Proceedings of the 1992 Winter Simulation Conference
ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson

BUILDING CORRECT SIMULATION MODELS IS DIFFICULT

Enver Yücesan
INSEAD
European Institute of Business Administration
Boulevard de Constance
77305 Fontainebleau Cedex
France

Sheldon H. Jacobson
Department of Operations Research
Weatherhead School of Management
Case Western Reserve University
Cleveland, OH 44106-7235
U.S.A.

ABSTRACT

An application of the theory of computational complexity to the problem of verifying certain structural properties of discrete event simulation models is illustrated. Such modeling questions as accessibility of states, ordering of events, ambiguity of model specifications, and execution stalling are asserted to be NP-complete decision problems. These results imply that it is highly unlikely that a polynomial-time algorithm can be devised to verify such properties. The consequences of these assertions cover a wide range of modeling and analysis questions in simulation.

1 MOTIVATION

In the study of discrete event dynamic systems (DEDS), the utility of analytical methods such as Markov chains and queueing theory is limited to relatively simple systems. Computer simulation is the most promising experimental technique for the analysis of complex configurations. Simulation modeling, however, is largely an art; developing and implementing an error-free simulation model is a difficult task. In fact, Nance [1981] points out that *"simulation still carries the label of an expensive, uncertain problem solving technique that represents the court of last resort."* The apparent ad hoc and complex reputation of discrete event simulation is due to the lack of a comprehensive modeling framework (analogous to differential equations for continuous variable dynamic systems) that facilitates specification and implementation of discrete event models.

A first step in improving the effectiveness of discrete event simulation is to construct *"a model development and management environment in which tools or procedures can be used to support modeling and analysis"* [Overstreet and Nance, 1985]. Efforts to construct such an environment abound. Henriksen [1983], Balci [1986], Balci and Nance [1987, 1992], and Balmer and Paul [1990] discuss the requirements for simulation model development environments (SMDE). On the other hand, Evans et al. [1967], Zeigler [1976], Nance [1981], Torn [1981], Overstreet [1982], Schruben [1983], Overstreet and Nance [1985], Nance [1987], Schruben and Yücesan [1987], Glynn and Iglehart [1988], and Som and Sargent [1989] have introduced various formalisms for discrete event simulation modeling. With a narrower focus, Simscript [Markowitz et al., 1963], Simula [Dahl and Nygaard, 1967], and GPSS [Gordon, 1961] provide a particular language to support modeling and simulation. Many simulation languages such as SLAM II [Pritsker, 1986] and SIMAN [Pegden et al., 1990] as well as simulators such as XCELL+ [Conway et al., 1987] and WITNESS [Gilman and Billingham, 1989] have adopted this pragmatic approach.

An important feature of these modeling and analysis platforms is the support tools they provide to detect potential problems with model specifications and to assist in the construction of model implementations. For example, simulation languages provide utilities to detect structural errors in simulation programs, which operate in an analogous fashion to compilers for high-level programming languages. Simulators eliminate the need for programming altogether by providing a set of basic building blocks and allowing the construction of a model implementation by simply parameterizing these blocks. Various formalisms offer a list of guidelines in an attempt to provide diagnostic tools to verify certain structural properties of models.

This paper asserts that the problem of verifying certain structural properties of discrete event simulation models is an intractable decision problem. More specifically, four questions associated with simulation modeling and structural analysis (accessibility of states, ordering of events, ambiguity

of model implementations, and execution stalling) are NP-complete, hence, intractable problems under the current theory of computational complexity [Garey and Johnson, 1979]. An important implication of these results is that it is unlikely that polynomial-time algorithms can be constructed to verify certain structural properties of models; the diagnostic procedures provided by various formalisms are either heuristics that only apply to specific problem instances or they are merely rules of thumb found to work "when applied with care."

