Failure Analysis of a Complex System Based on Partial Information about Subsystems, with Potential Applications to Aircraft Maintenance

Vladik Kreinovitch, Christelle Jacob, Didier Dubois, Janette Cardoso, Martine Ceberio, Ildar Batyrshin

Appl. Comput. Math., V.xx, N.xx, 20xx, pp.xx-xx

FAILURE ANALYSIS OF A COMPLEX SYSTEM BASED ON PARTIAL INFORMATION ABOUT SUBSYSTEMS, WITH POTENTIAL APPLICATIONS TO AIRCRAFT MAINTENANCE

V. KREINOVICH[1], C. JACOB[2,3], D. DUBOIS[3], J. CARDOSO[2], M. CEBERIO[1], I. BATYRSHIN[4]

Abstract. In many real-life applications (e.g., in aircraft maintenance), we need to estimate the probability of failure of a complex system (such as an aircraft as a whole or one of its subsystems). Complex systems are usually built with redundancy allowing them to withstand the failure of a small number of components. In this paper, we assume that we know the structure of the system, and, as a result, for each possible set of failed components, we can tell whether this set will lead to a system failure. For each component A, we know the probability $P(A)$ of its failure with some uncertainty: e.g., we know the lower and upper bounds $\underline{P}(A)$ and $\overline{P}(A)$ for this probability. Usually, it is assumed that failures of different components are independent events. Our objective is to use all this information to estimate the probability of failure of the entire the complex system. In this paper, we describe several methods for solving this problem, including a new efficient method for such estimation based on Cauchy deviates.

Keywords: complex system, probability of failure, interval uncertainty.

AMS Subject Classification: 90B25, 68T37, 65G40, 60K10, 62N05

1. Formulation of the Problem

It is necessary to estimate the probability of failure for complex systems. In many practical applications, we need to estimate the probability of failure of a complex system. The need for such estimates comes from the fact that in practice, while it is desirable to minimize risk, it is not possible to completely eliminate it: no matter how many precautions we take, there are always some very low probability events that can potentially lead to a system's failure. All we can do is to make sure that the resulting probability of failure does not exceed the desired small value p_0. For example, the probability of a catastrophic event is usually required to be at or below $p_0 = 10^{-9}$.

In aircraft design and maintenance, we need to estimate the probability of a failure of an aircraft as a whole and of its subsystems. At the design stage, the purpose of this estimate is to make sure that this probability of failure does not exceed the allowed probability p_0. At the maintenance stage, this estimate helps to decide whether a maintenance is needed: if the probability of failure exceeds p_0, some maintenance is required to bring this probability down to the desired level p_0 (or below).

[1]University of Texas at El Paso, Computer Science Department, 500 W. University, El Paso, TX 79968, USA
e-mail: vladik@utep.edu, mceberio@utep.edu
[2]Institut Supérieur de l'Aéronautique et de l'Espace (ISAE), DMIA department, Campus Supaéro, 10 avenue Édouard Belin, Toulouse, France
e-mail: jacob.christelle.11@gmail.com, cardoso@isae.fr
[3]Institut de Recherche en Informatique de Toulouse (IRIT), 118 Route de Narbonne 31062 Toulouse Cedex 9, France
e-mail: dubois@irit.fr
[4]Instituto Mexicano de Petróleo, Ejec Central Lázaro Cardenas Norte 152, Col. San Bartolo Atepehuacan México D.F., C.P. 07730
e-mail: batyr@imp.mx
Manuscript received xx.

Information available for estimating system's probability of failure: general description. Complex systems consist of subsystems, which, in turn, consist of components (or maybe of subsubsystems which consist of components). So, to estimate the probability of failure of a complex system, we need to take into account when the failure of components and subsystems lead to the failure of the complex system as a whole, and how reliable are these components and subsystems.

From the failure of components and subsystems to the failure of the complex system as a whole. Complex systems are usually built with redundancy allowing them to withstand the failure of a small number of components. Usually, we know the structure of the system, and, as a result, for each possible set of failed components, we can tell whether this set will lead to a system failure. So, in this paper, we will assume that this information is available.

How reliable are components and subsystems? What do we know about the reliability of individual components? For each component A, there is a probability $P(A)$ of its failure. When we have a sufficient statistics of failures of this type of components, we can estimate this probability as the relative frequency of cases when the component failed. Sometimes, we have a large number of such cases, and as a result, the frequency provides a good approximation to the desired probability – so that, in practice, we can safely assume that we know the actual values of these probabilities $P(A)$.

If only a few failure cases are available, it is not possible to get an accurate estimate for $P(A)$. In this case, the only information that we can extract from the observation is the interval $\mathbf{P}(A) = [\underline{P}(A), \overline{P}(A)]$ that contains the actual (unknown) value of this probability.

This situation is rather typical for aircraft design and maintenance, because aircrafts are usually built of highly reliable components – at least the important parts of the aircraft are built of such components – and there are thus very few observed cases of failure of these components.

Fuzzy case. In some cases, especially on the design stage, we do not yet have failure statistics, so we have to rely on expert estimates, estimates based on the expert's experience with similar components. These estimates are usually formulated by using words from natural language, like "about 1%". A natural way to describe such expert estimates is to use fuzzy techniques (see, e.g., [12, 17]), i.e., to describe each such estimate as a *fuzzy number* $\mathcal{P}(A)$. A fuzzy number means, crudely speaking, that for different degrees of uncertainty α, we have an interval $\mathcal{P}(A, \alpha)$ that contains the actual (unknown) probability $P(A)$ with this degree of uncertainty: e.g., the interval $\mathcal{P}(A, 0)$ contains $P(A)$ with guarantee (uncertainty 0), while the interval $\mathcal{P}(A, 0.5)$ contains $P(A)$ with uncertainty 0.5.

Are the component failures independent events or they are caused by a common cause? In many practical situations, failures of different components are caused by independent factors. For example, for an aircraft, possible failures of mechanical subsystems can be caused by the material fatigue, while possible failures of electronic systems can be caused by the interference of atmospheric electricity (e.g., when flying close to a thunderstorm) – and material fatigue and atmospheric electricity are independent events. As a result, usually, it is assumed that failures of different components are independent events.

