NUCLEAR SCIENCE AND ENGINEERING: **146**, 71–87 (2004)

Calculating Nuclear Power Plant Vulnerability Using Integrated Geometry and Event/Fault-Tree Models

Douglas E. Peplow,* C. David Sulfredge, Robert L. Sanders, and Robert H. Morris

Oak Ridge National Laboratory, Building 5700, MS 6172
P.O. Box 2008, Oak Ridge, Tennessee 37831

and

Todd A. Hann

Defense Threat Reduction Agency
Alexandria, Virginia

Received October 22, 2002
Accepted March 17, 2003

Abstract—*Since the events of September 11, 2001, the vulnerability of nuclear power plants to terrorist attacks has become a national concern. The results of vulnerability analysis are greatly influenced by the computational approaches used. Standard approximations used in fault-tree analysis are not applicable for attacks, where high component failure probabilities are expected; two methods that do work with high failure probabilities are presented. Different blast modeling approaches can also affect the end results. Modeling the structural details of facility buildings and the geometric layout of components within the buildings is required to yield meaningful results.*

I. INTRODUCTION

Reports in the popular news media have indicated that nuclear power plants are prime targets for terrorist organizations. A likely avenue for such an attack is a bomb carried by car or truck, similar to the recent events listed in Table I. Car bombs require less preparation, skill, or manpower than complex attacks such as those of September 11, 2001.

The managements of nuclear power plants, as well as other infrastructure targets, need to know the parts of their facilities where a bomb explosion could lead to facility shutdown—or in the worst case, core damage (potential of release of hazardous materials). These areas need to be identified so that they can be adequately protected.

To determine the areas where nuclear facilities are vulnerable, a calculational tool is needed that can quickly evaluate the effects of a bomb explosion in or around the buildings of a facility and determine the probable impact on facility operation as well as the probability of an accompanying radiological release. The Visual Interactive Site Analysis Code (VISAC) developed at Oak Ridge National Laboratory (ORNL) does this using a geometric model of the facility coupled to an event/fault-tree model of plant systems to analyze the effects of blasts. The event/fault-tree models associated with facility vulnerability calculations often involve unreliable systems (systems with high component failure probabilities resulting from an attack scenario). For VISAC to analyze such situations accurately, ORNL had to develop some novel techniques for evaluating event/fault trees associated with unreliable systems.

II. UNRELIABLE EVENT/FAULT-TREE CALCULATION

Event/fault-tree calculations have been used in the nuclear industry for a long time. Popular software tools

*E-mail: peplowde@ornl.gov

TABLE I

Recent Terrorist Attacks Against American Targets Using Car-Bomb Technologies

Date	Target/Location	Delivery/Material	TNT Equivalent (lb)	Reference
April 1983	U.S. Embassy Beirut, Lebanon	Van	2 000	www.beirut-memorial.org
October 1983	U.S. Marine Barracks Beirut, Lebanon	Truck, TNT with gas enhancement	12 000	www.usmc.mil
February 1993	World Trade Center New York	Van, urea nitrate, and hydrogen gas	2 000	www.interpol.int
April 1995	Murrah Federal Building Oklahoma City, Oklahoma	Truck, ammonium nitrate fuel oil	5000	U.S. Senate documents
June 1996	Khobar Towers Dhahran, Saudi Arabia	Tanker truck, plastic explosive	20 000	www.fbi.gov
August 1998	U.S. Embassy Nairobi, Kenya	Truck, TNT, possibly Semtex	1 000	News reports, U.S. Senate documents
August 1998	U.S. Embassy Dar es Salaam, Tanzania	Truck	1 000	U.S. Senate documents
October 2000	Destroyer USS Cole Aden Harbor, Yemen	Small watercraft, possibly C-4	440	www.al-bab.com news.bbc.co.uk

used for event/fault-tree analysis typically rely on cut set approaches. These programs take advantage of low failure probabilities to use several approximations that greatly speed up the calculations. The codes are designed for reliable systems, such as a commercial pressurized water reactor, which has an estimated core melt frequency of $\sim 1.0 \times 10^{-5}$ to 2.0×10^{-4}/yr under normal conditions.[1]

Unlike typical nuclear power plant safety analysis, the damage in a terrorist attack would be inflicted intentionally, and the failure probabilities for many of the basic components can be quite high or even unity. These high failure probabilities tend to violate the assumptions used in the formulation of typical cut set methods for fault-tree analysis, causing them to give erroneous and sometimes ridiculous answers. When large failure probabilities are introduced into an event/fault-tree system, it can become an unreliable system for which the current analysis techniques were not designed.

This section will first review the methods used for typical event/fault-tree analysis and the common approximations, show how these approximations do not perform well for high failure probabilities, and then describe two methods that do work well for high or low basic event failure probabilities.

II.A. Brief Description of Typical Fault-Tree Methods

For this discussion, consider the simple event/fault-tree system shown in Fig. 1. A set of N independent basic events e_n is used in a series of AND and OR gates to create I top-level fault-tree system gates (sys_1 through sys_I). Figure 2 shows an event tree consisting of three top-level fault trees: sys_1, sys_2, and sys_3.

Evaluation of each sequence in the event tree for independent component failure probabilities would be quite simple if no basic events were used more than once in the system of fault trees. Without these common events, each fault tree could be calculated independently of the

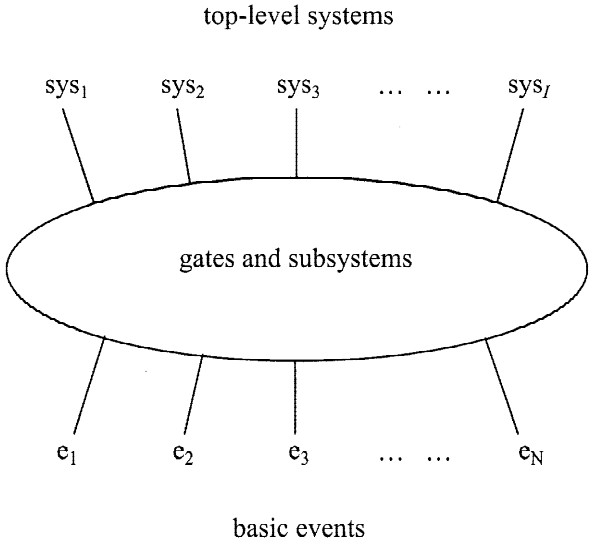

Fig. 1. The essence of an event/fault-tree problem. Basic events N feed into a set of gates and subsystems, resulting in I top-level gates that are used in an event tree.

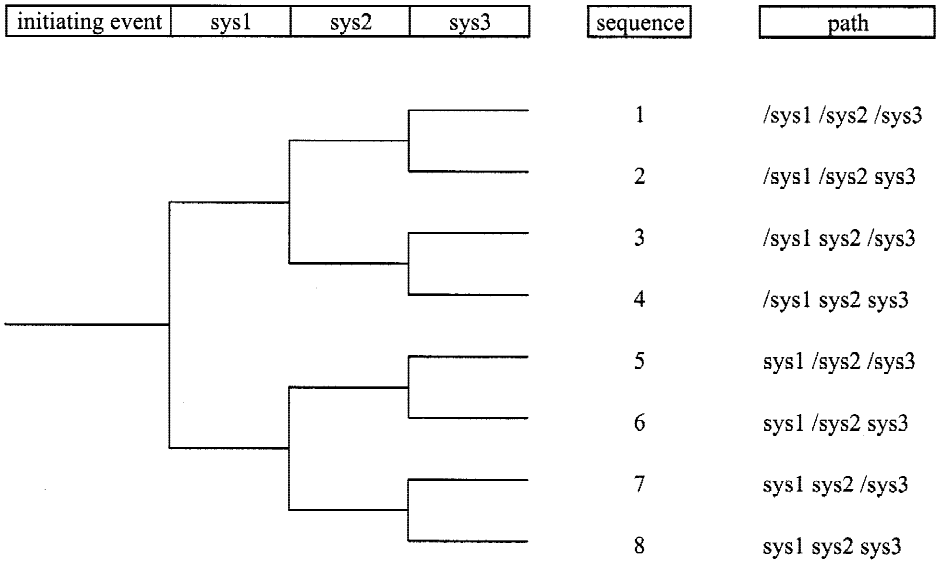

Fig. 2. A simple event tree made from three top-level gates.

others using a bottom-up approach. For independent events, the probability of an AND gate made of K basic events is found by simply multiplying the probability $P(e_i)$, of each basic event e_i together:

$$P(e_1 e_2 \ldots e_K) = P(e_1)P(e_2)\ldots P(e_K) \ . \qquad (1)$$

For mutually exclusive events, the probability of an AND gate is zero.