To our knowledge, this paper represents the first application of the theory of computational complexity and NP-completeness to assess the difficulty of various simulation modeling questions. The treatment here is rather theoretical with little contribution to the practice of model verification. This is, however, a first step in formalizing -hence, gaining insight into- the (widely known) complexity of the model verification problem. Current research aims at identifying "easy" instances of this problem and developing heuristic algorithms to solve them. That effort should be of greater interest to practitioners.

This paper is organized as follows: Section 2 defines the structural properties of interest. The intractability results are discussed in Section 3 along with their implications for simulation modeling and analysis. Section 4 presents some concluding comments.

2 DEFINITIONS

The development of model specifications can be facilitated through the use of software tools designed to identify potential problems in the model. This paper asserts that decision problems associated with determining certain structural properties of discrete event simulation models are intractable. The best one can hope to do is construct model-specific procedures. In this section, the concepts associated with the intractability results are carefully defined. Some of the definitions are based on the developments in [Overstreet, 1982] and [Schruben, 1991].

A *model specification* is a representation of the system under study, reflecting the objectives of the study and the assumptions of the analysis. A model specification could be in a form such as Generalized Semi-Markov Processes (GSMP) [Shedler, 1987; Glynn and Iglehart, 1988], Condition Specifications [Overstreet and Nance, 1985] or Simulation Graphs [Schruben and Yücesan, 1987].

A *model implementation* is a translation of the model specification into a computer executable form. This could be a Simula [Dahl and Nygaard, 1967] or a high-level language (such as C or Pascal) program. To summarize, a model specification defines *what* a model does while a model implementation defines *how* the behavior is to be achieved [Overstreet and Nance, 1985].

The *state* of a system is a complete description of the system. A description of the state of a discrete event system includes values for all of its numerical attributes as well as any schedule it may have for the future. A system may be represented through a countably infinite number of states. Changes in the states, which occur at discrete points in time, are called *events*.

Within a GSMP representation, the system undergoes a state transition when an event associated with the occupied state occurs. Each of the several possible events associated with a state may trigger the next transition. Each event has its own distribution for determining the next state. At each transition, new events may be scheduled and clocks indicating the time until these events are scheduled to occur are set through an independent mechanism. If a scheduled event does not trigger a transition, but is associated with the next state, its clock continues to run. If such an event is not associated with the next state, it is canceled and its clock is turned off. More formally, let $\mathbf{S}$ be a finite or countable set of states and $\mathbf{E}$ be a finite set of events. For $S \in \mathbf{S}$, $\mathbf{E}(S)$ denotes the set of all events that can occur when the process is in state S. In state S, the occurrence of an event $E \in \mathbf{E}(S)$ triggers a transition to state S'. The probability that the transition under event E is from state S to state S' is denoted by $p(S';S,E)$. The actual event $E \in \mathbf{E}(S)$ that triggers a transition in state S depends on the clocks associated with the events in $\mathbf{E}(S)$. Each clock records the remaining time until its associated event triggers a state transition.

With these building blocks, it is now possible to define the structural properties of interest. These properties are defined here informally to provide some intuition about the types of questions we are posing. In the next section, however, the verification questions are stated formally as decision problems and are asserted to be NP-complete.

A state, S, is said to be *accessible* if there exists a valid finite sequence of events whose execution leads into S. Two events, A and B, are said to be *order-independent* if the execution of event A followed immediately by event B leads into the same state as the execution of event B followed immediately by event A, provided that the executions of both AB and BA are valid. Two implementations of a model specification are called *ambiguous* if the same sequence of events executed on the two

implementations lead to two different states. A model implementation is said to *stall* if there exists a finite sequence of events whose execution leads into a state where the termination condition is not satisfied and the events list is empty.

The theory of computational complexity provides a well-defined framework to assess the tractability of decision problems. Decision problems in the class NP are those problems for which a potential solution can be verified in polynomial time in the size of the problem instance. The complete problems for this class (that is, NP-complete problems) are the hardest problems in NP such that, if one such problem could be solved in polynomial time, then all problems in NP could be solved in polynomial time. For a comprehensive description of the theory of computational complexity, see Garey and Johnson [1979].