However, sometimes, we know that the failures of different components are caused by a common cause. In this case, the failures of different components are no longer independent events. In such situations, often, do not have any specific information about the correlation between these failures. This is a typical situation in aircraft design and maintenance – we have very few failures of each component, not enough to determine the exact probability of each such failure, and we have even fewer situations when two components failed, so an empirical determination of such correlations is out of question.

What we do in this paper. Our objective is to use all this information to estimate the probability of failure of the entire complex system. In this paper, we describe new methods for such estimation; some preliminary results first appeared in [8, 14].

Comment. In most of this paper, we make a usual assumption that failures of different components are independent events. Sometimes, we know that the failures of different components are caused by a common cause; corresponding algorithms are described, e.g., in [1, 2, 3, 8].

2. Simplest Case: Component Failures are Independent and Failure Probabilities $P(A)$ Are Exactly Known

Let us start our analysis with the simplest case when the component failures are independent and the failure probabilities $P(A)$ for different components A are known exactly. As we mentioned, we assume that there exist efficient algorithms that, given a list of failed components, determines whether the whole system fails or not. In this case, it is always possible to efficiently estimate the probability P of the system's failure by using Monte-Carlo simulations. Specifically, we select the number of simulations N. Then, for each component A, we simulate a Boolean variable *failing(A)* which is true with probability $P(A)$ and false with the remaining probability $1-P(A)$. This can be done, e.g., if we take the result r of a standard random number generator that generates values uniformly distributed on the interval $[0,1]$ and select *failing(A)* to be true if $r \le P(A)$ and false otherwise: then the probability of this variable to be true is exactly $P(A)$.

Then, we apply the above-mentioned algorithm to the simulated values of the variables *failing(A)* and conclude whether for this simulation, the system fails or not. As an estimate for the probability of the system's failure, we then take the ratio $p \stackrel{\text{def}}{=} f/N$, where f is the number of simulations on which the system failed. From statistics, it is known that the mean value of this ratio is indeed the desired probability, that the standard deviation can be estimated as $\sigma = \sqrt{p\cdot(1-p)/N} \le 0.5/\sqrt{N}$, and that for sufficiently large N (due to the Central Limit Theorem), the distribution of the difference $P-p$ is close to normal. Thus, with probability 99.9%, the actual value P is within the three-sigma interval $[p-3\sigma, p+3\sigma]$.

This enables us to determine how many iterations we need to estimate the probability P with accuracy 10% (and certainty 99.9%): due to $\sigma \le 0.5/\sqrt{N}$, to guarantee that $3\sigma \le 0.1$, it is sufficient to select N for which $3\cdot 0.5/\sqrt{N} \le 0.1$, i.e., $\sqrt{N} \ge (3\cdot 0.5)/0.1 = 15$ and $N \ge 225$. It is important to emphasize that this number of iterations is the same no matter how many components we have – and for complex systems, we usually have many thousands of components.

Similarly, to estimate this probability with accuracy 1%, we need $N = 22,500$ iterations, etc. These numbers of iterations work for all possible values P. In practical applications, the desired probability P is small, so $1-P \approx 1$, $\sigma \approx \sqrt{P/N}$ and the number of iterations, as determined by the condition $3\sigma \le 0.1$ or $3\sigma \le 0.01$, is much smaller: $N \ge 900\cdot P$ for accuracy 10% and $N \ge 90,000\cdot P$ for accuracy 1%.

Comment. In many cases, there are also efficient analytical algorithms for computing the desired probability of the system's failure; see, e.g., [4, 5, 6, 20].

3. Important Subcase of the Simplest Case: When Components are Very Reliable

In many practical applications (e.g., in important subsystems related to aircrafts), components are highly reliable, and their probabilities of failure $P(A)$ are very small. In this case, the above Monte-Carlo technique for computing the probability P of the system's failure requires a large number of simulations, because otherwise, with high probability, in all simulations, all the components will be simulated as working properly.

For example, if the probability of a component's failure is $P(A) = 10^{-3}$, then we need at least a thousand iteration to catch a case when this component fails; if $P(A) = 10^{-6}$, we need at least a million iterations, etc.

In such situations, Monte-Carlo simulations may take a lot of computation time. In some applications, e.g., on the stage of an aircraft design, it may be OK, but in other cases, e.g., on

the stage of routine aircraft maintenance, the airlines want fast turnaround, and any speed up is highly welcome.

To speed up such simulations, we can use a re-scaling idea; see, e.g., [8, 10]. Specifically, instead of using the original values $P(A)$, we use re-scaled (larger) values $\lambda \cdot P(A)$ for some $\lambda \gg 1$. The value λ is chosen in such a way that the resulting probabilities are larger and thus, require fewer simulations to come up with cases when some components fail. As a result of applying the above Monte-Carlo simulations to these new probabilities $\lambda \cdot P(A)$, we get a probability of failure $P(\lambda)$.

In this case, one can show that while the resulting probabilities $\lambda \cdot P(A)$ are still small, the probability $P(\lambda)$ depends on λ as $P(\lambda) \approx \lambda^k \cdot P$ for some positive integer k.

Thus, to find the desired value P, we repeat this procedure for two different values $\lambda_1 \neq \lambda_2$, get the two values $P(\lambda_1)$ and $P(\lambda_2)$, and then find both unknown k and P from the resulting system of two equations with two unknowns: $P(\lambda_1) \approx \lambda_1^k \cdot P$ and $P(\lambda_2) \approx \lambda_2^k \cdot P$.

To solve this system, we first divide the first equation by the second one, getting an equation $P(\lambda_1)/P(\lambda_2) \approx (\lambda_1/\lambda_2)^k$ with one unknown k, and find $k \approx \ln(P(\lambda_1)/P(\lambda_2))/(\lambda_1/\lambda_2)$. Then, once we know k, we can find P as $P \approx P(\lambda_1)/\lambda_1^k$.

4. Cases When We Know the Probabilities $P(A)$ with Uncertainty: Fuzzy Uncertainty Can Be Reduced to Interval One

In many cases, we do not know the exact probabilities $P(A)$ of the component's failure, we only know the intervals $\mathbf{P}(A)$ that contain these probabilities, or, even more generally, a fuzzy number $\mathcal{P}(A)$ that describes this probability.

