = \min\{x | u \le F_X(x)\}$, where the choice of F is a modeling decision. For many classical distributions the inversion is closed form, making implementation straightforward. Numerical methods exist for the normal, gamma, beta, and other classical distributions that lack a close-form inverse. (Devroye 1986, Schmeiser 1980). Schmeiser and Kachitvichyanukul (1986, 1989) and Kachitvichyanukul, Cheng and Schmeiser (1988) have investigated random-variate-generation methods that are nearly the inverse transformation yet almost as fast as state-of-the-art methods.

We discuss the inverse transformation for generating stochastic processes, focusing on some processes commonly used in Monte Carlo studies. For example, Song and Schmeiser (1989, Section 3.1) discuss AR(1), EAR1, and two-state Markov chain models to obtain a specified process mean, variance of the sample mean for sample size n, and sum of autocorrelations $\gamma_0 = \sum_{h=-\infty}^{\infty} \rho_h$. In this and other work we have found it useful use both common random numbers and antithetic variates, for which we need inverse-transformation algorithms.

The concept is simple: derive and invert, either analytically or numerically, the distribution function of the next observation X_i conditional on the past observations $X_1, X_2, \cdots, X_{i-1}$. We focus on Markov processes, since having to consider only recent observations simplifies the problem. In Sections 2 through 6 we consider the discrete-time processes ARMA, EAR, M/M/1 wait-time in queue, M/M/1 sojourn time, and Markov chains.

We provide subroutines for the AR(1), EAR(1), M/M/1 wait-time in queue, M/M/1 sojourn time, and two-state Markov chains. In our implementations, the subroutine returns the next observation as a function of the process parameters and the previous observation. If the previous observation is infeasible, then the subroutine generates the next observation from the steady-state distribution. Thus initializing the previous value to an infeasible value and calling the routine repeatedly yields a sequence of steady-state observations. Alternatively, transient effects can be studied by sampling the initial value from the (possibly degenerate) distribution of interest.

2. ARMA TIME SERIES

Consider the ARMA (p,q) process

$$X_i = \mu_i + \sum_{h=1}^{p} \phi_h (X_{i-h} - \mu_{i-h}) + \sum_{h=1}^{q} \theta_h E_{i-h} + E_i$$

where we allow μ_i and the distribution of E_i to vary with time but assume that the error terms E_i are from a process that is independent of past error terms and past observations. The inverse transformation of X_i conditional on the previous observations and errors is obtained by simply substituting $F_{E_i}^{-1}(U_i)$ for E_i, where U_i is the i^{th} random number. Thus, any time-series process defined by a deterministic relationship to the past modified by an independent error term can be generated in this way. Barone (1987) discusses steady-state initialization for ARMA processes, including the multivariate extension.

Steady-state initialization is trivial for the AR(1) process with normal marginal distribution, as illustrated in the subroutine *par*1. In this routine the error terms are generated with IMSL's subroutine anorin, but other numerical approximations could be used, such as the rough but convenient one from Ramberg and Schmeiser (1972) mentioned in the program's comments.

```
      subroutine par1 (xmean, xsd, phi, iseed, x)
c.....bruce schmeiser and tina song
c     july 1989
c     purdue university
c.....purpose:
c     generate one observation from a first-order
c       autoregressive time series with normal marginal
c       distribution.
c.....method:
c     if the last observation is more than ten standard
c       deviations from the mean, then this observation
c       is generated from the steady-state marginal
c       distribution. otherwise, this observation is
c       generated from the conditional distribution
c       given the previous observation, using the
c       inverse transformation.
c.....input
c       xmean: process mean
c       xsd:   process standard deviation
c              (forced to be nonnegative)
c       phi:   process lag-1 autocorrelation
c              (forced into [-1, 1])
c       iseed: random-number seed
c       x:     previous observation
c.....output
c       iseed: random-number seed
c       x:     this observation
c.....other routines used
c       rand:  a u(0,1) random-number generator
c       anorin: imsl's standard-normal inverse
c         transformation. a reasonable quick
c         approximation is
c             anorin = (u**.135 - (1-u)**.135) / .1975
      u = rand(iseed)
      z = anorin(u)
      if (xsd .lt. 0.) xsd = -xsd
      if (abs(x-xmean) .gt. 10.*xsd) then
c         ...generate from the steady-state distribution...
        x = xmean + (xsd*z)
      else
c         ...generate from the conditional distribution...
        c = 1. - phi*phi
        if (c .lt. 0.) then
          if (phi .le. -1.) x = xmean - (x - xmean)
        else
          x = xmean + phi*(x-xmean) + (xsd*sqrt(c)*z)
        endif
      endif
      return
      end
```