In the next section, the four verification questions are asserted to be NP-complete decision problems, and the implications for modeling are discussed. To our knowledge, the formal treatment exhibited in this paper has not been previously attempted. Prior work of Overstreet [1982], and Overstreet and Nance [1985], within the formalism of *condition specifications*, shows that certain decision problems can be reduced to the Halting Problem [Hopcroft and Ullman, 1979], and, thus, classifies them as undecidable decision problems. Yücesan [1989] presents analogous results within the context of Simulation Graph Models. Similar work has been done on Petri Nets [Peterson, 1977]. Torn [1981] applied some of these results in the context of simulation modeling. Fox and Landi [1968], on the other hand, point out that, if the model can be depicted as a finite-state Markov chain or if the model contains a finite state imbedded Markov chain which represents the only states of interest, it is possible to algorithmically identify particular states. They provide a polynomial-time algorithm to identify ergodic subchains and transient states of a stochastic matrix. This paper, however, adopting restricted definitions of accessibility, ordering, ambiguity, and stalling, classifies these decision problems as *decidable but intractable*.

3 RESULTS AND IMPLICATIONS

It is necessary to distinguish between two different causes of intractability. The first one, which is the focus of this paper, is that the decision problem is so difficult that a prohibitively large amount of time is needed to find a solution. The second one is that the solution itself is so extensive that the length of the expression required to describe it is prohibitive. As Garey and Johnson state [1979, p.11]: *"intractability of this sort is by no means insignificant, and it is important to recognize it when it occurs. However, in most cases, its existence is apparent from the problem definition. In fact, this type of intractability can be regarded as a signal that the problem is not defined realistically, because we are asking for more information than we could ever hope to use."*

It is, therefore, assumed that the space required to represent a discrete event simulation model implementation is of size n. This is the number of tape cells on a Turing machine needed for the representation of an implementation of the model and has no (direct) relationship with the number of possible states in the model. A model can be represented by a program of size n, yet there can be a countably infinite number of states. On the other hand, the space needed to represent a particular state is assumed to be a polynomial function of n, or simply $O(p_1(n))$. Analogously, an event, which is assumed to be encoded in $O(p_3(n))$, can be executed in polynomial time, or simply $O(p_2(n))$.

The verification questions are now stated as decision problems. In these definitions, the notation "$E_0E_1E_2...E_k \Rightarrow S$" denotes that the execution of the sequence of events $E_0, E_1, E_2,...,E_k$ leads to state S, while "$E_0E_1E_2...E_i \Rightarrow S' \neq S$ for i<m" implies that the execution of these events leads to any valid state except state S. Also note that E_0, the initialization event, is assumed to establish the initial state of the model implementation at the start of execution.

ACCESSIBILITY
Instance: A discrete event simulation model implementation,
An initial event, E_0,
A state, S,
A non-negative, finite integer, M.
Question: Does there exist a sequence of events $E_1, E_2,..., E_m$, with $m \leq M$, such that the execution of the sequence yields
$E_0E_1E_2...E_m \Rightarrow S$,
while $E_0E_1E_2...E_i \Rightarrow S' \neq S$ for i=1,2,..., m-1?

THEOREM 1: ACCESSIBILITY is NP-complete.
Proof: See Jacobson and Yücesan [1992].

Theorem 1 asserts that the problem of determining whether a particular state is reachable is an intractable decision problem. Several corollaries follow.

Corollary 1: Verification, the process of determining whether a model implementation performs as intended, is an intractable decision problem.