In the case of intervals, different combinations of values $P(A) \in \mathbf{P}(A)$ lead, in general, to different values P. Let us denote the dependence of P on the values $P(A)$ by f: $P = f(P(A), P(B), \ldots)$. In this case, we want to know the range of possible values of the desired probability P:

$$\mathbf{P} = [\underline{P}, \overline{P}] = f(\mathbf{P}(A), \mathbf{P}(B), \ldots) \stackrel{\text{def}}{=}$$

$$\{f(P(A), P(B), \ldots) : P(A) \in \mathbf{P}(A), P(B) \in \mathbf{P}(B), \ldots\}.$$

The problem of computing such an interval is a particular case of the general problem of interval computations, i.e., a problem of computing the range of a given function $f(x_1, \ldots, x_n)$ when each of the variables x_i takes value from a given interval $\mathbf{x}_i$; see, e.g., [7, 11, 16] and references therein.

When each probability is described by a fuzzy number $\mathcal{P}(A)$, i.e., by a membership function $\mu_A(P)$ that assigns, to every real number P, a degree to which this number is possible as a value of $P(A)$, we want to find the fuzzy number $\mathcal{P}$ that describes $f(P(A), P(B), \ldots)$. A natural way to define the corresponding membership function $\mu(P)$ leads to Zadeh's extension principle:

$$\mu(P) = \sup\{\min(\mu_A(P(A)), \mu_B(P(B)), \ldots) : f(P(A), P(B), \ldots) = P\}.$$

It is known that from the computational viewpoint, the application of this formula can be reduced to interval computations.

Specifically, for each fuzzy set with a membership function $\mu(x)$ and for each $\alpha \in (0, 1]$, we can define this set's *α-cut* as $\mathcal{X}(\alpha) \stackrel{\text{def}}{=} \{x : \mu(x) \geq \alpha\}$. Vice versa, if we know the α-cuts for all α, we, for each x, can reconstruct the value $\mu(x)$ as the largest value α for which $x \in \mathcal{X}(\alpha)$. Thus, to describe a fuzzy number, it is sufficient to find all its α-cuts.

It is known that when the inputs $\mu_i(x_i)$ are fuzzy numbers, and the function $y = f(x_1, \ldots, x_n)$ is continuous, then for each α, the α-cut $\mathcal{Y}(\alpha)$ of y is equal to the range of possible values of $f(x_1, \ldots, x_n)$ when $x_i \in \mathcal{X}_i(\alpha)$ for all i:

$$\mathcal{Y}(\alpha) = f(\mathcal{X}_1(\alpha), \ldots, \mathcal{X}_n(\alpha));$$

see, e.g., [**?**, 12, **?**, 17]. This is how processing fuzzy data is often done – by reducing to interval computations.

$$\mathbf{P} = [\underline{P}, \overline{P}] = f(\mathbf{P}(A), \mathbf{P}(B), \ldots).$$

In particular, for our problem, once know the α-cuts $\mathcal{P}(A, \alpha)$ corresponding to different components A, we can find the α-cuts $\mathcal{P}(\alpha)$ corresponding to the desired probability P as the corresponding interval range:

$$\mathcal{P}(\alpha) = f(\mathcal{P}(A, \alpha), \mathcal{P}(B, \alpha), \ldots).$$

So, if we know how to solve our problem under interval uncertainty, we can also solve it under fuzzy uncertainty – e.g., by repeating the above interval computations for $\alpha = 0, 0.1, \ldots, 0.9, 1.0$.

In view of this reduction, in the following text, we will only consider the case of interval uncertainty.

5. Component Failures are Independent, Failure Probabilities $P(A)$ Are Known with Interval Uncertainty: Monotonicity Case

In view of the above analysis, let us now consider the case when the probabilities $P(A)$ are only known with interval uncertainty, i.e., for each component A, we only know the interval $[\underline{P}(A), \overline{P}(A)]$ that contains the actual (unknown) value $P(A)$. We still assume that failures of different components are independent events.

Let us start with the simplest subcase when the dependence of the system's failure is monotonic with respect to the failure of components. To be precise, we assume that if for a certain list of failed components, the system fails, it will still fail if we add one more components to the list of failed ones. In this case, the smaller the probability of failure $P(A)$ for each component A, the smaller the probability P that the system as a whole will fail. Similarly, the larger the probability of failure $P(A)$ for each component A, the larger the probability P that the system as a whole will fail.

Thus, to compute the smallest possible value $\underline{P}$ of the failure probability, it is sufficient to consider the values $\underline{P}(A)$. Similarly, to compute the largest possible value $\overline{P}$ of the failure probability, it is sufficient to consider the values $\overline{P}(A)$. Thus, in the monotonic case, to compute the range $[\underline{P}, \overline{P}]$ of possible values of overall failure probability under interval uncertainty, it is sufficient to solve two problems in each of which we know probabilities with certainty:

- to compute $\underline{P}$, we assume that for each component A, the failure probability is equal to $\underline{P}(A)$;
- to compute $\overline{P}$, we assume that for each component A, the failure probability is equal to $\overline{P}(A)$.

6. In Practice, the Dependence is Sometimes Non-Monotonic

In some practically reasonable situations, the dependence of the system's failure on the failure of components is non-monotonic; see, e.g., [8]. This may sound counter-intuitive at first glance: adding one more failing component to the list of failed ones suddenly makes the previously failing system recover, but here is an example when exactly this seemingly counter-intuitive behavior makes perfect sense. Please note that this example is over-simplified: its only purpose is to explain, in intuitive terms, the need to consider non-monotonic case.

To increase reliability, systems include duplication: for many important functions, there is a duplicate subsystem ready to take charge if the main subsystem fails. How do we detect that the main system failed? Usually, a subsystem contains several sensors; sensors sometimes fail, as a result of which their signal no longer reflect the actual value of the quantity they are supposed to measure. For example, a temperature sensor which is supposed to generate a signal proportional to the temperature, if failed, produces no signal at all, which the system will naturally interpret as a 0 temperature. To detect the sensor failure, subsystems often use statistical criteria. For

example, for each sensor i, we usually know the mean m_i and the standard deviation σ_i of the corresponding quantity. When these quantities are independent and approximately normally distributed, then, for the measurement values x_i, the sum $X^2 \stackrel{\text{def}}{=} \sum_{i=1}^{n} \frac{(x_i - m_i)^2}{\sigma_i^2}$ is the sum of n standard normal distributions and thus, follows the chi-square distributed with n degrees of freedom. So, if the actual value of this sum exceeds the threshold corresponding to confidence level $p = 0.05$, this means that we can confidently conclude that some of the sensors are malfunctioning. If the number n of sensors is large, then one malfunctioning sensor may not increase the sum X^2 too high, and so, its malfunctioning will not be detected, and the system will fail. On the other hand, if all n sensors fail, e.g., show 0 instead of the correct temperature, each term in the sum will be large, the sum will exceed the threshold – and the system will detect the malfunctioning. In this case, the second redundant subsystem will be activated, and the system as a whole will thus continue to function normally.