3. EAR(1) TIME SERIES

Now consider the EAR(1) process

$$X_i = \phi X_{i-1} + B_i E_i$$

where E_i is from a sequence of iid exponential random variables with mean μ and B_i is from an independent sequence of Bernoulli random variables with probability of success ϕ. The marginal distribution of X_i is exponential with mean μ and the h-lag autocorrelation is ϕ^h. See Lewis (1980).

Given the previous observation, x_{i-1}, the next observation can be obtained by generating E_i and B_i and applying the definition. This is straightforward using the inverse transformation of each: $E_i = -\mu \ln(1-U_i)$ and $B_i = I(V_i \geq 1)$, where I(.) denotes the indicator function. But this approach requires two random numbers, U_i and V_i, and is therefore not the inverse transformation.

The distribution function of X_i conditional on $X_{i-1} = x_{i-1}$ is zero to the left of ϕx_{i-1}, has a jump of height ϕ at ϕx_{i-1}, and is $\phi + (1-\phi)[1 - e^{(-(x_i - x_{i-1})/\mu)}]$ to the right of ϕx_{i-1}. The inverse transformation is then $x_i = \phi x_{i-1}$ if $u_i \leq \phi$ and $x_i = \phi x_{i-1} - \mu \ln(1-u_i)$ if $u_i > \phi$.

The inverse transformation differs from the first method only in that after the first random number, u_i, is used to generate the Bernoulli observation it is rescaled and used again to generate the exponential error term. This type of rescaling is commonly used in random-variate generation to reduce the number of required random numbers.

The subroutine *pear*1 implements the inverse transformation. If the previous value is negative, then the new value is generated from the steady-state distribution. The mean, *xmean*, must be nonnegative and the lag-1 autocorrelation, *phi*, must lie in [0,1].

```
      subroutine pear1 (xmean, phi, iseed, x)
c.....bruce schmeiser and tina song
c     july 1989
c     purdue university
c.....purpose:  generate one ear(1)-process observation
c
c      ear(1):  x(i) = phi * x(i-1)          w.p. phi
c                    = phi * x(i-1) + e(i)   w.p. 1-phi,
c               where e(i) is exponential
c
c     method:  inverse transformation
c.....input
c      xmean:  process mean
c      phi:    lag-1 autocorrelation
c      iseed:  random-number seed
c      x:      previous observation
c.....output
c      iseed:  random-number seed
c      x:      next observation
c.....other routine used
c      rand:  random-number generator
c
      u = rand(iseed)
      if (x .lt. 0.) then
c        ...generate from the steady-state distribution...
         x = xmean * (- alog(1.-u))
      else
c        ...generate conditional on the previous x
         if (u .le. phi) then
            x =  phi*x
         else
            u = (u-phi) / (1.-phi)
            x = (phi*x)  + (xmean * (-alog((1.-u))))
         endif
      endif
      return
      end
```

4. M/M/1 QUEUE WAITING TIMES

Consider the waiting time in queue, W_i, for the i^{th} customer in a single-queue single-server model. The classic method for generating a sequence of dependent W's is to use the FIFO recursion

$$W_i = \max(0,\ W_{i-1} + S_{i-1} - A_i)\ ,$$

where S_i and A_i are the service and interarrival times of the i^{th} customer. Two random variates are needed to generate each successive W in straightforward application of the FIFO recursion.

