Effective Random Number Generation for Simulation Analyses Based On Neural Networks

V. Zorkadis¹ and D.A. Karras²

¹ University of Ioannina, Dept.of Computer Science, Greece, e-mail: zorkadis@cs.uoi.gr
² University of Piraeus, Dept. of Business Administration, Rodu 2, Ano Iliupolis, Athens 16342, Greece, e-mail:

ABSTRACT: This paper presents a novel methodology for generating (pseudo) random number sequences of good quality. It is intended that the proposed random number generators be used in simulation analyses. The suggested approach is based on the exploitation of the learning capabilities of Multi Layer Perceptron (MLP) neural networks and more specifically on their ability to estimate their weights through minimizing a suitable objective function. The quality of the resulting random numbers is evaluated by applying them to the simulation of a single-queuing system aiming at defining its parameters which are then, compared to the analytically calculated ones. It is found that, in terms of this task, the proposed MLP random number generator behaves very favorably compared to other traditional ones.

Key-words: Neural Networks, Simulation Analysis, Performance Analysis.

1. INTRODUCTION

Simulation analyses of stochastic systems incorporate random components, which require methods to obtaining numbers that are random in some sense. These random number sequences can be obtained through use of random number generators, which are involved in several other applications too. For instance, authentication mechanisms may use nonces, i.e., random numbers to protect against replay attacks like the corresponding mechanisms in ITU X.509 [1]. Symmetric and asymmetric cryptographic systems like DES, IDEA, RSA [2] that are employed for confidentiality purposes and as basic element of other security protocols require random cryptographic keys, should the cryptoanalysis be hard. Furthermore, integrity mechanisms such as ISO 8731-2 [3] or cryptographic key exchange mechanisms such as the Diffie Hellman Protocol [4] or the construction of digital signatures like the ElGamal or Digital Signature scheme (DSS) need the generation and use of random numbers. In addition, random numbers are used for the generation of traffic and message padding, in order to protect against traffic analysis attacks and for the computation of strong and efficient stream ciphers [5].

The main properties these random bit sequences should satisfy include, uniformity in the interval [0,1], pairwise uncorrelation, low storage requirements, possibility to produce many times the same sequence of random numbers and the facility to produce various such streams. In addition to the above properties, fast calculation is very important for simulation analyses. The great majority of random number generators used for traditional applications are linear congruential generators, which behave statistically very well, except in terms of unpredictability, since there exists a linear functional relation connecting the numbers of the sequence. A sequence of random numbers produced by these generators is defined as follows:

\[ Z_i = (aZ_{i-1} + c) \mod m \]

where \( m \), \( a \) and \( c \) are the coefficients, i.e., the modulus, the multiplier and the increment, correspondingly. \( Z_0 \) is the seed or initialization value. All are nonnegative integers. Each random number can be expressed, as mentioned above, as a function of another random number or of its predecessors or of the seed and the coefficients:

\[ Z_i = \left[ \frac{a^{-1} + ca^{-1}}{a - 1} \right] \mod m \]

So, if the coefficients and the seed or any random number belonging in the sequence is known, then all the numbers of the sequence can be inferred. True random numbers are independent from each other and therefore unpredictable but they are rarely employed, since it is difficult to obtain and they might be not reproducible. It is more common that numbers that behave like random numbers are obtained by means of an algorithm, i.e. a pseudorandom number generator. Next, we briefly describe some of the widely used generators, the DES in the output feedback mode (OFB) combined with a further element and linear congruential generators.

Data Encryption Standard (DES) and, recently IDEA, are the most widely used symmetric encryption systems. The input to the encryption function is the plain text in blocks and the key. The plain text block is 64 bits and the key 56 bits in DES and 128 bits in IDEA in length. The encryption and decryption algorithm of DES relies on permutations, substitutions and xor-operations under the control of 16 subkeys obtained from the initial key. On the other hand, the
encryption and decryption algorithms of IDEA rely on xor-operations and modular additions and multiplications. DES and IDEA can operate under various modes such as Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB). The OFB mode can be used as a pseudorandom number generator for key generation and stream cipher computation.