This is exactly the case of non-monotonicity: when only one sensor fails, the system as a whole fails; however, if, in addition to the originally failed sensor, many other sensors fail, the system as a whole becomes functioning well. Other examples of non-monotonicity can be due to the fact that some components may be in more than two states [9].

In the following text, we will consider the non-monotonic case, in which a simple algorithm (given above) is not applicable.

7. What If We Have Few Components, With Respect to Which the Dependence is Non-Monotonic

Let us start with the simplest case when there are only few components with respect to which the dependence is non-monotonic, and with respect to all other components, the dependence *is* still monotonic. In this case, for all monotonic components B, as before, we can take $P(B) = \overline{P}(B)$ when we are computing $\overline{P}$, and take $P(B) = \underline{P}(B)$ when we are computing $\underline{P}$.

For non-monotonic components A, for computing each of the values $\underline{P}$ and $\overline{P}$, we need to to take into account all possible values $P(A) \in [\underline{P}(A), \overline{P}(A)]$. For each such component, by using the formula of full probability, we can represent the probability P of the system's failure as follows:

$$P = P(A) \cdot P(F|A) + (1 - P(A)) \cdot P(F|\neg A),$$

where $P(F|A)$ is the conditional probability that the system fails under the condition that the component A fails, and $P(F|\neg A)$ is the conditional probability that the system fails under the condition that the component A does not fail. The conditional probabilities $P(F|A)$ and $P(F|\neg A)$ do not depend on $P(A)$, so the resulting dependence of P on $P(A)$ is linear. A linear function attains it minimum and maximum at the endpoints. Thus, to find $\underline{P}$ and $\overline{P}$, it is not necessary to consider all possible values $P(A) \in [\underline{P}(A), \overline{P}(A)]$, it is sufficient to only consider two values: $P(A) = \underline{P}(A)$ and $P(A) = \overline{P}(A)$.

For each of these two values, for another non-monotonic component A', we have two possible options $P(A') = \underline{P}(A')$ and $P(A') = \overline{P}(A')$; thus, in this case, we need to consider $2 \times 2 = 4$ possible combinations of values $P(A)$ and $P(A')$.

In general, when we have k non-monotonic components $A_1, \ldots, A_k$, it is sufficient to consider 2^k possible combinations of values $\underline{P}(A_i)$ and $\overline{P}(A_i)$ corresponding to each of these components. When k is small, this is doable – and, as our preliminary experiments show, works very well.

This procedure requires times which grows as 2^k. As we mentioned earlier, when k is large, the needed computation time becomes unrealistically large.

8. General Case: the Problem is NP-Hard, and Even Checking Whether a Component Is Monotonic Is NP-hard

Natural question. The fact that the above algorithm requires unrealistic exponential time raises a natural question: is it because our algorithm is inefficient or is it because the problem itself is difficult?

The problem is NP-hard. In the general case, when no assumption is made about monotonicity, the problem is as follows:

- we have a propositional formula F with n variables A_i – each variable A_i is true if the corresponding component fails, and the formula F is true if the system as whole fails;
- for each component i, we know the interval $[\underline{P}(A_i), \overline{P}(A_i)]$ that contains the actual (unknown) $P(A_i)$ that this component fails;
- we assume that the failures of different components are independent events.

Different values $P(A_i) \in [\underline{P}(A_i), \overline{P}(A_i)]$ lead, in general, to different values of the probability P that F is true (i.e., that the system failed). Our objective is to compute the range $[\underline{P}, \overline{P}]$ of possible values of this probability.

One can easily show that, in general, this problem is NP-hard (for precise definitions, see, e.g., [15, 18]). Indeed, it is well known that the following *propositional satisfiability* problem SAT is NP-hard: given a propositional formula F, check whether this formula is satisfiable, i.e., whether there exist values A_i that make it true. We will prove that our problem is NP-hard by reducing SAT to it: for every particular case F of SAT, there is a particular case of our problem whose solution leads to a solution to the F. Indeed, for every formula F, let us take $[\underline{P}(A_i), \overline{P}(A_i)] = [0, 1]$ for all variables i.

If the formula F is not satisfiable, then F is always false, so the probability of its being true is 0. In this case, the range of possible values of P consists of a single value 0: $[\underline{P}, \overline{P}] = [0, 0]$.

On the other hand, if F is satisfiable, e.g., if F is true for A_1 true, A_2 false, etc., then we can take $P(A_1) = 1$, $P(A_2) = 0$, etc., and conclude that under these probabilities, F is always true, i.e., $P = 1$. Thus, in this case, $\overline{P} = 1$.

So, by computing $\overline{P}$, we will get either $\overline{P} = 0$ or $\overline{P} = 1$. In the first case, F is satisfiable, in the second case, it is not. The reduction proves that our problem is NP-hard.

Even checking monotonicity is NP-hard. Let us show that even checking monotonicity is NP-hard. Intuitively, monotonicity with respect to a component A means that if the system was in a failing state, and we change the state of A to failing, the system remains failing. In precise terms, monotonicity means that if the formula F was true, and we change the value of the variable A from false to true, then F remains true.

Let us prove that the problem of checking monotonicity is NP-hard by reducing SAT to this problem. Indeed, for every propositional formula $F(A_1, \ldots, A_n)$, we can form a new formula $F' \stackrel{\text{def}}{=} F(A_1, \ldots, A_n) \,\&\, \neg A_0$.

When the original formula F is not satisfiable, then F is always false and thus, the new formula F' is also always false. In this case, F' is, by definition, monotonic with respect to A_0.