An immediate implication of Theorem 1 is that it is not a trivial exercise to assess whether a discrete event simulation model implementation is "logically connected," that is, whether all other events are reachable from the initialization event. Such a procedure would be useful in verifying the correctness of the logic in a model implementation. Several procedures have been suggested for model verification. They range from manual verification of logic to checking against known solutions, from modular testing to stress testing [Bratley et al., 1987; p.9]. Whitner and Balci [1989] offer a taxonomy and a comprehensive classification for model verification techniques; they include *informal* approaches such as desk checking and walkthrough, *formal* approaches such as proof of correctness and lambda calculus, *static* approaches such as syntax analysis and consistency checking, *dynamic* approaches such as black-box testing and white-box testing as well as *symbolic* approaches such as symbolic execution and cause-effect graphing, and *constraint* approaches such as assertion checking and boundary analysis. Each technique has a varying level of formality, complexity, and cost for both human and computer resources. The corollary asserts that all these procedures are heuristics, as are the diagnostics generated by simulation languages or simulators (e.g., structural checking and flow analysis in XCELL+ [Conway et al., 1987]).

Corollary 2: The determination of a valid experimental frame is an intractable decison problem.

For some input parameter values, a model implementation might be "logically connected" while, for other values, it might not be. One could use such information to determine appropriate ranges for input parameters over which a particular simulation can be correctly applied. The corollary asserts that it is not possible to devise such a polynomial-time validation procedure. Hence, the determination of a valid *experimental frame* [Zeigler, 1976] for a simulation study is an intractable decision problem. Path analysis, cause-effect graphing, stress testing, black-box testing, and white-box testing [Whitner and Balci, 1989] are useful heuristic techniques to address this problem.

The detection of initialization bias is an important problem in steady-state simulations. For example, Schruben [1982] and Schruben et al. [1983] propose tests to determine whether a set of observations is contaminated with initialization bias. Welch [1983] proposes a simple technique to determine a truncation point in the output series. All of these procedures assume a priori that the system can reach steady state. Theorem 1, however, implies

Corollary 3: A priori determination of whether the process will achieve steady state after m events are executed is an intractable decision problem.

If the state of interest represents the occurrence of a rare event, then Theorem 1 implies

Corollary 4: A priori determination of whether a rare event will be observed during a particular execution of the model implementation is an intractable decision problem.

On the other hand, if the state represents the satisfaction of run termination conditions, then Theorem 1 implies

Corollary 5: A priori determination of whether the execution of the model implementation will terminate in finite time is an intractable decision problem. (See also Theorem 4 and its implications.)

Structural analysis, data flow analysis, execution tracing, path analysis, and cause-effect graphing [Whitner and Balci, 1989] are heuristic approaches that could prove useful in addressing these problems.

ORDERING

Instance: A discrete event simulation model implementation,

An initial event, E_0,

Two distinct states, S_1 and S_2,

A non-negative, finite integer, K.

Question: Does there exist a sequence of events $E_1, E_2, ..., E_k$, with $k \leq K$, such that the execution of the sequence yields

$E_0E_1...E_{k-2}E_{k-1}E_k \Rightarrow S_1$ and

$E_0E_1...E_{k-2}E_kE_{k-1} \Rightarrow S_2$

while

$E_0E_1...E_iE_{i+1} \Rightarrow S"$

$E_0E_1...E_{i+1}E_i \Rightarrow S"$ for i=1,2,...,k-2,

with $S" \neq S_1$ and $S" \neq S_2$?

THEOREM 2: ORDERING is NP-complete.

Proof: See Jacobson and Yücesan [1992].

Theorem 2 asserts that it is unlikely to construct a polynomial-time algorithm to determine the outcome of an interchange in the execution of a sequence of events. Such heuristic techniques as syntax analysis, data flow analysis, execution tracing, execution monitoring, symbolic execution, and path analysis

[Whitner and Balci, 1989] could be useful. An important implication is concerned with simultaneously scheduled events. Since it is highly unlikely to devise a polynomial-time algorithm which determines the outcome of arbitrary handling of simultaneous events, it may be desirable to assign *execution priorities* to ensure a logically correct model implementation. Schruben [1983] presents rules of thumb to detect potential problems with simultaneously scheduled events; Sargent [1988] and Som and Sargent [1989] develop mechanisms to assign event execution priorities. The theorem asserts that these are useful heuristic procedures.