Based on the OFB of symmetric cryptosystems, like DES, cryptographically strong pseudorandom number generators are some of the most commonly employed, mainly in security mechanisms but, also, in other traditional applications, like simulation analyses. As a pseudorandom number generator the output feedback operation mode of symmetric cryptosystems is employed. According to this method the encryption function of the symmetric cryptosystem is, at first, applied to an initialization variable under the control of a cryptographic key. The resulting cipher is the new input to the encryption function. Subsequently, the output of the encryption function, i.e., the cipher is the pseudorandom bit string or number. The OFB can be used as a pseudorandom number generator for key generation and stream cipher computation.

The OFB mode can be used as a pseudorandom number generator (PMMLCP). It computes numbers in the interval $[0,1)$ using the following formula:

$$Z_i = 630360016Z_{i-1} \bmod (2^{31} - 1).$$

The multiplier $630360016$ is suggested by Payne, Rabung and Bogyo [6].

This paper presents a novel approach for constructing robust random number generators to be used in simulation studies, which are based on feedforward Artificial Neural Network (ANN) techniques. Since ANNs, in general, are parallel and distributed processing devices they can be implemented in parallel hardware and consequently, they can be used for real-time applications such as random number generation for simulation, which require fast calculations as discussed above.

It is well known that ANNs possess very interesting function approximation capabilities making them a very powerful tool in many scientific disciplines. For instance, feedforward ANNs of the MultiLayer Perceptron (MLP) type have the theoretical ability to approximate arbitrary nonlinear mappings as well as their differentials [7]. These capabilities are based on their learning properties, which are related to the optimization process of a cost function. These ANN of the MLP type are exactly the ones employed in this paper as random number generators for simulation analyses. More specifically, it is proposed that if, instead of training such a model to approximate an unknown function involving a finite number of its samples, as usually met in Neural Network applications, we train it to decorrelate its hidden nodes outputs then, we could produce at each learning epoch a sequence of strong (pseudo)random numbers satisfying independence and uniformity property. Thus, in the suggested approach, the MLP error function is replaced by a function measuring the decorrelation of its hidden nodes outputs. After a very few only training epochs, these MLP hidden nodes outputs form a sequence of real numbers satisfying the properties required by random bit sequences for simulation analyses.

It is experimentally illustrated that the suggested generators are very effective when applied to simulation problems, exhibiting a successful performance favorably compared to that of well known in the literature random number generators, like the inversive and the linear congruential ones. The simulation of a classical queuing system has been selected as such in the present study.

The organization of this paper is as follows. Section 2 describes the suggested novel procedure for generating strong (pseudo)random numbers by invoking MLP training techniques and illustrates the task of a single queuing system simulation, which has been selected for evaluating the suitability of the proposed random bit sequence generator in simulation analyses applications. Section 3 gives an account of the experimental study conducted. Finally, section 4 concludes the paper and discusses the prospects of our approach.

### 2. THE MLP BASED (PSEUDO) RANDOM NUMBER GENERATOR

The MLP learning property of the weight adaptation to be based on the optimization of a flexible, free to select, objective function, is exploited in the design of improved Random Number Generators (RNG) for simulation analyses.

Concerning this characteristic we could exploit it by considering as the target of the MLP training process not the minimization of the usually used sum of squared output errors cost function but instead, as the maximization of a measure of the decorrelation of the functioning of its hidden units. Such a process could lead to the estimation of hidden neurons activations which are not correlated to one another and thus, satisfy the decorrelation property of true random numbers as discussed above.

The above mentioned principle determines exactly the RNG process proposed here.

1. We consider a large 3-layer MLP with one hidden layer containing as many hidden neurons as random numbers to be generated.
2. The cost function that is minimized at each epoch is the covariance of every two hidden neurons activations, over all training patterns, since this can be a measure of the decorrelation of these neurons activations, as follows:

$$\sum (O_{ip} - <O_i>) \times (O_{jp} - <O_j>)$$

where, the symbol $<X>$ denotes mean value of a random variable and the sum is considered, at
each epoch, over all training patterns P and hidden neurons I, J.

3. After a few only training epochs, which ensures fast calculation of all the random numbers, this “training process” ends with the estimation of the MLP weights so that its hidden neuron outputs be uncorrelated to one another.