When F is satisfiable, this means that F is equal to "true" for some values of $A_1, \ldots, A_n$. In this case, for $A_0 =$"false", the new formula F' is true, but if we make $A_0 =$"true", F' becomes false. Thus, in this case, F' is not monotonic with respect to A_0.

Summarizing: the new formula F' is monotonic with respect to A_0 if and only if the original formula F was satisfiable. This reduction proves that checking monotonicity is indeed NP-hard.

9. A Practically Important Case when Dependence May Be Non-Monotonic but Intervals Are Narrow: Towards a New Algorithm

General non-monotonic case: a possible algorithm. For each component A, by using the formula of full probability, we can represent the probability P of the system's failure as follows:

$$P = P(A) \cdot P(F|A) + (1 - P(A)) \cdot P(F|\neg A),$$

where $P(F|A)$ is the conditional probability that the system fails under the condition that the component A fails, and $P(F|\neg A)$ is the conditional probability that the system fails under the condition that the component A does not fail. The conditional probabilities $P(F|A)$ and $P(F|\neg A)$ do not depend on $P(A)$, so the resulting dependence of P on $P(A)$ is linear. A linear function attains it minimum and maximum at the endpoints. Thus, to find $\underline{P}$ and $\overline{P}$, it is not necessary to consider all possible values $P(A) \in [\underline{P}(A), \overline{P}(A)]$, it is sufficient to only consider two values: $P(A) = \underline{P}(A)$ and $P(A) = \overline{P}(A)$.

For each of these two values, for another component A', we have two possible options $P(A') = \underline{P}(A')$ and $P(A') = \overline{P}(A')$; thus, in this case, we need to consider $2 \times 2 = 4$ possible combinations of values $P(A)$ and $P(A')$.

In general, when we have k components $A_1, \ldots, A_k$, it is sufficient to consider 2^k possible combinations of values $\underline{P}(A_i)$ and $\overline{P}(A_i)$ corresponding to each of these components. This procedure requires times which grows as 2^k. As we mentioned earlier, when k is large, the needed computation time becomes unrealistically large.

Natural question. The fact that the above algorithm requires unrealistic exponential time raises a natural question: is it because our algorithm is inefficient or is it because the problem itself is difficult?

The problem is NP-hard. In the general case, when no assumption is made about monotonicity, the problem is as follows:

- Let F be a propositional formula with n variables A_i
- for each variable A_i, we know the interval $[\underline{P}(A_i), \overline{P}(A_i)]$ that contains the actual (unknown) $P(A_i)$ that this variable is true;
- we assume that the Boolean variables are independent.

Different values $P(A_i) \in [\underline{P}(A_i), \overline{P}(A_i)]$ lead, in general, to different values of the probability P that F is true (e.g., that the system fails). Our objective is to compute the range $[\underline{P}, \overline{P}]$ of possible values of this probability.

In [8], we have proven that, in general, the problem of computing the desired range $[\underline{P}, \overline{P}]$ is NP-hard. From the practical viewpoint, this means, that (unless P=NP, which most computer scientists believe to be not true), there is no hope to avoid non-feasible exponential time. Since we cannot have a feasible algorithm that is applicable to all possible cases of the general problem, we therefore need to restrict ourselves to practically important cases – and try to design efficient algorithms that work for these cases. This is what we do in this paper.

A practically important case of narrow intervals. When there is enough information, the intervals $[\underline{P}(A), \overline{P}(A)]$ are narrow. If we represent them in the form

$$[\widetilde{P}(A) - \Delta(A), \widetilde{P}(A) + \Delta(A)],$$

with $\widetilde{P}(A) = \dfrac{\underline{P}(A) + \overline{P}(A)}{2}$ and $\Delta(A) = \dfrac{\overline{P}(A) - \underline{P}(A)}{2}$, then values $\Delta(A)$ are small, so we can safely ignore terms which are quadratic or of higher order in terms of $\Delta P(A)$.

Linearization: analysis of the problem. In the case of narrow intervals, the difference $\Delta P(A) \stackrel{\text{def}}{=} P(A) - \widetilde{P}(A)$ is bounded by $\Delta(A)$ and thus, also small: $|\Delta P(A)| \le \Delta(A)$. Hence, we can expand the dependence of the desired system failure probability

$$P = P(P(A), \ldots) = P(\widetilde{P}(A) + \Delta P(A), \ldots)$$

into Taylor series and keep only terms which are linear in $\Delta P(A)$:

$$P \approx \widetilde{P} + \sum_A c_A \cdot \Delta P(A),$$

where $\widetilde{P} \stackrel{\text{def}}{=} P(\widetilde{P}(A), \ldots)$ and $c_A \stackrel{\text{def}}{=} \dfrac{\partial}{\partial P(A)} P(\widetilde{P}(A), \ldots)$.

For those A for which $c_A \geq 0$, the largest value of the sum $\sum\limits_A c_A \cdot \Delta P(A)$ (when $\Delta P(A) \in [-\Delta(A), \Delta(A)]$) is attained when $\Delta P(A)$ attains its largest possible value $\Delta(A)$. Similarly, when $c_A < 0$, the largest possible values of the sum is attained when $\Delta P(A) = -\Delta(A)$. In both cases, the largest possible value of the term $c_A \cdot \Delta P(A)$ is $|c_A| \cdot \Delta(A)$. Thus, the largest possible value of P is equal to $\widetilde{P} + \Delta$, where

$$\Delta \stackrel{\text{def}}{=} \sum_A |c_A| \cdot \Delta(A).$$

Similarly, one can show that the smallest possible value of P is equal to $\widetilde{P} - \Delta$, so the range of possible values of the failure probability P is $[\widetilde{P} - \Delta, \widetilde{P} + \Delta]$.

We already know how to compute $\widetilde{P}$ – e.g., we can use the Monte-Carlo approach. How can we compute Δ?