A second implication of this result concerns determining when certain variance reduction techniques (VRT) will be successful. For instance, the effectiveness of common random numbers, antithetic variates, or control variates typically depends upon the synchronization of events between pairs or sets of runs. Such synchronizations ensure that correlations with the correct sign have been induced. Thus, *"automating variance reduction within simulation languages so that it is always valid and consistently useful is a difficult and important problem"* [Schmeiser, 1990]. This theorem implies that it is unlikely to algorithmically determine whether perfect or satisfactory synchronization is achieved.

Another implication of this result concerns the verification of the sufficient *commuting condition*, which validates the application of infinitesimal perturbation analysis on discrete event simulation models [Glasserman, 1991]. Very roughly, the commuting condition requires that the state reached from another state through the occurrence of two events be independent of their order. Glasserman [1991, p.53] proposes the use of an "event diagram" (similar to state transition diagrams for Markov chains) to verify this condition by "simply following the arrows." He further states that (p.54): *"drawing a complete diagram is rarely feasible, but it is often possible to identify patterns that make it sufficient to consider certain critical regions that reveal whether or not [the commuting condition] is satisfied. To some extent, this begs the question, since one must know where the 'critical' region lies. Our experience is that it becomes quite clear where to look after drawing just a few circles and arrows."* Theorem 2, however, implies

Corollary 6: The verification of the commuting condition that validates infinitesimal perturbation analysis is an intractable decision problem.

AMBIGUITY

Instance: Two implementations, DES1 and DES2, of a discrete event simulation model

specification,

An initial event, E_0,

Two distinct states, S_1 and S_2,

A non-negative, finite integer, M.

Question: Does there exist a sequence of events $E_1, E_2, ..., E_m$, with $m \leq M$, such that the executions of the sequence in the two implementations, DES1 and DES2, yield

DES1: $E_0E_1E_2...E_m \Rightarrow S_1$

DES2: $E_0E_1E_2...E_m \Rightarrow S_2$

while

DES1: $E_0E_1...E_i \Rightarrow S' \neq S_1$ and

DES2: $E_0E_1...E_i \Rightarrow S' \neq S_2$ for i=1,2,...,m-1?

THEOREM 3: AMBIGUITY is NP-complete.

Proof: See Jacobson and Yücesan [1992].

Theorem 3 asserts that it is unlikely to algorithmically verify whether simulation model implementations are ambiguous. Two corollaries immediately follow.

Corollary 7: Model verification, which refers to checking whether the model implementation executes as it is intended in the model specification, is an intractable decision problem.

All the standard tools for debugging any computer program also apply to debugging a simulation model implementation. Further methods such as the Turing test are described in [Law and Kelton, 1991; Chapter 5]. Such checks, however, are rarely exhaustive. In fact, the corollary asserts that it is unlikely that a polynomial-time algorithm can be constructed for model verification. This is consistent with the assertion of Whitner and Balci [1989] that *"current state-of-the-art model proving techniques are simply not capable of being applied to even the simplest general modeling problems. (..) However, the advantage of realizing the proof of correctness - complete programmed model verification- is so great that when the capability is realized, it will revolutionize the verification software."*

It may be desirable to know when two simulation model implementations are, in some measurable sense, close to one another. The motivation for seeking such an "equivalence" between model implementations is largely logistical: one implementation may execute faster or may have more modest data requirements. However,

Corollary 8: The problem of establishing equivalences between simulation models with respect to their behavior is an intractable decision problem.

Hence, the definitions for model equivalence presented in [Overstreet, 1982], [Schruben, 1983], and [Sargent, 1988] yield intractable decision problems since they are based on the behavior of state variables during a particular run. In addition, the validity of the conceptual algorithm for the development of an efficient model implementation (event reduction) presented in [Som and Sargent, 1989] cannot be verified with a polynomial-time algorithm.