In general, the probabilities of an OR gate composed of K basic events can be evaluated using

$$P(e_1 + e_2 + \cdots + e_K) = \sum_{k=1}^{K} P(e_k) - \sum_{k<j=2}^{K} P(e_k e_j) + \cdots + (-1)^{(K-1)} P(e_1 e_2 \ldots e_K) \ . \qquad (2a)$$

If the events are independent and each is only used once, then Eq. (2a) reduces to

$$P(e_1 + e_2 + \cdots + e_K) = 1 - \prod_{k=1}^{K} (1 - P(e_k)) \ . \qquad (2b)$$

For mutually exclusive events, Eq. (2a) reduces to just its first term. When the events are not mutually exclusive, but all have small failure probabilities, the first term in Eq. (2a) provides a reasonable upper bound.

The event tree sequences are easily found by multiplying either system failure probabilities $P(sys_i) = F_i$ or system success probabilities $1 - F_i$ along each sequence. For example, the probability of a sequence that results from the success of sys_1, the failure of sys_2, and the success of sys_3 is found by

$$P(seq_3) = (1 - F_1)F_2(1 - F_3) \ . \qquad (3)$$

Unfortunately, real systems usually have common events. Fault trees cannot be evaluated independently if they share any basic components. At a branch in the event tree, the failure rate of a system F_i potentially depends on the success or failure of all event-tree branches evaluated prior to sys_i.

II.A.1. Brute Force Technique

A brute force technique can be used to evaluate the event-tree probabilities by calculating the outcome of every combination of basic events (either failed or not-failed) weighted by the probability of occurrence of that combination. For the N basic events, there are 2^N possible combinations of failed/not-failed events that need to be evaluated. This is a Bayesian approach because the 2^N combinations form a mutually exclusive set, one of which is certain to occur. The probability of each case occurring is the product of the failure or nonfailure probabilities for each of its basic events. The logic of the fault trees and the event tree is then evaluated to determine the event-tree sequence resulting from that combination. This method tends to be quick because the fault-tree logic is evaluated using integer arithmetic (only 1s and 0s) instead of multiplying and adding fractional probabilities. This is the most straightforward way that an event-tree problem can be solved, but it quickly becomes intractable as N gets large. For $N = 30$, there are

2^{30} (more than 10^9) combinations that need to be calculated. Even with modern computers, a brute force technique using every basic event is not feasible for real problems.

II.A.2. Monte Carlo Solution

Instead of evaluating every possible combination of the basic events, Monte Carlo methods can be used to sample the problem and estimate the probabilities of each sequence in the event tree. Binary states of failed/not-failed are assigned to each basic event in accordance with its failure probability, the fault trees are evaluated, and the path through the event tree is determined. Similar to the brute force method, evaluations of the fault trees and event tree are fairly quick because they can be done with integer arithmetic. The drawback of Monte Carlo for reliable systems is that very large numbers of histories are required to get the uncertainty of the answers down to an acceptable level. For highly reliable systems, Monte Carlo is a horribly inefficient method to solve event/fault-tree problems because hardly any of the histories lead to a failure sequence in the event tree. Though not exact, Monte Carlo does offer the ability to calculate an uncertainty with each answer, as opposed to methods that only give an upper or lower bound to the real answer.

II.A.3. Minimal Cut Set Analysis

The usual approach for general event-tree/fault-tree problems is to recast the problem into a set of combinations of basic events that will cause systems to fail or sequences in an event tree to be followed. For any given fault tree or for a certain sequence in an event tree, the minimal cut sets can be found by a variety of techniques.[2]

For example, suppose there are J combinations of basic events that will cause sequence 3 in the Fig. 2 event tree to be followed. These minimal cut sets are $C_1, C_2, ..., C_J$, where each C_j represents a combination of basic events. (Minimal means that the cut sets have been simplified by Boolean algebra so that they contain no redundant events and no cut set is a subset of another.) The probability of a cut set is found by computing the product of each basic event probability in the cut set. For example, if cut set 16 is the combination of events 3, 7, and 12 all failing, the probability of the cut set is simply found (assuming independent events) by

$$P(C_{16}) = P(e_3 e_7 e_{12}) = P(e_3)P(e_7)P(e_{12}) \ . \quad (4)$$

The probabilities of the cut sets can be combined exactly using a binomial expansion and eliminating redundancies from each term.

The probability of the third sequence is then found as the probability of the union of all of its cut sets:

$$P(seq_3) = P(C_1 + C_2 + \cdots + C_J) \quad (5a)$$

$$= \sum_i^J P(C_i) - \sum_{i<j=2}^J P(C_i C_j) + \sum_{i<j<k=3}^J P(C_i C_j C_k) - \cdots + (-1)^{(J-1)} P(C_1 C_2 \ldots C_J) \ . \quad (5b)$$

This yields $2^J - 1$ terms, which, similar to the aforementioned brute force technique, is not practical for large systems. Hence, some approximations must be invoked in order to use cut sets to solve real problems.

SAPHIRE (Ref. 3), a very popular probabilistic risk assessment tool, can be made to solve event/fault-tree problems using Eq. (5b). The user can specify how many passes to make, where each pass adds (or subtracts) the next summation term shown in Eq. (5b). The first pass will be an upper bound, the next a lower bound, the next an upper bound, and so on. For typical problems, the terms get small quite quickly, and the calculated probability of a sequence converges very quickly. For unreliable systems, the upper and lower bounds found by the first few passes can be very far from the true answer, requiring many more passes for convergence. Calculating all J passes will give the exact answer.

Another common approximation used in forming the collection of cut sets is not to include those systems that have not failed. For example, sequence 3 in Fig. 2 could be approximated by forming the cut sets for only the failure of sys_2 instead of the combination of $/sys_1\, sys_2 / sys_3$, which would have many more cut sets (note that the / is used for the *not* operation). For reliable systems, this approximation works well, which is the case for the majority of safety systems at nuclear installations.

II.A.4. Rare Events Approximations

In typical problems, the failure probabilities of the basic components are small enough that the overlap correction terms in Eq. (5b) are very small. The evaluation of the probability for a union of cut sets can then be approximated by just the first term of the summation:

$$P(C_1 + C_2 + \cdots + C_J) = P(C_1) + P(C_2) + \cdots + P(C_J) \ . \quad (6)$$

For highly reliable systems, this works fine because the multiplication of two or more small events is very small. Each fault tree is computed independently from the others and combined in the event tree. Any success probabilities in an event-tree sequence are often approximated simply as unity instead of using $1 - F_i$.

II.A.5. Minimal Cut-Set Upper Bound

The rare-events approximation tends to overestimate the failure probability of each sequence in the event tree. A better approximation is to use a sequence-based method of minimal cut sets. Here, each tree and each sequence in the event tree are reduced to their minimal cut sets, and the probability for each sequence is given by the probability for the union of all its cut sets C_j with $1 \le j \le J$. Hence,

$$P(\text{seq}_i) = P(C_1 + C_2 + \cdots + C_J) \ , \qquad (7)$$

which can be bounded for most event/fault-tree systems by assuming each cut set C_j is independent of the others. In this case, Eq. (5b) reduces to

$$P(\text{seq}_i) \le 1 - (1 - P(C_1)) \times (1 - P(C_2)) \cdots (1 - P(C_J)) \ . \qquad (8)$$

An even better upper bound can be formed if any of the basic events appear in every cut set by using

$$P(\text{seq}_i) \le P(C_c) \left[1 - \prod_{j=1}^{J} (1 - P(C_j)/P(C_c)) \right] \ , \qquad (9)$$

where C_c is the set of any events common to every cut set.[4] The minimal cut-set methodology forms the basis of most of the fault-tree analysis software.[5]

There are still further techniques that can be used, but most are still approximations.[6] One thing the reader should remember from this section is that only the brute force and the cut-set analysis with all terms included are exact, but they both require far too many calculations to implement for large systems.