4. Due to well known properties of the sigmoidal nonlinearity \[g\], there is high possibility that an MLP could be brought either into a situation where its output is near 0.5 or into saturation, depending on the magnitude of the input I of the sigmoidal function \(g\). Thus, for these two cases we have,

\[ I = 0 \Rightarrow g(I) = 1/2 \]
\[ I = \pm \infty \Rightarrow g(I) = 1(0). \]

Therefore, in practice the MLP hidden units outputs cannot be used alone as a mechanism for producing random numbers uniformly distributed in the range \([0,1]\). To this end, we have considered the Unix-function \(mod\), which outcomes the non-integer part of a real number, as the required mechanism for aiding an MLP hidden unit output to acquire the desired properties, since the first digits of its decimal part are predictable. Consequently, the formula for the kth random number produced as the modified output of the kth hidden unit of the MLP proposed random number generator is as follows.

\[ O_k = \text{mod} \left(1000 \times g\left(\sum W_i I_i\right)\right) \]

Following the above illustrated steps a sequence of (pseudo)random numbers is produced whose quality is quantitatively evaluated by utilizing the simulation test presented in the next paragraphs.

We examined the quality of the pseudorandom number generators demanded for simulation analyses, by means of a simulation test based on queuing theory \([8,9]\). The model, we have selected to test the generators is the M/M/1 queue, the simplest nontrivial system, one property of which is that both the interarrival and the service times fulfil the Markov memoryless property, i.e., they are exponential in the continuous case. This queuing system has an exact solution, so that it is possible to derive the average system time \(T\), the average number of system customers \(N\) and the average waiting time \(W\) and thus, the average number of waiting in queue customers \(NQ\). If \(\bar{\epsilon}\) is the arrival rate and \(I\) the service rate then, \(T = 1/(1-\bar{\epsilon})\), \(N = \bar{\epsilon}T\), \(W = T-x = (x=1/1)\), \(NQ = \bar{\epsilon}W\). In order to evaluate the (pseudo)random number generators following this approach, we analyze this M/M/1 queuing system by simulation and compare the obtained results with those calculated by means of the above formulae. To this end, during simulation analysis of such a model, we have to generate random variates from an exponential distribution. For each exponential variate \(y\) we use a variate \(u\) that is uniformly distributed in \([0,1]\), i.e. a (pseudo)random number. From this random number \(u\) and the mean of the interarrival times \(\bar{\epsilon}\), we obtain the exponential variate as \(y = -\bar{\epsilon} \log u\). Obviously, the random numbers \(u\) are produced by the RNG under consideration and comparison.

3. EXPERIMENTAL STUDY

An experimental study has been carried out in order to demonstrate the efficiency of the suggested in section 2 procedure for designing pseudorandom number generators, concerning their performance with respect to the simulation test mentioned in the same section. More specifically, the following experiments have been conducted by applying this simulation test on

1. A random sequence produced by the DES algorithm.
2. A random sequence produced by the prime modulus multiplicative linear congruential pseudorandom (PMMLCP) number generator mentioned in the introduction and frequently used in simulation studies (thus, denoted by \texttt{randsim} in the relevant curves of the following graphs).
3. A random sequence produced by the one way-function approach (hash functions), namely, by the Message Digest (MD5) technique \([3,5]\).
4. A random sequence produced by the MLP based proposed pseudorandom number generator described in section 2. The MLPs herein employed have the 3-5000-1 architecture.

All the sequences herein produced and compared have 5000 points. All the results obtained from the above specified experiments concerning the simulation test are presented in the graphs of figure 1. These graphs demonstrate that the proposed MLP generator outperforms, or in the worst case (regarding the second graph) is comparable to, the traditional ones, especially in the case of a large number of customers.

4. CONCLUSIONS AND PROSPECTS

It has been studied for the first time how MLP can be used in creating effective (pseudo) random bit sequences to be used in simulation analyses. More specifically, one mechanism for turning MLP into such a generator has been identified. This mechanism relies on their learning capabilities to optimize a cost function meaning the decorrelation of MLP hidden nodes outputs. The results obtained are quite promising but the presented technique has high storage requirements. The issue of pursuing other such techniques, but with low storage requirements, for improving traditional generators for simulation analyses is under investigation.
REFERENCES