We want to generate each waiting time using a single random number via the inverse transformation conditional on the value of the previous waiting time. Now other than the previous waiting time, the next waiting time is dependent only upon the difference of S_{i-1} and A_i, which are independent. So the desired algorithm is to generate X_i, the difference between S_{i-1} and A_i, from the inverse transformation F_X^{-1}, and set $W_i = \max(0,\ W_{i-1} + X_i)$. Since the X process is i.i.d., we suppress subscripts for convenience.

For M/M/1 models with arrival rate λ and service rate μ we substitute exponential interarrival-time and service-time distributions to obtain the M/M/1-QT distribution function

$$F_X(x) = \mathrm{P}\{S - A \le x\} = \begin{cases} \left(\dfrac{\mu}{\lambda+\mu}\right)e^{\lambda x} & \text{if } x \le 0 \\ 1 - \left(\dfrac{\lambda}{\lambda+\mu}\right)e^{-\mu x} & \text{if } 0 \le x\ , \end{cases}$$

which can be found by conditioning on either S or A. The inverse transformation is

$$F_X^{-1}(u) = \begin{cases} \lambda^{-1}\ \ln(u(\lambda+\mu)/\mu) & \text{if } u \le \mu/(\lambda+\mu) \\ -\mu^{-1}\ \ln((1-u)(\lambda+\mu)/\lambda) & \text{if } u \ge \mu/(\lambda+\mu)\,. \end{cases}$$

Random variates are obtained by generating the argument u as a U(0,1) random number.

Subroutine *pmm*1*qt* generates the next queue waiting time from the M/M/1 queueing system with arrival rate *arate* and service rate *srate* using the inverse transformation. If the last wait time is negative, then the new wait time is from the steady-state distribution function $F_W(w) = 1 - (\lambda/\mu)e^{-(\mu-\lambda)w}$ for $w \ge 0$.

```
      subroutine pmm1qt (arate, srate, iseed, waitq)
c.....bruce schmeiser and tina song
c      september 1988
c      purdue university
c.....purpose:
c      generate the next waiting time (in the queue)
c        from an m/m/1 queue conditional upon the
c        previous waiting time
c.....method:
c        inverse transformation
c.....input
c        arate: the arrival rate
c        srate: the service rate
c        iseed: random-number seed
c        waitq: if nonnegative, the last wait time
c               if negative, generate from steady-state
c.....output
c        iseed: random-number seed
c        waitq: the generated waiting (in queue) time
c.....intermediate variable
c        x:    service time - interarrival time
c.....other routine used
c        rand:  uniform (0,1) random-number generator
c
      u = rand(iseed)
      if (waitq .lt. 0.) then
c
c        ...generate a steady-state waiting time...
c
         tau = arate / srate
         if (u .lt. 1.-tau) then
            waitq = 0.
         else
            waitq = - alog((1.-u)/tau) / (srate - arate)
         endif
      else
c
c        ...generate conditional on the previous waitq...
c
         ratio = srate / (srate + arate)
         if (u .lt. ratio) then
            x =  alog (  u /   ratio ) / arate
         else
            x = - alog ((1.-u)/(1.-ratio)) / srate
         endif
         waitq  = waitq + x
         if (waitq .lt. 0.)  waitq = 0.
      endif
      return
      end
```

Since it requires only a single logarithm evaluation, the inverse transformation for the M/M/1-QT is faster than the usual method of generating the interarrival time and service time separately using the inverse transformation for each. But speed is not the primary issue, since the usual method can be made almost as fast by using a simple trick: generate A and S by partitioning an Erlang-2 random variable into two independent exponential random variables:

Set $Z \leftarrow -\ln(U_1 U_2)$,
Set $A \leftarrow U_3 Z$,
Set $S \leftarrow Z - A$,
Set $W \leftarrow \max(0,\ W + (S/\mu) - (A/\lambda))$,

where U_1, U_2, and U_3 are independent uniform (0,1) random numbers.

5. M/M/1 SOJOURN TIMES

Another time series with steady-state exponential marginal distribution is composed of adjacent M/M/1 sojourn times, T_i, the time spent by the i^{th} customer waiting in the queue plus the time in service. The steady-state mean is $(\mu - \lambda)^{-1}$, where λ and μ are the arrival rate and service rate, respectively.