II.A.6. Example Problem

To demonstrate the aforementioned methods and how they break down for unreliable systems, consider the following example problem distributed with the SAPHIRE code system. It consists of two top-level gates forming three sequences using 17 basic events. The event tree is shown in Fig. 3. The two top-level gates are expressed in terms of the basic events:

$$\begin{aligned} \text{ccs} = {} & \text{tank} + \text{cmov1} + \text{dgb} \\ & + (\text{ccva} + \text{cmova} + \text{cpumpa} + \text{dga}) \\ & \times (\text{ccvb} + \text{cmovb} + \text{cpumpb} + \text{dgb}) \end{aligned} \qquad (10)$$

and

$$\begin{aligned} \text{ecs} = {} & \text{tank} + \text{emov1} + \text{dga} \\ & + (\text{ecva} + \text{emova} + \text{epumpa} + \text{dga}) \\ & \times (\text{ecvb} + \text{emovb} + \text{epumpb} + \text{dgb}) \ . \end{aligned} \qquad (11)$$

Three cases are used with different degrees of failure probabilities, going from a very reliable system to an unreliable system. The failure probabilities for each case are shown in Table II. For each case, the probability of each sequence in the event tree is calculated using the following methods:

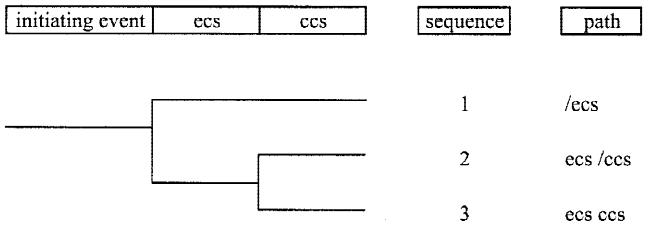

Fig. 3. The SAPHIRE example problem event tree using two top-level system gates.

1. brute force, the sum of 2^{17} (131 072) combinations of basic events failed or not failed
2. cut-set rare events approximation, with *not* terms included, Eq. (6)
3. minimal cut-set upper bound, with *not* terms included, Eq. (8)
4. cut-set exact method, Eq. (5b), for the first few passes.

For the first sequence, there are only two cut sets:

$$\begin{aligned} /\text{ecs} = {} & /\text{epumpa} /\text{emova} /\text{ecva} /\text{tank} /\text{dga} /\text{emov1} \\ & + /\text{tank} /\text{dga} /\text{ecvb} /\text{emov1} /\text{emovb} /\text{dgb} \\ & /\text{epumpb} \ . \end{aligned} \qquad (12)$$

TABLE II

The Failure Probabilities Used for Each Case of the SAPHIRE Example Problem

Basic Event	Case A	Case B	Case C
tank	0.0000001	0.000001	0.00001
dga	0.02	0.2	0.2
dgb	0.02	0.2	0.2
cmov1	0.001	0.01	0.1
emov1	0.001	0.01	0.1
ccva	0.0001	0.001	0.01
cmova	0.005	0.05	0.5
cpumpa	0.003	0.03	0.3
ccvb	0.0001	0.001	0.01
cmovb	0.005	0.05	0.5
cpumpb	0.003	0.03	0.3
ecva	0.0001	0.001	0.01
emova	0.005	0.05	0.5
epumpa	0.003	0.03	0.3
ecvb	0.0001	0.001	0.01
emovb	0.005	0.05	0.5
epumpb	0.003	0.03	0.3

For the second sequence there are 21 cut sets:

$$\begin{aligned}\text{ecs /ccs} = {} & \text{ecva emovb /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} + \text{ecva epumpb /cmov1 /tank /ccvb} \\ & \text{/cmovb /cpumpb /dgb} + \text{epumpa emovb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} \\ & + \text{epumpa epumpb /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} + \text{emova ecvb /cmov1 /tank} \\ & \text{/ccvb /cmovb /cpumpb /dgb} + \text{emova emovb /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} \\ & + \text{emova emovb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} + \text{ecva ecvb /cmov1 /tank} \\ & \text{/ccvb /cmovb /cpumpb /dgb} + \text{emova epumpb /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} \\ & + \text{ecva ecvb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} + \text{emov1 /cmov1 /tank /ccvb} \\ & \text{/cmovb /cpumpb /dgb} + \text{epumpa epumpb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} \\ & + \text{epumpa ecvb /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} + \text{ecva emovb /cmov1 /tank /dga} \\ & \text{/cmova /ccva /cpumpa /dgb} + \text{ecva epumpb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} \\ & + \text{emova ecvb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} + \text{emov1 /cmov1 /tank /dga} \\ & \text{/cmova /ccva /cpumpa /dgb} + \text{epumpa emovb /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} \\ & + \text{emova epumpb /cmov1 /tank /dga /cmova /ccva /cpumpa /dgb} + \text{epumpa ecvb /cmov1} \\ & \text{/tank /dga /cmova /ccva /cpumpa /dgb} + \text{dga /cmov1 /tank /ccvb /cmovb /cpumpb /dgb} \ . \end{aligned} \quad (13)$$

For the third sequence (ecs ccs), there are 110 cut sets of up to four basic events each. To compute this exactly, 110 passes of Eq. (5b), evaluating 2^{110} (1.298×10^{33}) terms, would be necessary. This is quite a bit more than the 2^{17} evaluations of the whole system required by the brute force method. For typical problems in nuclear accident analysis, only a few passes of Eq. (5b) are required to obtain reasonable results.

The results for each case are shown in Tables III, IV, and V. All of the aforementioned methods were calculated using specific-purpose routines written in Java. They were all run on the same machine, a 2-GHz Pentium IV.

For the reliable system, Case A shown in Table III, the cut-set methods do quite well. For the first sequence, the upper bound does well, and the exact answer is found in two passes because there are only two cut sets. The other sequences also compare well to the brute force method, and the exact answer converges to six decimal places with only three passes.

For Case B shown in Table IV, the minimal cut-set upper bound is only off a bit compared to the brute force answer, but the second and third sequences require a few more passes to converge.

For Case C shown in Table V, the sum of the minimal cut-set upper bound for the three sequences is 1.36, clearly indicating a poor approximation. Even after 5.5 h to complete six passes, the second and third sequences are nowhere close to being converged.

II.B. Methods for Unreliable Systems

Obviously, the only way to improve the cut-set methodology is to include some of the terms in Eq. (5b) that

TABLE III

Results of the SAPHIRE Example Problem, Case A

		Cut-Set Methods					
				Equation (5b)			
	Exact	Rare Events	Minimal Cut Upper Bound	Pass 1	Pass 2	Pass 3	Pass 4
Seq 1	0.978799	1.922789	0.998604	1.922789	0.978799	0.978799	0.978799
Seq 2	0.020444	0.021471	0.021430	0.021471	0.020443	0.020444	0.020444
Seq 3	0.000757	0.000765	0.000765	0.000765	0.000757	0.000757	0.000757
Time[a]				0.03	0.06	1.34	41.5

[a]Measured in seconds.

TABLE IV

Results of the SAPHIRE Example Problem, Case B

		Cut-Set Methods					
				Equation (5b)			
	Exact	Rare Events	Minimal Cut Upper Bound	Pass 1	Pass 2	Pass 3	Pass 4
Seq 1	0.775422	1.312375	0.887109	1.312375	0.775422	0.775422	0.775422
Seq 2	0.156047	0.167554	0.164225	0.167554	0.155467	0.156274	0.155991
Seq 3	0.068531	0.076675	0.074701	0.076675	0.067969	0.068566	0.068527
Time[a]				0.03	0.06	1.36	41.5

[a]Measured in seconds.

TABLE V

Results of the SAPHIRE Example Problem, Case C

		Cut-Set Methods							
				Equation (5b)					
	Exact	Rare Events	Minimal Cut Upper Bound	Pass 1	Pass 2	Pass 3	Pass 4	Pass 5	Pass 6
Seq 1	0.379904	0.449060	0.399268	0.4490	0.3799	0.3799	0.3799	0.3799	0.3799
Seq 2	0.209737	0.389429	0.327886	0.3894	0.0812	0.3198	0.1162	0.2813	0.1648
Seq 3	0.410358	0.975697	0.631936	0.9757	−0.4369	2.0129	−3.0632	8.4509	−18.425
Time[a]				0.03	0.06	1.38	41.2	1013	19712

[a]Measured in seconds.

were left out by the various approximation methods. The more terms that are included, the better will be the approximation. The difficulty is that the number of terms needed for a given precision is problem-specific and cannot be predicted. One would have to keep adding terms until the answer stabilized to that given precision. For unreliable systems, higher-order terms can still be significant so that a large number of terms are required to obtain precise results. To solve large unreliable systems, two methods that do not rely on cut sets are presented.