How to compute Δ: numerical differentiation and its limitations. A natural idea is to compute all the partial derivatives c_A and to use the above formula for Δ. By definition, c_A is the derivative, i.e.,

$$c_A = \lim_{h \to 0} \frac{P(\widetilde{P}(A) + h, \widetilde{P}(B), \widetilde{P}(C), \ldots) - P(\widetilde{P}(A), \widetilde{P}(B), \widetilde{P}(C), \ldots)}{h}.$$

By definition of the limit, this means that to get a good approximation for c_A, we can take a small h and compute

$$c_A = \frac{P(\widetilde{P}(A) + h, \widetilde{P}(B), \widetilde{P}(C), \ldots) - P(\widetilde{P}(A), \widetilde{P}(B), \widetilde{P}(C), \ldots)}{h}.$$

This approach to computing derivatives is called *numerical differentiation.*

The problem with this approach is that each computation of the value

$$P(\widetilde{P}(A) + h, \widetilde{P}(B), \widetilde{P}(C), \ldots)$$

by Monte-Carlo techniques requires a lot of simulations, and we need to repeat these simulations again and again as many times as there are components. For an aircraft, with thousands of components, the resulting increase in computation time is huge. Moreover, since we are interested in the difference $P(\widetilde{P}(A) + h, \ldots) - P(\widetilde{P}(A), \ldots)$ between the two probabilities, we need to compute each of these probabilities with a high accuracy, so that this difference would be visible in comparison with the approximation error $\sim 1/\sqrt{N}$ of the Monte-Carlo estimates. This requires that we further increase the number of iterations N in each Monte-Carlo simulation and thus, even further increase the computation time.

Cauchy deviate techniques: reminder. In order to compute the value $\sum\limits_A |c_A| \cdot \Delta(A)$ faster, one may use a technique based on *Cauchy distributions* (e.g., [13, 19]) , i.e., probability distributions with probability density of the form $\rho(z) = \dfrac{\Delta}{\pi \cdot (z^2 + \Delta^2)}$; the value Δ is called the *scale parameter* of this distribution, or simply a *parameter*, for short.

Cauchy distribution has the following property: if z_A corresponding to different A are independent random variables, and each z_A is distributed according to the Cauchy law with parameter $\Delta(A)$, then their linear combination $z = \sum\limits_A c_A \cdot z_A$ is also distributed according to a Cauchy law, with a scale parameter $\Delta = \sum\limits_A |c_A| \cdot \Delta(A)$.

Therefore, using Cauchy distributed random variables δ_A with parameters $\Delta(A)$, the difference

$$c \stackrel{\text{def}}{=} P(\widetilde{P}(A)+\delta_A, \widetilde{P}(B)+\delta_B, \ldots) - P(\widetilde{P}(A), \widetilde{P}(B), \ldots) = \sum_A c_A \cdot \delta_A$$

is Cauchy distributed with the desired parameter Δ. So, repeating this experiment N_c times, we get N_c values $c^{(1)}, \ldots, c^{(N_c)}$ which are Cauchy distributed with the unknown parameter, and from them we can estimate Δ. The bigger N_c, the better estimates we get.

Comment. To avoid confusion, we should emphasize that the use of Cauchy distributions is a computational technique, *not* an assumption about the actual distribution: indeed, we know that the actual value of $\Delta P(A)$ is bounded by $\Delta(A)$, but for a Cauchy distribution, there is a positive probability that the simulated value is larger than $\Delta(A)$.

Cauchy techniques: towards implementation. In order to implement the above idea, we need to answer the following two questions:

- how to simulate the Cauchy distribution;
- how to estimate the parameter Δ of this distribution from a finite sample.

Simulation can be based on the functional transformation of uniformly distributed sample values: $\delta_A = \Delta(A) \cdot \tan(\pi \cdot (r_A - 0.5))$, where r_A is uniformly distributed on the interval $[0,1]$.

In order to estimate Δ, we can apply the Maximum Likelihood Method

$$\rho(c^{(1)}) \cdot \rho(c^{(2)}) \cdot \ldots \cdot \rho(c^{(N_c)}) \to \max,$$

where $\rho(z)$ is a Cauchy distribution density with the unknown Δ. When we substitute the above-given formula for $\rho(z)$ and equate the derivative of the product with respect to Δ to 0 (since it is a maximum), we get an equation

$$\frac{1}{1+\left(\dfrac{c^{(1)}}{\Delta}\right)^2} + \ldots + \frac{1}{1+\left(\dfrac{c^{(N_c)}}{\Delta}\right)^2} = \frac{N_c}{2}.$$

Its left-hand side is an increasing function that is equal to $0 (< N_c/2)$ for $\Delta = 0$ and $> N_c/2$ for $\Delta = \max\left|c^{(k)}\right|$; therefore the solution to this equation can be found by applying a bisection method to the interval $\left[0, \max\left|c^{(k)}\right|\right]$.

It is important to mention that we assumed that the function P is reasonably linear when the values δ_A are small: $|\delta_A| \le \Delta(A)$. However, the simulated values δ_A may be larger than $\Delta(A)$. When we get such values, we do not use the original function P for them, we use a normalized function that is equal to P within the given intervals, and that is extended linearly for all other values; we will see, in the description of an algorithm, how this is done.

Cauchy deviate technique: main algorithm.

- Apply P to the values $\widetilde{P}(A)$ and compute $\widetilde{P} = P(\widetilde{P}(A), \widetilde{P}(B), \ldots)$.
- For $k = 1, 2, \ldots, N_c$, repeat the following:
  - use the standard random number generator to compute n numbers $r_A^{(k)}$ that are uniformly distributed on the interval $[0,1]$;
  - compute Cauchy distributed values $c_A^{(k)} = \tan(\pi \cdot (r_A^{(k)} - 0.5))$;
  - compute the largest value of $|c_A^{(k)}|$ so that we will be able to normalize the simulated measurement errors and apply P to the values that are within the box of possible values: $K = \max\limits_A |c_A^{(k)}|$;
  - compute the simulated measurement errors $\delta_A^{(k)} := \Delta(A) \cdot c_A^{(k)}/K$;
  - compute the simulated probabilities $P^{(k)}(A) = \widetilde{P}(A) + \delta_A^{(k)}$;
  - estimate $P(P^{(k)}(A), P^{(k)}(B), \ldots)$ and then compute

$$c^{(k)} = K \cdot (P(P^{(k)}(A), P^{(k)}(B), \ldots) - \widetilde{P});$$

- Compute Δ by applying the bisection method to solve the corresponding equation.

Resulting gain and remaining limitation. By using the Monte-Carlo techniques, we make sure that the number of iterations N_c depends only on the accuracy with which we want to find the result and not on the number of components. Thus, when we have a large number of components, this method is faster than numerical differentiation.