The conditional sojourn-time distribution follows from the FIFO recursive relationship $T_i = S_i + \max(0, T_{i-1} - A_i)$, where S_i denotes service time and A_i denotes interarrival time for the i^{th} customer. This recursion, which is valid for any service and interarrival distributions, leads to

$$P\{T_i > t_i \mid T_{i-1} = t_{i-1}\} = \int_0^{\max(0,t_{i-1}-t_i)} dF_{A_i}(a_i)$$

$$+ \int_{\max(0,t_{i-1}-t_i)}^{t_{i-1}} P\{S_i > a_i - (t_{i-1} - t_i)\}\, dF_{A_i}(a_i)$$

$$+ \int_{t_{i-1}}^{\infty} P\{S_i > t_i\}\, dF_{A_i}(a_i)\ ,$$

where the three terms correspond to waiting in the queue longer than t_i, arriving to a busy server but spending less than t_i in the queue, and arriving to an idle server. Using exponential distribution functions leads to the M/M/1 conditional sojourn-time distribution

$$F_{T_i|T_{i-1}=t_{i-1}}(t_i) = e^{-\lambda \max(0,\, t_{i-1}-t_i)}$$

$$- \frac{\lambda}{\lambda+\mu} e^{-\mu(t_i - t_{i-1})} [e^{-(\lambda+\mu)\max(0, t_{i-1}-t_i)} - e^{-(\lambda+\mu)t_{i-1}}]$$

$$- e^{-\mu t_i - \lambda t_{i-1}}$$

$$= \begin{cases} \frac{\mu}{\lambda+\mu} e^{-\lambda t_{i-1}} (e^{\lambda t_i} - e^{-\mu t_i}) & \text{if } 0 \le t_i \le t_{i-1} \\ 1 - c e^{-\mu t_i} & \text{if } t_{i-1} < t_i \,. \end{cases}$$

where $c = \frac{\mu}{\lambda+\mu} e^{-\lambda t_{i-1}} + \frac{\lambda}{\lambda+\mu} e^{\mu t_{i-1}}$.

Setting $t_i = t_{i-1}$, we find

$$P\{T_i \le t_{i-1}\} = \frac{\mu}{\lambda+\mu}(1 - e^{-(\lambda+\mu)t_{i-1}}) \,.$$

If the random number u is less than this quantity, we invert the left part of $F_{T_i|T_{i-1}=t_{i-1}}$ and otherwise we invert the right part. The closed-form inversion of the right part is

$$t_i = F^{-1}_{T_i|T_{i-1}=t_{i-1}}(u_i) = -\ln((1-u_i)/c) \,/\, \mu \,.$$

Unfortunately, the left part has no closed-form inverse. In the algorithm below, we use binary search for t_i in the interval $(0, t_{i-1})$. The search is improved with the closed-form lower bound obtained by ignoring the term $e^{-\mu t_i}$ in $F_{T_i|T_{i-1}=t_{i-1}}$. With a better search method or with less-stringent convergence criteria, the speed of the algorithm would improve, but with any reasonable-speed random-number generator the inverse transformation is slower than generating from the FIFO recursion, especially if the Erlang-2 trick is used.