II.B.1. Brute Force Methods Revisited

Returning to the brute force method but looking at it a bit closer, one will recognize that calculating every possible combination of the basic events is a waste of time because only the common events are causing the difficulty in the problem. Looking at a typical system of fault trees in more detail (Fig. 4) reveals that of the N

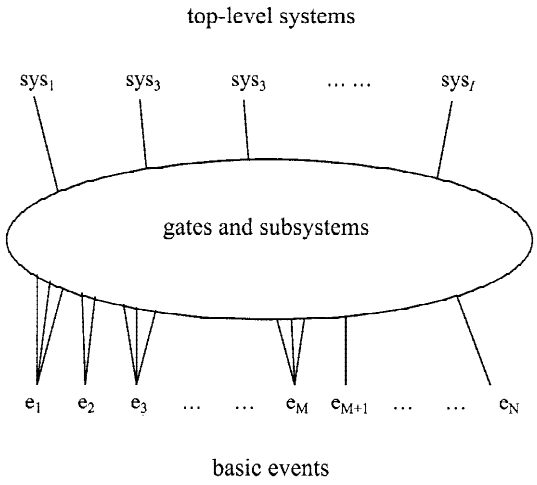

Fig. 4. A closer look at an event/fault-tree problem. Basic events N feed into a set of gates and subsystems, but only M of them are common events.

basic events, only M of them are common events, while the other N-M are each only used once.

To calculate the entire system, one really has to go through only 2^M combinations of the common events. With each combination of common events, the rest of the fault-tree system can be calculated using the rules of AND and OR gates to propagate the numbers through and obtain sequence probabilities of the event tree. These are then weighted by the probability of that combination of common events occurring. Every combination is summed up, and the exact answer is obtained, no matter if the failure probabilities are low or high. The drawback to this method is that for real systems, 2^M combinations of basic events, while it may be significantly less than 2^N, are often still too large.

The number of common basic events can be reduced if one looks at only one sequence at a time. For a typical event tree, not every top-level gate is used in the sequence. For any given sequence using a few top-level fault trees, only N^* basic events are used (with $N^* \leq N$), and only M^* ($M^* \leq M$) of them are common events (common to this set of top-level fault trees). Only 2^{M^*} combinations need to be added to obtain the final answer.

One more thing can be done to further reduce the number of common basic events in a sequence—the removal of events with probabilities of exactly 0 or 1. These events can be considered house events (events that are set failed or not failed before the analysis) and do not need to be considered two ways in the weighted sum. If there are H events that have a probability of either 0 or 1, then only $2^{(M^*-H)}$ combinations need to be added together. (Note that if basic events with 0 or 1 for failure probabilities are removed before generating cut sets, the problem size there is greatly reduced also.)

For the aforementioned SAPHIRE example problem, there are only three common events in the entire problem. Computing only one sequence at a time, the first sequence has only one common event, dga. The other sequences have three common events: tank, dga, and dgb. The computational results are identical to the brute force results but are calculated in a time of <1 s.

For vulnerability problems, the sequence-based brute force method (and removing house events) works very well. The blast problem used later in this paper contains 156 basic events, about half of which are common events. Rarely, in any of the thousands of calculations shown in the following sections, did $M^* - H$ ever exceed 15. If the number of events $M^* - H$ does become too large, then a Monte Carlo technique is used.

II.B.2. Monte Carlo Solutions Revisited

Recall that the main drawback of using Monte Carlo was the inordinate number of histories required to obtain answers with reasonable uncertainties. However, this problem arises for reliable systems, not the unreliable systems that we are concerned with here. The number of trials necessary for adequate precision depends on the magnitude of the event-tree sequence probabilities we want to estimate (but not on the details or level of complexity in the event/fault trees). Similar to the brute force techniques, we can use Monte Carlo on every basic event, on only the common events, or in a sequence-based approach.

The easiest way to program a simulation is to use the probability of each basic event to determine a failed or not-failed state for that event. With every event assigned a failure probability of 1 or 0, the fault trees can easily be evaluated, and exactly one sequence in the event tree will be taken. Calculations of the fault trees and event tree are quick because they can be done with integer arithmetic. The probability p of any sequence in the event tree is found to be the number of trials n resulting in that sequence divided by the total number of trials H. In these calculations, it is assumed that all of the basic event probabilities are statistically independent and H is relatively large. It is also assumed that the normal approximation to the binomial distribution is valid, which requires that p is not very close to 0 or 1 and enough Monte Carlo trials are conducted so that both np and $n(1-p)$ are >10 (Ref. 7).

In an analog Monte Carlo game, one can show that for an outcome of estimated probability $p = n/H$, the uncertainty in that estimate (1σ) is[7]

$$\sigma = \sqrt{\frac{(1-p)p}{H}} \quad . \tag{14}$$

For a given uncertainty σ (something like 0.01), one could find the total number of histories required to achieve that uncertainty to be

$$H = \frac{(1-p)p}{\sigma^2} \quad . \tag{15}$$

For example, after only a thousand Monte Carlo trials, the probability of some sequence was 0.412 with an uncertainty (1σ) of 0.015. If we wish to reduce the uncertainty to 0.005, we would need a total of 9690 histories, or 8690 more.

If a sequence probability p is close enough to 0 or 1 that $np < 10$ or $n(1-p) < 10$, the normal approximation to the binomial distribution ceases to be valid. Equation (14) then underestimates the uncertainty in p. Under these conditions, the binomial distribution is best approximated by a Poisson distribution, and the uncertainty in p can be approximately bounded using a χ^2 technique.[7] However, this situation seldom occurs in facility vulnerability analysis.

In our problems, sequence probabilities generally ranged from a few percent to almost 50%, and it was found that 5000 histories gave 1σ uncertainties of <0.01 in $\sim$1 s. Given all other uncertainties involved in modeling a terrorist attack, this level of precision was

considered adequate for our needs and for typical users of the software we developed to analyze such scenarios.

A variation on the aforementioned standard Monte Carlo method is to pick failed/not-failed states only for the common events. Here also, house events can be removed from the list of common events, reducing the number even further. These values can be combined with the normal failure probabilities of the other events, and a probability can be found for each sequence in the event tree. This has two advantages over Monte Carlo with every basic event. First, not as many events will be chosen as random variables, making things a bit quicker. The second advantage is that with every history, each sequence of the event tree gets some score, resulting in a lower variance for the same number of histories.

II.B.3. Example Problem

The same SAPHIRE example problem can be used to demonstrate the methods for unreliable systems. Using a brute force method but recognizing that only 3 of the 17 basic events are common events should reduce the calculation time by a factor of $\sim 2^{17}/2^3$. Of course, with the overhead time associated with loading the problem data, the actual speedups are not that large. Results for the three cases of the SAPHIRE example problem are shown in Tables VI, VII, and VIII.

For the brute force methods, the computation time does not depend on the failure probability values of the basic events. For each case, the brute force method using just the three common components calculated the same answer as the standard brute force method but more than 300 times more quickly.

To demonstrate the Monte Carlo techniques, the program was told to treat all 17 basic events as common components. Two calculations were run for each case of the SAPHIRE example problem, and these results are also shown in Tables VI, VII, and VIII. The first case requested that the maximum uncertainty (errmax) for each sequence be ≤ 0.01. The minimum number of trials for any of the calculations was set to 1000, and the actual number of trials used was determined by the uncertainties of the results as the calculations progressed. These cases all ran with times slightly smaller than the two-pass cut-set methods. The results match the exact answers fairly well, given that the requested uncertainty

TABLE VI

Results of the SAPHIRE Example Problem, Case A

	Brute Force Method			Monte Carlo Method		
	Commons	17	3	Errmax Histories	0.01 1000	0.001 19800
Seq 1 Seq 2 Seq 3		0.978799 0.020444 0.000757	0.978799 0.020444 0.000757		0.9830 ± 0.0041 0.0170 ± 0.0041 0.0 ± 0.0	0.9798 ± 0.0100 0.0197 ± 0.0100 0.0004 ± 0.0001
Time[a]		2.3	0.0063		0.22	0.33

[a]Measured in seconds.

TABLE VII

Results of the SAPHIRE Example Problem, Case B

	Brute Force Method			Monte Carlo Method		
	Commons	17	3	Errmax Histories	0.01 1800	0.001 185000
Seq 1 Seq 2 Seq 3		0.775422 0.156047 0.068531	0.775422 0.156047 0.068531		0.7744 ± 0.0098 0.1617 ± 0.0086 0.0639 ± 0.0057	0.7750 ± 0.0010 0.1570 ± 0.0009 0.0680 ± 0.0006
Time[a]		2.2	0.0062		0.036	3.5

[a]Measured in seconds.