STALLING
Instance: A discrete event simulation model implementation,
An initial event, E_0,
A state, S*,
A stopping condition, C,
A non-negative, finite integer, K.
Question: Does there exist a sequence of events $E_1, E_2, ..., E_k$, with $k \leq K$, such that the execution of the sequence yields:
$E_0E_1E_2...E_k \Rightarrow S^*$
while 1. C is not satisfied,
2. The events list (L) is empty?

THEOREM 4: STALLING is NP-complete.
Proof: See Jacobson and Yücesan [1992].

Theorem 4 has both practical and theoretical implications. The practical implication is concerned with model verification. The result shows that it is not possible to algorithmically assess whether or not a simulation run will not stall and terminate in finite time. Since the built-in diagnostic utilities cannot detect such a problem, most commercial simulation packages terminate a run (by default) if the events list becomes empty at any instant during execution regardless of the termination conditions. Structural analysis, top-down testing, bottom-up testing, execution profiling, and path analysis [Whitner and Balci, 1989] could detect gross errors.

From a theoretical point of view, Theorem 4 implies

Corollary 9: The determination of valid stopping conditions for simulations is an intractable decision problem.

For example, the algorithm presented in [Duersch and Schruben, 1986] implementing confidence interval estimation techniques that determines how long a simulation should be run to produce results that satisfy a *predetermined* relative precision criterion is a heuristic procedure, since there is no *a priori* guarantee that the simulation implementation will achieve the desired precision criterion. This implication also applies to the heuristic formulas developed by Whitt [1989] to estimate the simulation run lengths required to achieve desired statistical precision in planning queueing simulations. Structural analysis, white-box testing, path analysis, and cause-effect graphing [Whitner and Balci, 1989] could provide additional help.

4 CONCLUDING COMMENTS

Discrete event simulation is a popular problem-solving technique which has had mixed success. It is argued that a first step in improving the effectiveness of simulation is to *"recognize the need for a model development and management environment in which tools can be used to support modeling and analysis"* [Overstreet and Nance, 1985]. Such a platform should support the production of a model specification and its analysis in order to [Balci and Nance, 1987]:

1. detect potential problems with the model specification (*Model Analyzer* within the SMDE),
2. assist in the construction of a model implementation (*Model Translator* within the SMDE),
3. verify the executable version of the simulation model (*Model Verifier* within the SMDE),
4. construct useful model documentation (*Model Generator* within the SMDE).

The results presented in this paper assert that it is unlikely that polynomial-time algorithms can be devised to support the first two tasks listed above. That is, this paper argues that the problem of verifying certain structural properties of discrete event simulation models is intractable. In particular, accessibility of states, ordering of events, ambiguity of model implementations, and execution stalling are asserted to be NP-complete. These results are unifying in that all the structural questions, as different as they are, are equally as hard as each other under the theory of NP-completeness.

The major implication of these results is that the verification procedures, that must be provided in a model development and management environment, cannot be automated through appropriate polynomial-time algorithms, unless all the decision problems in NP are polynomially solvable. At best, one can rely on rules of thumb or problem-specific procedures such as the diagnostic tools provided by simulation languages or modeling guidelines offered by different formalisms. In other words, computable heuristics are the best that can be found.

These theoretical results hence strongly support the

development of heuristic procedures [Whitney and Balci, 1989] for such intractable problems. Moreover, identifying particular model instances (restrictions) where the four NP-complete decision problems are tractable (i.e., polynomially solvable) would offer insight into the type of models for which issues such as model validation and verification are tractable, certain variance reduction techniques can be guaranteed, or validating infinitesimal perturbation analysis can be done through a polynomial-time algorithm. Current research aims at identifying such instances and constructing heuristic algorithms to address these problems, which should provide practical assistance in verifying certain structural properties of simulation models.

ACKNOWLEDGEMENTS

The authors would like to thank Bennett Fox for his helpful comments on an earlier version of this paper and Osman Balci for introducing the authors to the rich literature on programmed model verification.

The first author gratefully acknowledges financial support from the INSEAD R&D Programme under Project 2121R. The second author gratefully acknowledges financial support from the Weatherhead School of Management through the Dean's Research Fellowship Fund and the Center for the Management of Science and Technology .