The design of the algorithm is the same as that for the M/M/1-QT process. Again, a steady-state variate is returned if the previous time is negative, allowing either transient or steady-state simulation.

```
      subroutine pmm1st (arate, srate, eps, iseed, t)
c.....bruce schmeiser and tina song
c       july 1989
c       purdue university
c.....purpose:
c       generate one m/m/1 sojourn time (in the system)
c         conditional on the last sojourn time.
c.....method:
c       if the previous time is negative, this time is
c       generated from the steady-state distribution.
c       otherwise, this time is generated from the
c       conditional distribution given the previous time,
c       using the inverse transformation, which requires
c       a search when this time is less than the previous
c       time.
c.....input
c       arate: arrival rate
c       srate: service rate
c       eps:   allowable error in t (in the binary search)
c       iseed: random-number seed
c       t:     previous observed sojourn time
c.....output
c       iseed: random-number seed
c       t:     observed sojourn time
c.....other routine used
c       rand:  a u(0,1) random-number generator
      u = rand(iseed)
      tl = t
      if (tl .lt. 0.) then
c       ...generate a steady-state sojourn time...
        t = - alog(1.-u) / (srate-arate)
      else
c       ...generate  sojourn time conditional on tl...
        rate2 = arate + srate
        sratio= srate / rate2
        if (u .gt. sratio*(1.-exp(-rate2*tl))) then
c           ...t is greater than tl...
           c = sratio*exp(-arate*tl)
     1         + (arate/rate2)*exp(srate*tl)
           t = -alog((1.-u)/c) / srate
        else
c           ...t is less than tl.  binary search...
           c  =  sratio * exp(-arate*tl)
           if (u .le. c)  then
             bottom = 0.
           else
             bottom = alog(u/c) / arate
           endif
           top = tl
           i = 0
 10          i = i + 1
             t = (bottom + top) * .5
             if (exp(arate*t) - exp(-srate*t) .lt. u/c) then
               bottom = t
             else
               top    = t
             endif
           if(i.lt.20 .and. abs(bottom-top).gt.eps) go to 10
        endif
      endif
      return
      end
```

6. MARKOV CHAINS

Realizations from a discrete-time Markov chain are obtained by sampling from the conditional distribution associated with the current state. An observation is the value associated with each state, and the states are ordered in increasing value. The inverse transformation then requires only that we use the inverse transformation of the conditional distribution associated with the current state. When the state space is discrete and large, the inverse transformation is slow if implemented crudely. Index tables can speed execution time; see Fishman and Moore (1984).

Subroutine *p2smc* generates one observation from a two-state Markov chain having arbitrary states with equal limiting probabilities and a specified lag-1 autocorrelation. If the previous value x is not (floating-point) equal to one of the two states, then the next x is generated from the steady-state distribution.

```
      subroutine p2smc (state1, state2, rho1, iseed, x)
c     bruce schmeiser and tina song
c      july 1989
c      purdue university
c     purpose: generate one observation from the
c      symmetric two-state Markov chain process
c      having one-step transition matrix
c
c                      state1 state2
c           P = state1  | 1-p    p  |
c               state2  | p    1-p  | ,
c
c      where p = (1-rho1) /2. the lag-1 autocorrelation
c      is rho1. the limiting probabilities are
c      P{X=state1} = P{X=state2} = .5.
c     method: inverse transformation
c     input:
c       state1: value of the first state
c       state2: value of the second state
c       rho1:  the lag-1 autocorrelation
c       iseed:  random-number seed
c       x:     if c = state1 or c = state2, the last state
c              otherwise, generate from steady-state
c     output:
c       iseed:  random-number seed
c       x:     generated state
c     other routine used:
c       rand:  uniform (0,1) random-number generator
c
      u = rand (iseed)
      p = (1. - rho1) / 2.
      c = min (state1, state2)
      d = max (state1, state2)
      if ((x .ne. c) .and. (x .ne. d)) then
c
c     ...generate from the steady-state distribution
c
         if (u .le. 0.5) then
            x = c
         else
            x = d
         endif
      else
c
c     ...generate from the conditional distribution
c
         if (x .eq. c) then
            if (u .le. 1.-p) then
               x = c
            else
               x = d
            endif
         else
            if (u .le. p) then
               x = c
            else
               x = d
            endif
         endif
      endif
      return
      end
```

7. DISCUSSION

We have discussed inverse transformations for some simple stochastic processes commonly used in Monte Carlo studies. Only the two M/M/1 processes required analysis. While the concept of the inverse transformation applies to most stochastic processes, many appear intractable. Examples include the gamma time-series processes of Lewis (1982) and Schmeiser and Lal (1982), as well as most queuing systems. Nevertheless, additional tractable processes are probably available, perhaps simple inventory models.

ACKNOWLEDGMENTS

This research was supported by the National Science Foundation under Grant No. DMS-871779 to Purdue University. We acknowledge helpful discussions with David Kelton, Barry Nelson, and Michael Taaffe.