TABLE VIII

Results of the SAPHIRE Example Problem, Case C

	Brute Force Method			Monte Carlo Method		
	Commons	17	3	Errmax Histories	0.01 2500	0.001 24500
Seq 1 Seq 2 Seq 3		0.379904 0.209737 0.410358	0.379904 0.209737 0.410358		0.3696 ± 0.0096 0.2024 ± 0.0080 0.4280 ± 0.0099	0.3800 ± 0.0010 0.2097 ± 0.0008 0.4103 ± 0.0010
Time[a]		2.2	0.0063		0.048	4.5

[a]Measured in seconds.

was only 0.01. Note that no uncertainty estimate can be found for the values given by the cut-set methods.

The second Monte Carlo calculation used a maximum uncertainty of 0.001, ten times as small as the first calculation, which should require 10^2 times as many histories and ten times as much time. This is shown to be true, except for the least severe damage case, where the minimum number of histories was used.

III. BLAST MODELING

In addition to the ability to evaluate unreliable event/fault-tree systems accurately, VISAC must be able to calculate blast damage to concrete structures and plant critical components. There are two common approaches for numerical modeling of blast effects: One method involves hydrocodes such as CTH and DYNA-3D, which are based on first-principle solutions for the conservation equations of mass, momentum, and energy in the shock-wave interactions, combined with sophisticated equations of state for the materials involved. While hydrocode calculations are excellent for looking at the details of specific shock-wave behavior, the computer run times required are too long to use them for calculations involving large numbers of components or structures. The alternative approach makes use of empirical correlations based on experimental test data to represent blast effects on components and structures, as done in the EVA-3D (Ref. 8) and MEVA (Ref. 9) family of weapons-effects codes. The second approach is the blast modeling methodology selected for incorporation into VISAC. Using correlations allows VISAC to analyze overall nuclear plant vulnerability in a fast-running code without trying to reduce the analysis to first principles.

The blast assessment algorithms programmed into VISAC by ORNL were adapted from correlations that have been used for several years in the EVA-3D (Ref. 8) and MEVA (Ref. 9) codes. These correlations actually date back to empirical test data on weapons effects that were generated by the National Defense Research Committee (NDRC) during World War II (Ref. 10). Curve fits to the NDRC test data (which is now declassified) are available for concrete wall breach probability, blast overpressure as a function of scaled distance from an explosion, and expected overpressure enhancement due to reflective surfaces surrounding the charge. For example, the peak overpressure P of a blast in air as a function of charge weight w and distance from the blast r can be found to follow the approximate rule:

$$P = 1307.3(r/w^{1/3})^{-2.2715} , \qquad (16)$$

where the charge weight is in pounds of TNT equivalent, the standoff distance is in feet, and the pressure is in pounds per square inch. This overpressure will breach a concrete wall of thickness t (in feet) if

$$t \leq \frac{w^{1/3}}{5.56(r/w^{1/3}) + 2.1} . \qquad (17)$$

Reducing the NDRC data to manageable formulas like these is made possible by relying on Hopkinson or cube root scaling[11] to correlate the blast phenomena in terms of a small number of normalized parameters.

Once VISAC has applied the NDRC correlations to determine which concrete walls are breached by a blast and how the shock overpressure is propagated through air spaces exposed to the explosion, it is necessary to calculate kill probabilities for the critical components. These component failure probabilities then serve as input for the event/fault-tree models that assess overall facility kill probability and the potential for a radiological release from the facility. Thus, each critical component in the VISAC facility model must have an associated fragility function expressed in terms of blast overpressure.

An overpressure fragility function consists of a plot showing the component kill probability versus the peak overpressure experienced by the component, as seen in the example function given by Fig. 5. Typically, fragility

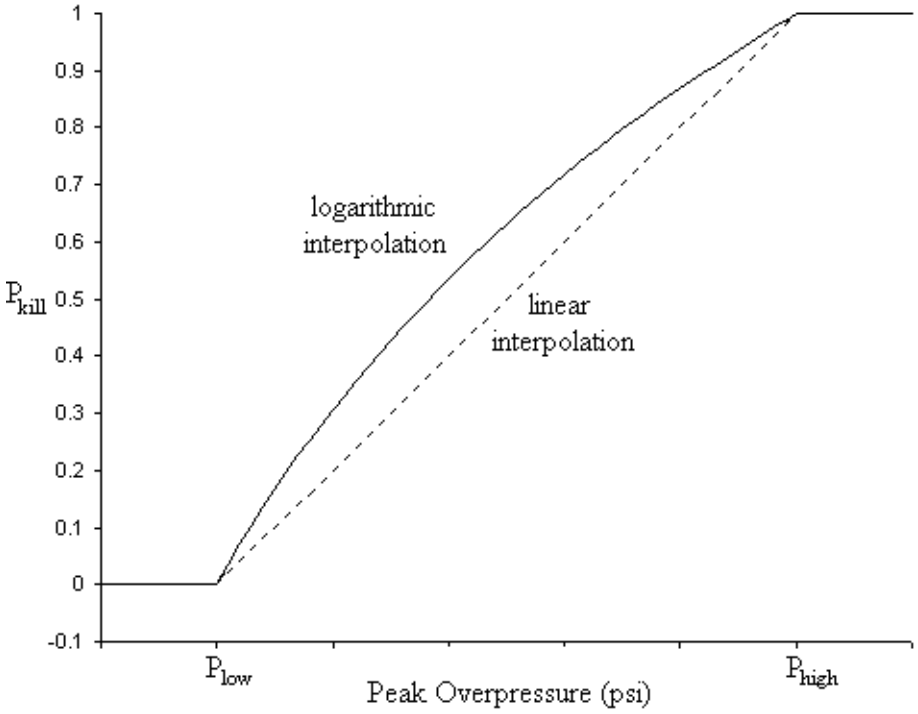

Fig. 5. Example showing the general form for a component fragility function in terms of the blast peak overpressure.

functions are defined by specifying a minimum overpressure P_{low}, below which the component kill probability is zero and a maximum overpressure P_{high}, above which it is unity. The fragility function for a component is then interpolated between P_{low} and P_{high} using either a linear approximation or a logarithmic fit of the form

$$P_{kill} = \log(P/P_{low})/\log(P_{high}/P_{low}) \qquad (18)$$

so that any component exposed to an overpressure P between P_{low} and P_{high} is assigned a fractional kill probability between 0 and 1. Estimates of overpressure fragilities for various categories of equipment can be found in Refs. 8, 12, and 13. These sources were used to develop the fragility functions for blast modeling in VISAC.

III.A. Blast Modeling in VISAC

III.A.1. Uniform Ray Tracing

Uniform ray-tracing mode sends out rays evenly spaced in three dimensions from the blast location. Each ray is followed from the blast outward. If the ray intersects a concrete region, either the concrete is broken and a hole installed or the ray stops (insufficient energy to penetrate), as determined by the thickness of the wall, the distance from the blast location, and the power of the blast. If the ray intersects a critical component, a fractional kill probability is assigned based on the component's overpressure fragility function and the calculated peak overpressure at the component's location.

Lists of the broken walls and damaged components are maintained. If a component is struck by more than one ray, the total failure probability for that component is calculated as the independent union of the kill probabilities for each path by which an overpressure is propagated to the component.

The user sets only the number of rays in the horizontal plane. A similar spacing is then used to generate oblique rays outside the horizontal plane. The more rays that are thrown, the more accurate the calculation will be. The time of calculation is proportional to the total number of rays cast. For larger blasts in these extensive models, many rays may need to be thrown to ensure accurate results, because blast effects will propagate out to a considerable distance from the charge where the rays will be more diffuse.

III.A.2. Direct Ray Tracing

Direct ray-tracing mode sends out rays in the horizontal plane, one ray vertically up and one ray vertically down, and a ray aimed at the centroid of the bounding box for each critical component. Once the ray directions are chosen, everything else proceeds like the uniform ray tracing. For determining component failure, the distance between the blast and the first intersection of the ray with the component is used. With small components

and large blasts, this mode may miss fewer components than uniform ray tracing, getting better results in less time.

III.A.3. Continuous Air Regions

Occasionally, a region of thick concrete between the blast location and a critical component will prevent the aforementioned algorithms from recording a critical component kill, even if there is a continuous air region that would have propagated the real blast wave. For this reason, a third mode is available in VISAC—the continuous air region mode, similar in concept to the EVA-3D methodology.