REFERENCES

Balci, O. (1986) Requirements for Model Development Environments. *Computers and Operations Research.* Vol.13.1 (Jan.-Feb.). pp.53-67.

Balci, O. and R.E. Nance (1987) Simulation Model Development Environments: A Research Prototype. *Journal of the Operational Research Society.* Vol.38.8. pp.753-763.

Balci, O. and R.E. Nance (1992) The Simulation Model Development Environment: An Overview. Proceedings of the 1992 Winter Simulation Conference (Swain, Goldsman, and Wilson, Eds.)

Balmer, D.W. and R.J. Paul (1990) Integrated Support Environments for Simulation Modeling. Proceedings of the 1990 Winter Simulation Conference (Balci, Sadowski, and Nance, Eds.). pp.243-249.

Bratley, P., B.L. Fox, and L.E. Schrage (1987) *A Guide to Simulation.* Springer-Verlag. New York, NY.

Conway, R., W. Maxwell, J. McClain, and S. Worona (1987) *User's Guide to XCELL+.* The Scientific Press. Redwood City, CA.

Dahl, O.J. and N.K. Nygaard (1967) *Simula: A Language for Programming and Description of Discrete Event Systems. Introduction and User's Manual.* 5th Edition. Norwegian Computing Center, Oslo.

Duersch, R.R. and L.W. Schruben (1986) An Interactive Run Length Control for Simulations on PCs. Proceedings of the 1986 Winter Simulation Conference (Wilson, Henriksen, and Roberts, Eds.). pp.866-870.

Evans, J.W., J.F. Wallace, and J.L. Sutherland (1967) *Simulation Using Digital Computers.* Prentice Hall. Englewood Cliffs, NJ.

Fox, B.L. and D.M. Landi (1968) An Algorithm for Identifying the Ergodic Subchains and Transient States of a Stochastic Matrix. *Comm ACM.* Vol.11.9. pp.619-621

Garey, M.R. and D.S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company. New York, NY.

Gilman, A.R. and C. Billingham (1989) A Tutorial on SEE WHY and WITNESS. Proceedings of the 1989 Winter Simulation Conference (MacNair, Musselman, and Heidelberger, Eds.). pp.192-200.

Glasserman, P. (1991) *Gradient Estimation via Perturbation Analysis.* Kluwer Academic Publishers Group. Dordrecht, The Netherlands.

Glynn, P.W. and D.L. Iglehart (1988) Simulation Methods for Queues: An Overview. *Queueing Systems.* Vol.3.3. pp.221-256.

Gordon, G. (1961) A General Purpose System Simulation Program. Proceedings of Eastern Joint Simulation Conference, pp.87-104.

Henriksen, J.O. (1983) The Integrated Simulation Environment (Simulation Software of the 1990s). *Operations Research.* Vol.31.6. pp.1053-1073.

Hopcroft, J.E. and J.D. Ullman (1979) *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley Publishing Company, Inc. Reading, MA.

Jacobson, S.H. and E. Yücesan (1992) On the Intractability of Verifying Structural Properties of Discrete Event Simulation Models. INSEAD Working Paper Series. TM/92. Fontainebleau, France.

Law A.M. and W.D. Kelton (1991) *Simulation Modeling and Analysis.* 2nd Edition.

McGraw-Hill. New York, NY.

Markowitz, H.M., H.W. Karr and B. Hausner (1963) *SIMSCRIPT: A Simulation Programming Language*. Prentice-Hall. Englewood Cliffs, NJ.

Nance, R.E. (1981) The Time and State Relationships in Simulation Modeling. *Com ACM*. Vol.24.4. pp.173-179.

Nance, R.E. (1987) The Conical Methodology: A Framework for Simulation Model Development. The Proceedings of the Conference on Methodology and Validation (SCS). pp.38-43.

Overstreet, C.M. (1982) Model Specification and Analysis for Discrete Event Simulations. Unpublished PhD Dissertation. Virginia Tech. Blacksburg, VA.