The computation time of the new algorithm is smaller, but it is still not very fast. The reason is that the Cauchy method was originally was designed for situations in which we can compute the exact value of $P(P^{(k)}(A), P^{(k)}(B), \ldots)$. In our problem, these values have to be computed by using Monte-Carlo techniques, and computed accurately – and each such computation requires a lot of iterations. Instead of running the maximum likelihood, we can also just estimate Δ by means of the sample interquartile range instead of solving the non-linear equation. But this method will be less accurate.

Final idea to further decrease the needed number of simulations. (see, e.g., Section 5.4 of [19]) For each combination of values δ_A, the corresponding Monte-Carlo simulation produces not the actual probability $P(\widetilde{P}(A) + \delta_A, \widetilde{P}(B) + \delta_B, \ldots)$, but an *approximate* value

$$\widetilde{P}(\widetilde{P}(A) + \delta_A, \widetilde{P}(B) + \delta_B, \ldots) = P(\widetilde{P}(A) + \delta_A, \widetilde{P}(B) + \delta_B, \ldots) + c_n$$

that differs from the desired probability by a random variable c_n which is normally distributed with mean 0 and variance $\sigma^2 = \dfrac{\widetilde{P} \cdot (1 - \widetilde{P})}{N}$. As a result, the difference $c \stackrel{\text{def}}{=} \widetilde{P}(\widetilde{P}(A)+\delta_A, \widetilde{P}(B)+\delta_B, \ldots) - \widetilde{P}$ between the two observed probabilities can be represented as $c = c_c + c_n$, where $c_c \stackrel{\text{def}}{=} P(\widetilde{P}(A) + \delta_A, \widetilde{P}(B) + \delta_B, \ldots) - \widetilde{P}$ is, as we have mentioned, Cauchy distributed with parameter Δ, while

$$c_n = \widetilde{P}(\widetilde{P}(A) + \delta_A, \widetilde{P}(B) + \delta_B, \ldots) - P(\widetilde{P}(A) + \delta_A, \widetilde{P}(B) + \delta_B, \ldots)$$

is normally distributed with mean 0 and known standard deviation σ.

The components c_c and c_n are independent. Thus, for $c = c_c + c_n$, for the characteristic function $\chi(\omega) \stackrel{\text{def}}{=} E[\exp(\mathrm{i} \cdot \omega \cdot c)]$, we have

$$E[\exp(\mathrm{i} \cdot \omega \cdot c)] = E[\exp(\mathrm{i} \cdot \omega \cdot c_c) \cdot \exp(\mathrm{i} \cdot \omega \cdot c_n)] = E[\exp(\mathrm{i} \cdot \omega \cdot c_c)] \cdot E[\exp(\mathrm{i} \cdot \omega \cdot c_n)],$$

i.e., $\chi(\omega) = \chi_c(\omega) \cdot \chi_n(\omega)$, where $\chi_c(\omega)$ and $\chi_n(\omega)$ are characteristic functions of c_c and c_n. For Cauchy distribution and for the normal distribution, the characteristic functions are known: $\chi_c(\omega) = \exp(-|\omega| \cdot \Delta)$ and $\chi_n(\omega) = \exp(-\omega^2 \cdot \sigma^2)$. So, we conclude that $\chi(\omega) = \exp(-|\omega| \cdot \Delta - \omega^2 \cdot \sigma^2)$. Hence, to determine Δ, we can estimate $\chi(\omega)$, compute its negative logarithm, and then compute Δ (see the formula below).

Since the value $\chi(\omega)$ is real, it is sufficient to consider only the real part $\cos(\ldots)$ of the complex exponent $\exp(\mathrm{i} \cdot \ldots)$. Thus, we arrive at the following algorithm:

Algorithm. First, we use a lengthy Monte-Carlo simulation to compute the value $\widetilde{P} = P(\widetilde{P}(A), \widetilde{P}(B), \ldots)$. Then, for $k = 1, 2, \ldots, N$, we repeat the following:

- use a random number generator to compute n numbers $r_A^{(k)}$, that are uniformly distributed on the interval $[0, 1]$;
- compute $\delta_A^{(k)} = \Delta_i \cdot \tan(\pi \cdot (r_A^{(k)} - 0.5))$;
- use Monte-Carlo simulations to find the frequency (probability estimate) $\widetilde{P}(\widetilde{P}(A) + \delta_A^{(k)}, \widetilde{P}(B) + \delta_B^{(k)}, \ldots)$ and then

$$c^{(k)} = \widetilde{P}(\widetilde{P}(A) + \delta_A^{(k)}, \widetilde{P}(B) + \delta_B^{(k)}, \ldots) - \widetilde{P};$$

- for a real number $\omega > 0$, compute $\chi(\omega) = \dfrac{1}{N} \cdot \sum_{k=1}^{N} \cos\left(\omega \cdot c^{(k)}\right)$;

- compute $\Delta = -\dfrac{\ln(\chi(\omega))}{\omega} - \sigma^2 \cdot \dfrac{\omega}{2}$.

Comment. Of course, we also need, as before, to "reduce" the simulated values δ_A to the given bounds $\Delta(A)$.

10. What If We Do Not Assume Independence

Exact methods. If we do not assume independence, then, in principle, we can have different probabilities P of failure even when the failure probabilities $P(A_i)$ of all components are known exactly. For example, if the system consists of two components and it fails if both components fail, i.e., if $F = A_1 \,\&\, A_2$, then possible values of P take an interval

$$[\underline{P}, \overline{P}] = [\max(P(A_1) + P(A_2) - 1, 0), \min(P(A_1), P(A_2)].$$

In general, to describe probabilities of all possible combinations of statements A_i, it is sufficient to describe 2^n probabilities of *atomic* statements $A_1^{\varepsilon_1} \,\&\, \ldots \,\&\, A_n^{\varepsilon_n}$, where $\varepsilon_i \in \{-, +\}$, A^+ means A, and A^- means $\neg A$. These probabilities satisfy the condition that their sum is 1; each given probability $P(A_i)$ and the desired probability P can be described as a sum of some such probabilities, so the problem of finding the range of P becomes the particular case of *linear programming* problems, when we need to find the minimum and maximum of a linear function under linear constraints; see, e.g., [21].