The user selects the number of rays to be cast out in the horizontal direction, and a similar spacing is used to generate oblique rays in three-space. These rays are followed, but only a list of broken walls and holes in walls is maintained. The rays do not directly kill critical components. Once all of the holes in walls have been made, a list is made of air regions that are continuous with the air region containing the blast location. Each critical component is then checked to see if it exists in a region of air that is listed as connected to the blast. A probability of component failure is assigned as in the other damage propagation modes, using the distance between the blast location and the component's bounding box centroid for the range. As currently implemented, the probability of wall breakage is not factored into the final component failure.

Since the distance to the centroid of the component is used instead of throwing another ray to determine the closest distance from the blast to a point on the component, some small differences in the component kill probabilities will be seen with this mode of calculation.

III.B. Example Facility

The facility used in the following blast examples is a generic two-loop pressurized water reactor (PWR) (that does not represent any real facility). Five buildings are modeled: the containment building, the auxiliary building, the turbine building, the transformer building, and the screenhouse (which takes in water from a lake). These five buildings require 695 geometric solids (cones, cylinders, spheres, rectangular parallelepipeds, etc.), combined to form 482 physical regions, each made of concrete, steel, air, or soil. Of these physical regions, 167 are critical component regions that do not have a specified material because their damage is defined by their overpressure fragilities, some of which are shown in Table IX. The critical component regions are mapped into 156 basic events in the event/fault-tree model. Twenty-two top-level fault trees and 113 intermediate gates are defined using the basic events. Six event trees are used to determine the probability of two ultimate outcomes: facility kill and core damage. Facility kill represents the state where the facility is no longer able to operate and produce electricity. Core damage represents the state where cooling of the core has been interrupted sufficiently to cause fuel damage and the potential for a radiological release. At this time, the progress of severe accidents and the various containment mechanisms that prevent releases from the core to the environment are not modeled.

TABLE IX

Overpressure Fragility Data for the Critical Components Damaged in the Example Problems

	Component	P_{low} (psi)	P_{high} (psi)
41603	4160V bus 3	4.5	10
41604	4160V bus 4	4.5	10
A1A	Accumulator 1a	12	14
A1B	Accumulator 1b	12	14
AH1B	Atmospheric dump valve header 1b	6	10
AV1B1	Atmospheric dump valve 1b1	6	10
AV1B2	Atmospheric dump valve 1b2	6	10
AV1B3	Atmospheric dump valve 1b3	6	10
COND	Condenser	7.5	9
CONDP1A	Condensate pump 1a	12	16
CONDP1B	Condensate pump 1b	12	16
EXCIT	Exciter	5	20
GEN	Generator	5	20
HPT	High-pressure turbine	7.5	20
LPT1	Low-pressure turbine 1	7.5	20
LPT2	Low-pressure turbine 2	7.5	20
MAT	Main auxiliary transformer	4.5	10
MSR1A	Moisture separator and reheater 1a	3	7.5
MSR1B	Moisture separator and reheater 1b	3	7.5
MSR2A	Moisture separator and reheater 2a	3	7.5
MSR2B	Moisture separator and reheater 2b	3	7.5
MT1A	Main transformer 1a	4.5	10
MT1B	Main transformer 1b	4.5	10
MT1C	Main transformer 1c	4.5	10
P	Pressurizer	12	14
RAT	Reserve auxiliary transformer	4.5	10
RCSP1A	Primary coolant pump 1a	12	16
RCSP1B	Primary coolant pump 1b	12	16
SG1A	Steam generator 1a	12	14
SG1B	Steam generator 1b	12	14
SPTRN	Spare transformer	4.5	10
TAT	Tertiary auxiliary transformer	4.5	10
TEXH1	Turbine exhaust line 1	6	9
TEXH2	Turbine exhaust line 2	6	9
V	Reactor vessel	12	14

III.B.1. Example 1

As an example, consider a 100-lb charge of TNT located in the containment building of a generic two-loop PWR (that does not represent any real facility). The blast location is shown in Fig. 6. The damage caused by the blast was calculated in VISAC by each of the three

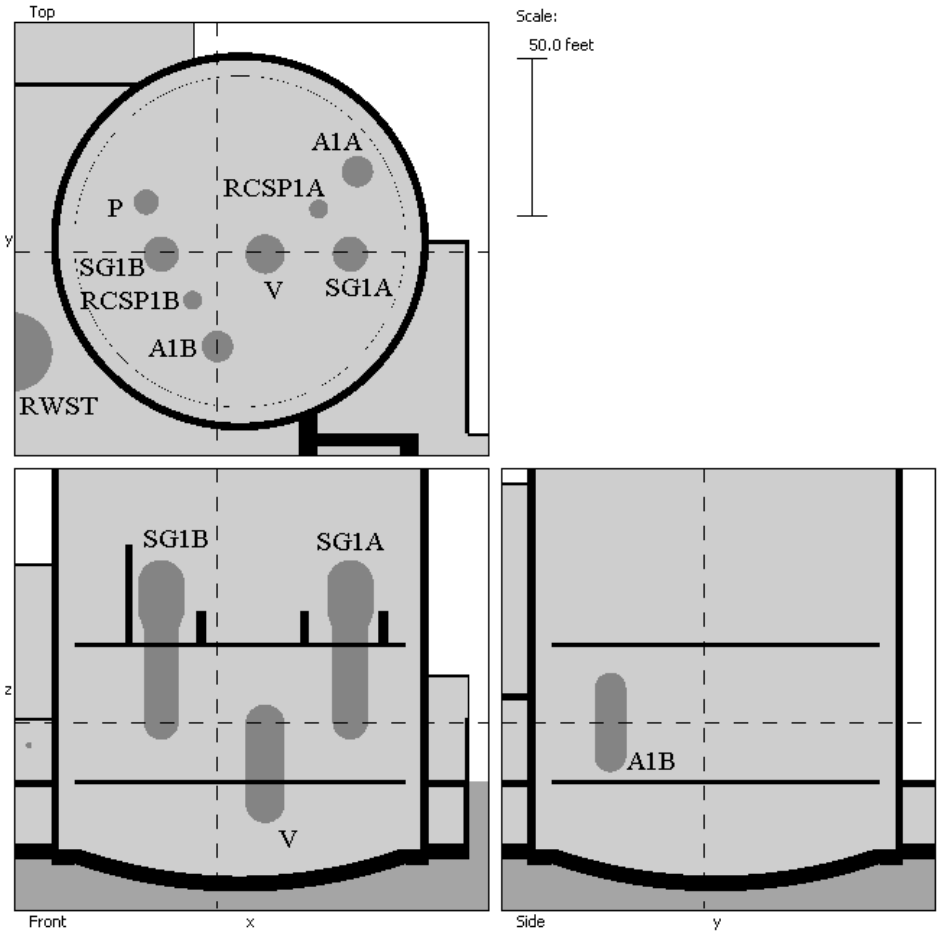

Fig. 6. Location of the 100-lb TNT charge (at the intersections of the dashed lines) inside the containment building.

modes using different numbers of cast rays (except for the direct mode, where the number of rays is equal to the number of components, 167). The results appear in Table X.

Each calculation took only a few seconds to finish. The results are fairly similar, except for the primary coolant pump 1a (RCSP1A). This pump is located such that the blast pressure on the component's surface varies from just above the minimum overpressure to slightly above its maximum overpressure. In the uniform ray-tracing mode with only 6 or 12 rays, the pump is not hit by a ray. With 48 rays, the pump is hit five times, one of which gives a failure probability of 1. In direct mode, only one ray hits the pump, and at that location, the distance is such that the failure probability is 0.93. In the continuous air region mode, the distance used in the calculation is a bit longer because it is the distance to the centroid, giving a failure probability of 0.23.

With this small blast in the containment, no walls were determined to be broken by any of the methods. Because only the components in a single air region were ever exposed to the blast, the results for the continuous air region mode were the same independent of the number of rays thrown.

This example was chosen to highlight the differences in the three algorithms. In most of the cases encountered in this project, the differences between algorithms were small. In nearly all of the cases, the final determination of core damage and facility kill were almost the same with all three algorithms.

III.B.2. Example 2

The second example consists of a 3500-lb TNT charge placed outside the turbine and transformer buildings, shown in Figs. 7a and 7b. Since the blast wave will affect components farther away than a small blast, more rays are used in the uniform ray-tracing technique. As more rays are added, more components are hit by the rays and recorded as damaged. With more rays, some components (such as the condensate pumps), are struck by rays at points closer to the blast, increasing their failure probabilities. Results of all three methods of calculation are shown in Table XI.