Overstreet, C.M. and R.E. Nance (1985) A Specification Language to Assist in Analysis of Discrete Event Simulation Models. *Com ACM*. Vol.28.2, pp.190-201.

Pegden, C.D., R.E. Shannon, and R.P. Sadowski (1990) *Introduction to SIMAN*. Systems Modeling Corporation. Pittsburgh, PA.

Peterson, J.L. (1977) Petri Nets. *Computing Surveys*. Vol.9.3. pp.223-252.

Pritsker, A.A.B. (1986) *Introduction to Simulation and SLAM II*. 3rd Edition. John Wiley & Sons. New York, NY.

Sargent, R.G. (1988) Event Graph Modeling for Simulation with an Application to Flexible Manufacturing Systems. *Management Science*. Vol.34.10. pp.1231-1251.

Schmeiser, B. (1990) Simulation Experiments in Handbook of Operations Research and Management Science within volume on *Stochastic Models* (D. Heyman and M. Sobel, Eds.) North Holland. New York, NY.

Schruben, L. (1982) Detecting Initialization Bias in Simulation Output. *Operations Research*. Vol.30.3. pp.569-590

Schruben, L. (1983) Simulation Modeling with Event Graphs. *Com ACM*. Vol.26.11. pp.957-963.

Schruben, L. (1991) *Sigma: A Graphical Simulation System*. The Scientific Press. San Francisco, CA.

Schruben, L., H. Singh, and L. Tierney (1983) Optimal Tests for Initialization Bias in Simulation Output. *Operations Research*. Vol.31.6. pp.1167-1178.

Schruben, L. and E. Yücesan (1987) On the Generality of Simulation Graphs. Technical Report #773. School of OR&IE, Cornell University. Ithaca, NY.

Shedler, G.S. (1987) *Regeneration and Networks of Queues*. Springer-Verlag. New York, NY.

Som, T.K. and R.G. Sargent (1989) A Formal Development of Event Graphs as an Aid to Structured and Efficient Simulation Programs. *ORSA Journal on Computing*. Vol.1.2. pp.107-125.

Torn, A.A. (1981) Simulation Graphs: A General Tool for Modeling Simulation Designs. *Simulation*. Vol.37. pp.187-194.

Welch, P.D. (1983) The Statistical Analysis of Simulation Results. *The Computer Performance Modeling Handbook* (S.S. Lavenberg, Ed.) pp.268-328. Academic Press. New York, NY.

Whitner, R.B. and O. Balci (1989) Guidelines for Selecting and Using Simulation Model Verification Techniques. Proceedings of the 1989 Winter Simulation Conference (MacNair, Musselman, and Heidelberger, Eds.). pp.559-568.

Whitt, W. (1989) Planning Queueing Simulations. *Management Science*. Vol.35.11. pp.1341-1366.

Yücesan, E. (1989) Simulation Graphs for the Design and Analysis of Discrete Event Simulation Models. Unpublished PhD Dissertation. Cornell University. Ithaca, NY.

Zeigler, B.P. (1976) *Theory of Modelling and Simulation*. John Wiley. New York, NY.

AUTHOR BIOGRAPHIES

ENVER YUCESAN is an Assistant Professor of Operations Management at the European Institute of Business Administration (INSEAD). He received a B.S. degree in Industrial Engineering from Purdue University, and M.S. and Ph.D. degrees in Operations Research from Cornell University. His research interests include issues related to construction and structural analysis of discrete event models, statistical analysis of simulation output as well as analysis of production-distribution systems.

SHELDON H. JACOBSON is an Assistant Professor in the Department of Operations Research in the Weatherhead School of Management at Case Western Reserve University. He received a B.Sc. and a M.Sc., both in Mathematics, from McGill University, and a M.S. and a Ph.D. in Operations Research from Cornell University. His research interests include simulation optimization and sensitivity analysis, frequency domain methods of analyzing simulation outputs, and computational complexity.