REFERENCES

Barone, P. (1987). A method for generating independent realizations of a multivariate normal stationary and invertible $ARMA(p,q)$ process. *Journal of Time Series Analysis* **8**, 125-130.

Devroye, L. (1986). *Non-Uniform Random Variate Generation*. Springer-Verlag.

Fishman, G.S. and Moore, L.R (1984). Sampling from a discrete distribution while preserving monotonicity. *The American Statistician* **38**, 219-223.

Kachitvichyanukul, V. Cheng, S.J. and Schmeiser, B.W. (1988) Fast Poisson and binomial algorithms for correlation induction. *Journal of Statistical Computation and Simulation* **29**, 17-33.

Lewis, P.A.W. (1980). Simple models for positive-valued and discrete-valued time series with ARMA correlation structure. In: *Multivariate Analysis — V* (P.R. Krishnaiah, ed.). North Holland, 151-166.

Lewis, P.A.W. (1982). Simple multivariate time series for simulations of complex systems. In: *Proceedings of the Winter Simulation Conference* (T.I. Ören, C.M. Delfosse and C.M. Shub, eds.), 389-390.

Ramberg, J.S. and Schmeiser, B.W. (1972). An approximate method for generating symmetric random variables. *Communications of the ACM* **15**, 987-990.

Schmeiser, B. (1980). Random variate generation: a survey. In: *Simulation with Discrete Models: A State of the Art View (Proceedings of the Winter Simulation Conference, volume 2) (T.I. Oren, C.M. Shub and P.F. Roth, eds.), 79-104.*

Schmeiser, B. and Kachitvichyanukul, V. (1986). Correlation induction without the inverse transformation. In: *Proceedings of the Winter Simulation Conference* (J.R. Wilson, J.O. Henriksen and S.D. Roberts, eds.). 266-274.

Schmeiser, B.W. and Kachitvichyanukul, V. (1989). Non-inverse correlation induction: guidelines for algorithm development. In: *Random Numbers and Simulation*, (J. Lehn and H. Neunzert, eds.). West Germany. Forthcoming.

Schmeiser, B.W. and Lal, R. (1982). Bivariate gamma random vectors. *Operations Research* **30**, 355-374.

Song, W.-M. T. and Schmeiser, B. (1989). Estimating standard errors: Empirical behavior of asymptotic mse-optimal batch sizes. In: *Computing Science and Statistics: Proceedings of the* 20th *Symposium on the Interface* (E.J. Wegman, D.T. Gantz and J.J. Miller, eds.), 575-580.

AUTHOR'S BIOGRAPHIES

BRUCE SCHMEISER is a professor in the School of Industrial Engineering at Purdue University. He received his undergraduate degree in mathematical sciences and master's degree in industrial and management engineering at The University of Iowa. His Ph.D. is from the School of Industrial and Systems Engineering at the Georgia Institute of Technology. He is the Simulation area editor of *Operations Research* and has served in editorial positions of *IIE Transactions, Communications in Statistics, B: Simulation and Computation, Journal of Quality Technology, American Journal of Mathematical and Management Sciences,* and the *Handbook of Industrial Engineering*. He is a past chairman of the TIMS College on Simulation and Gaming. He represents ORSA on the Winter Simulation Conference Board of Directors, currently serving as chairman. His research interests are the probabilistic and statistical aspects of digital-computer stochastic simulation, including input modeling, random-variate generation, output analysis, and variance reduction.

Bruce Schmeiser
School of Industrial Engineering
Purdue University
West Lafayette, IN 47907, U.S.A.
(317) 494-5422
schmeise@ecn.purdue.edu (arpanet)

WHEY-MING TINA SONG is a visiting assistant professor in the School of Industrial Engineering at Purdue University, where she received her Ph.D. in 1988. Her undergraduate degree in statistics and master's degree in industrial management are from Cheng-Kung University in Taiwan. She received master's degrees in applied mathematics and industrial engineering from the University of Pittsburgh.

Whey-Ming Tina Song
School of Industrial Engineering
Purdue University
West Lafayette, IN 47907, U.S.A.
(317) 494-5414
wheyming@ecn.purdue.edu (arpanet)