TABLE X

Comparisons of Component Failure Probabilities for Three Different Blast Propagation Algorithms for a 100-lb TNT Charge in the Containment Building*

			Uniform Ray Trace			Direct	Continuous Air Regions		
Rays	Vertical		2	2	2	2	2	2	2
	Plane		6	12	48	24	6	12	48
	Oblique		0	32	682	167	0	32	682
Component	P	Pressurizer		1.00	1.00	1.00	1.00	1.00	1.00
	V	Reactor vessel	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	A1A	Accumulator 1a							
	RCSP1A	Primary coolant pump 1a			1.00	0.93	0.23	0.23	0.23
	SG1A	Steam generator 1a							
	A1B	Accumulator 1b		1.00	1.00	1.00	1.00	1.00	1.00
	RCSP1B	Primary coolant pump 1b	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	SG1B	Steam generator 1b	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Outcome	Core damage		1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Facility kill		1.00	1.00	1.00	1.00	1.00	1.00	1.00

*Calculation times were only a few seconds.

The direct ray-tracing mode produces results similar to that of the uniform ray tracer with many rays. Differences in component failure rates occur because the direct method uses only one ray, directed at the centroid of the bounding box of the component, which probably does not strike the closest point on the component's surface. One downside to the direct method is shown in the example by the lack of component failure for the generator and exciter. Both of these components are mostly above a floor with a small portion below the floor. In the uniform ray-tracing cases, the portion below the floor is struck and the components are recorded as killed. In the direct ray-tracing mode, rays going to the component's centroid must pass through the concrete floor, which is

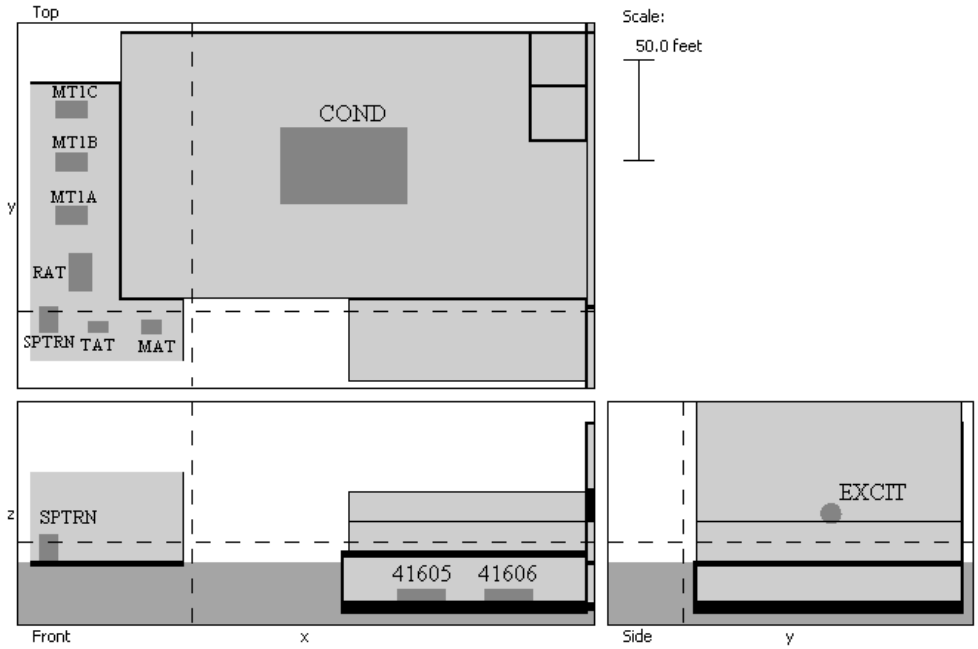

Fig. 7a. Location of the 3500-lb TNT charge outside the turbine and transformer buildings.

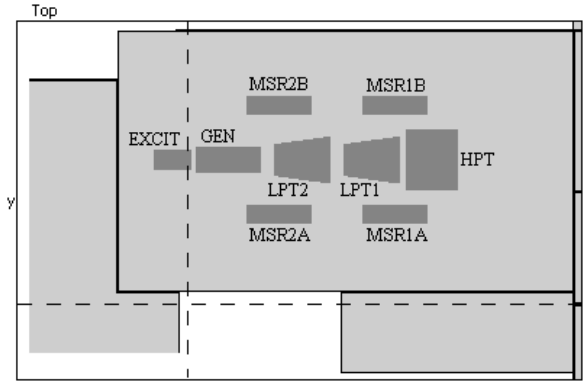

Fig. 7b. The transformer building components, 15 ft above the plane where the blast took place.

not broken by the charge at this distance. Therefore, the generator and exciter are not recorded as killed.

The continuous air region mode breaks a few more components in this example than the ray-tracing modes—the main transformers are the best example. For the ray-tracing methods, the rays striking the main transformers must pass through two walls, the second of which is not broken by the blast due to its distance from the blast. In the air region mode, the exterior wall of the transformer building is broken very close to the blast, exposing everything contained in the building to the blast. The atmospheric dump valves and moisture separators are similarly behind floors that are not broken in the ray-trace model but are broken close to the blast in the air region model.

TABLE XI

Comparisons of Component Failure Probabilities for Three Different Blast Propagation Algorithms for a 3500-lb TNT Charge Outside the Turbine and Transformer Buildings

			Uniform Ray Trace			Direct	Continuous Air Regions
Rays	Vertical Plane Oblique		2 24 158	2 48 682	2 96 2836	2 24 167	2 24 158
Component	41603	4160V bus 3				0.997	
	41604	4160V bus 4				0.997	0.106
	AH1B	Atmospheric dump valve header 1b					0.115
	AV1B1	Atmospheric dump valve 1b1					0.180
	AV1B2	Atmospheric dump valve 1b2					0.114
	AV1B3	Atmospheric dump valve 1b3					0.052
	COND	Condenser	1.000	1.000	1.000	1.000	1.000
	CONDP1A	Condensate pump 1a	0.01	0.01	0.322	0.064	1.000
	CONDP1B	Condensate pump 1b		0.551	0.861	0.355	1.000
	EXCIT	Exciter	0.602	1.000	1.000		1.000
	GEN	Generator	0.602	1.000	1.000		1.000
	HPT	High pressure turbine					0.038
	LPT1	Low pressure turbine 1			1.000	1.000	0.608
	LPT2	Low pressure turbine 2			1.000	1.000	1.000
	MAT	Main auxiliary transformer	1.000	1.000	1.000	1.000	1.000
	MSR1A	Moisture separator and reheater 1a					1.000
	MSR1B	Moisture separator and reheater 1b					1.000
	MSR2A	Moisture separator and reheater 2a	0.602	0.937	1	0.973	1.000
	MSR2B	Moisture separator and reheater 2b					1.000
	MT1A	Main transformer 1a	0.002	0.092	0.676	0.061	1.000
	MT1B	Main transformer 1b					1.000
	MT1C	Main transformer 1c					1.000
	RAT	Reserve auxiliary transformer	0.898	1.000	1.000	0.975	1.000
	SPTRN	Spare transformer	1.000	1.000	1.000	1.000	1.000
	TAT	Tertiary auxiliary transformer		1.000	1.000	1.000	1.000
	TEXH1	Turbine exhaust line 1	0.602	0.602	0.841	0.238	1.000
	TEXH2	Turbine exhaust line 2			0.997	0.956	1.000
Outcome	Core damage		0.046	0.045	0.045	0.044	0.045
	Facility kill		0.602	1.000	1.000	1.000	1.000

III.B. Geometry Fidelity

Geometric fidelity of the model determines the resolution of the final vulnerability analysis results. Instead of modeling the various critical components, one could simply model each building as a single basic event. This would simplify the fault trees a great deal, but it imposes the assumption that if any portion of the building were damaged, every safety system in that building is failed.

An example is shown in Figs. 8 and 9. Consider a generic two-loop PWR in the first case with all of its critical components removed and fault trees that have been reformulated to have only one basic event per building. The second case is a fully detailed model with all safety systems modeled as functions of the critical components. To compare the two cases, a 3500-lb TNT charge was placed 10 ft above the ground level at many locations around the buildings, simulating a truck bomb. The figures show those areas that lead to core damage and a potential radiological release from an attack of this type.

Contrast Fig. 8, the building-level model, with the results shown in Fig. 9 for a detailed model. The areas that lead to core damage are much smaller in the detailed model, showing that the building-level model was considerably overpredicting the vulnerability. This situation arises because actual vehicle bombs of realistic size almost never kill all the critical components within a building, as is implicitly assumed in the building-level analysis.

IV. CONCLUSIONS

Traditional event/fault-tree analysis techniques do not work well when input failure probabilities are high and the system is unreliable. Improvements to the cut-set methodology would be difficult because of the large number of terms involved.

For systems where the number of common events in each event-tree sequence is small, reliable, or unreliable, a variation of the brute force technique can be used to find the exact answer. This paper has shown several improvements to the brute force technique, but some problems can still be too large to obtain solutions in reasonable periods of time.

Monte Carlo methods are well suited to analyzing unreliable systems. Calculations can be made to any level of uncertainty desired. Like the brute force techniques, by focusing on the common events, a solution with lower variance can be calculated.

Correlation-based algorithms using scaled parameters allow a fast-running code to represent blast effects without resorting to hydrocode solutions.

The different blast modeling approaches used can also affect the end results. Modeling only the buildings without any critical components will give only a rough idea of the true facility vulnerability, which may not be particularly useful in designing countermeasures. Including geometric detail in the target facility model is required to obtain meaningful vulnerability analysis results.

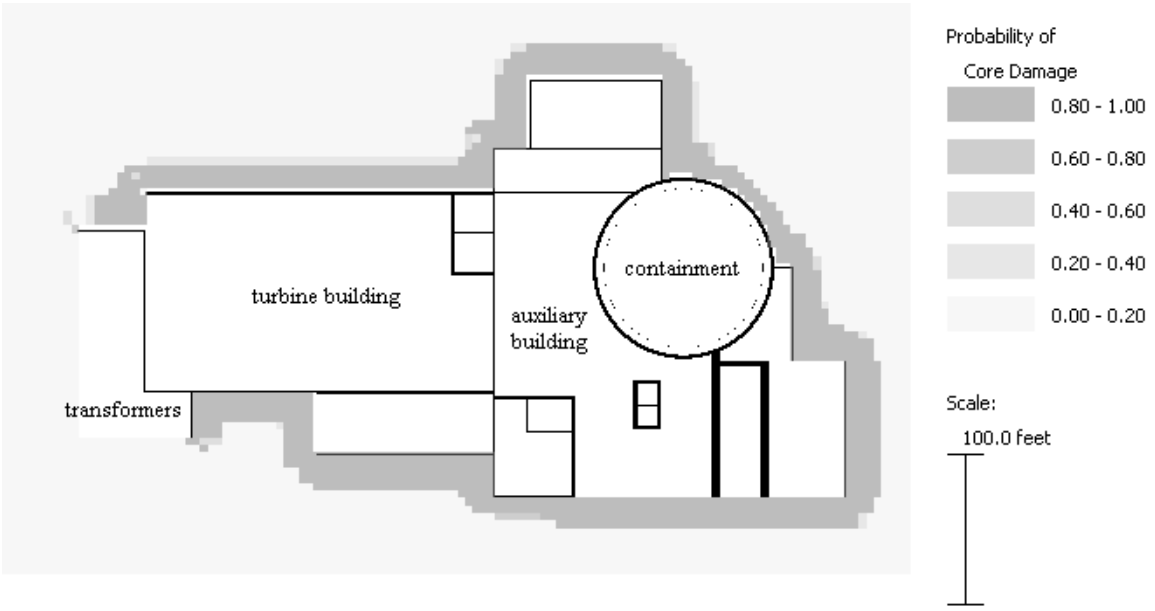

Fig. 8. The core damage probabilities resulting from a truck bomb using building-level models. This indicates that an explosion next to any building other than the transformer building will lead to core damage.

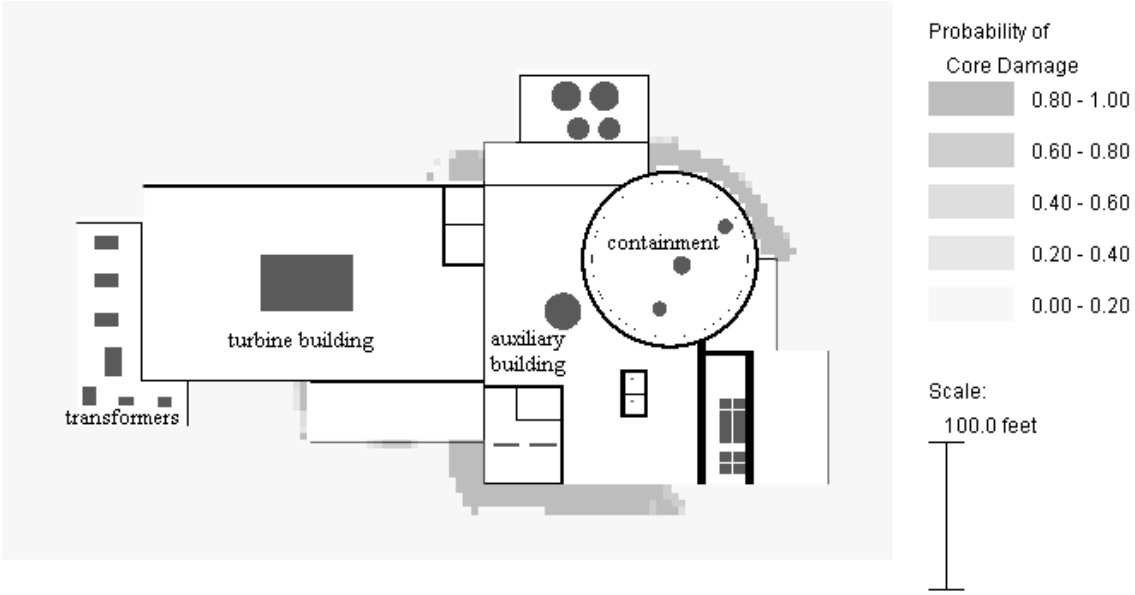

Fig. 9. The core damage probabilities resulting from a truck bomb using detailed critical components inside each building. This indicates that explosions next to the containment, next to the control room, and near critical equipment in the auxiliary building will lead to core damage.

ORNL's VISAC code successfully integrates the concepts of target geometric modeling, a correlation-based methodology for blast damage assessment, and unreliable event/fault-tree evaluation techniques to analyze nuclear power plant vulnerability.

ACKNOWLEDGMENTS

This work was funded by the Defense Threat Reduction Agency, Alexandria, Virginia.

Special thanks are extended to R. T. Santoro and R. J. Ellis of ORNL for reviewing the manuscript before its submission.

REFERENCES

1. "Reactor Safety Study—An Assessment of Accident Risks in U.S. Commercial Nuclear Power Plants," WASH-1400, NUREG-75/014, U.S. Nuclear Regulatory Commission (1975).

2. D. F. HAASL, N. H. ROBERTS, W. E. VESELY, and F. F. GOLDBERG, "Fault Tree Handbook," U.S. Nuclear Regulatory Commission (1981).

3. "SAPHIRE: Systems Analysis Programs for Hands-on Integrated Reliability Evaluations, Version 6.64," Idaho National Engineering and Environmental Laboratory (1999).

4. "FaultTree+ V8.0," Isograph (1998).

5. R. R. FULLWOOD and R. E. HALL, *Probabilistic Risk Assessment in the Nuclear Power Industry*, Pergamon Press, Oxford, England (1988).

6. N. J. McCORMICK, *Reliability and Risk Analysis*, Academic Press, New York (1981).

7. I. MILLER and J. E. FREUND, *Probability and Statistics for Engineers*, pp. 79, 108, and 240–246, Prentice-Hall, Englewood Cliffs, New Jersey (1976).

8. L. A. YOUNG, B. K. STREIT, K. J. PETERSON, D. L. READ, and F. A. MAESTAS, "Effectiveness/Vulnerability Assessments in Three Dimensions (EVA-3D) Versions 4.1F and 4.1C User's Manual—Rev. A," Wright Laboratory, U.S. Air Force (Nov. 29, 1995).

9. P. E. DUNN, J. E. MADRIGAL, D. A. PARSONS, J. C. PARTCH, D. A. VERNER, and L. A. YOUNG, "Modular Effectiveness/Vulnerability Assessment (MEVA) Software User's Manual," Wright Laboratory, U.S. Air Force (Apr. 23, 1999).

10. "Effects of Impact and Explosion," M. P. WHITE, (Ed.), Office of Scientific Research and Development (1946).

11. W. E. BAKER, *Explosions in Air*, University of Texas Press, Austin, Texas (1973).

12. S. GLASSTONE, "The Effects of Nuclear Weapons," U.S. Atomic Energy Commission (1962).

13. M. M. STEPHENS, "Minimizing Damage to Refineries from Nuclear Attack, Natural and Other Disasters," U.S. Department of the Interior (1970).