For example, in the above case $n = 2$, we have four non-negative unknowns $P_{++} = P(A_1 \,\&\, A_2)$, $P_{+-} = P(A_1 \,\&\, \neg A_2)$, $P_{-+} = P(\neg A_1 \,\&\, A_2)$, and $P_{--} = P(A_1 \,\&\, A_2)$ that satisfy the constraints $P_{++} + P_{+-} + P_{-+} + P_{--} = 1$, $P_{++} + P_{+-} = P(A_1)$, and $P_{++} + P_{-+} = P(A_2)$. Here, $P = P_{++}$, so, e.g., to find $\underline{P}$, we need to minimize P_{++} under these constraints.

Limitations of the exact methods. This works well if n is small, but for large n, this method requires an unrealistically long time.

Heuristic approximate methods. In principle, we can use technique similar to straightforward interval computations. Indeed, for simple formulas F like $\neg A_1$, $A_1 \vee A_2$, or $A_1 \,\&\, A_2$, we have explicit formulas for the range of the probability $P(F)$. So, to estimate the range of the probability P for an arbitrary formula F, we can do the following:

- we *parse* the expression F, i.e., represent is as a sequence of simple boolean operations – the same sequence that a computer computing F would follow, and
- replace each computation step with corresponding operations with probability ranges.

In this algorithm, at each moment of time, we keep the bounds on the probabilities $P(A_i)$ and on the probabilities $P(F_j)$ of the corresponding intermediate formulas.

The problem with this approach is that at each step, we ignore the dependence between the intermediate results F_j; hence intervals grow too wide. For example, the estimate for $P(A \vee \neg A)$ computed this way is not 1, but an interval containing the correct value 1.

A more accurate algorithm was proposed in [1, 2, 3]. In this algorithm, at each stage, besides the bounds on the values $P(F_j)$ (including the original bounds on $P(A_i)$, or – if available – exact values of $P(A_i)$), we also compute the bounds for the probabilities $P(F_j \,\&\, F_k)$, $P(F_j \,\&\, \neg F_k)$, $P(\neg F_j \,\&\, F_k)$, and $P(\neg F_j \,\&\, \neg F_k)$.

On each computation step, when we add a new intermediate result F_a (e.g., $F_a = F_b \,\&\, F_c$), we add bounds for the probabilities of the new statement F_a and of all possible combinations that include F_a, i.e., on the probabilities $P(F_a \,\&\, F_k)$, $P(F_a \,\&\, \neg F_k)$, $P(\neg F_a \,\&\, F_k)$, and $P(\neg F_a \,\&\, \neg F_k)$. To compute each of these probabilities, we use the known bounds on the probabilities of combinations of F_b and F_k, F_c and F_k, F_b and F_c, and get the desired bounds on the combinations of F_a and F_k by solving the corresponding linear programming problem. In this linear programming problem, we consider, as variables, $2^3 = 8$ probabilities of atomic statements $F_b^{\varepsilon_b} \,\&\, F_c^{\varepsilon_c} \,\&\, F_k^{\varepsilon_k}$.

At the end of the process, we get an interval $\widetilde{P}$. Similarly to interval computations, we can prove, by induction, that every possible value P of the system's failure is contained in this interval, i.e., that the interval $\widetilde{P}$ is an enclosure for the desired range $[\underline{P}, \overline{P}]$: $[\underline{P}, \overline{P}] \subseteq \widetilde{P}$.

This algorithm requires more computation time that the above straightforward algorithm – since for s intermediate steps, we now need s^2 estimations instead of s – but as a result, we get more accurate accurate, with smaller "excess width" for the resulting enclosure. For example, the probability $P(A \vee \neg A)$ is estimated as 1.

To get an even better accuracy, instead of probabilities of all possible combinations of two intermediate results, we can compute, on each step, probabilities of all possible combinations of three such results: F_i, F_j, and F_k. In this case, computation time grows as s^3 but the resulting enclosure is even more accurate. Similarly, we can have combinations of fours, fives, etc.

11. Conclusion

In this paper we considered the problem of estimating the probability of failure P of a complex system such as an aircraft, assuming we only know upper and lower bounds of probabilities of elementary events such as component failures. The assumptions in this paper is that failures of different components are independent events, and that there is enough information to ensure narrow probability intervals. The problem of finding the resulting range $[\underline{P}, \overline{P}]$ of possible values of P is computationally difficult (NP-hard). In this paper, we describe several efficient algorithms for computing the range. In particular, for the practically important case of narrow intervals $[\underline{P}(A), \overline{P}(A)]$, we describe an efficient method that uses Cauchy deviates to estimate the desired range $[\underline{P}, \overline{P}]$. Future works concern the estimation of intervals $[\underline{P}(A), \overline{P}(A)]$ from the imprecise knowledge of failure rates. Moreover, it is interesting to study what can be done in practice when the independence assumption on component failures no longer holds.

12. Acknowledgment

V. Kreinovich was supported by the National Science Foundation (NSF) grants HRD-0734825 and DUE-0926721 and by Grant 1 T36 GM078000-01 from the National Institutes of Health (NIH). C. Jacob was supported by a grant from @MOST Prototype, a joint project of Airbus, Institut de Recherche en Informatique deToulouse (IRIT), LAAS-CNRS, ONERA, and ISAE.

We are thankful to the Airbus colleagues for providing us examples of the practical problems, general guidance about which assumptions are reasonable and which are not for aircraft maintenance problems, and for testing our techniques.

We are also thankful to all the participants of the International Workshop on Soft Computing Applications and Knowledge Discovery SCAKD'2011 (Moscow, Russia, June 25, 2011) and the 10th Mexican International Conference on Artificial Intelligence MICAI'2011 (Puebla, Mexico, November 26 – December 4, 2011) for valuable discussions.

References

[1] Ceberio, M., Kreinovich, V., Chopra, S., Longpré, L., Nguyen, H.T., Ludäscher, B., Baral, C. Interval-type and affine arithmetic-type techniques for handling uncertainty in expert systems, *Journal of Computational and Applied Mathematics*, V.199, N.2, 2007, pp.403-410.

[2] Chopra, S., *Affine arithmetic-type techniques for handling uncertainty in expert systems*, Master's thesis, Department of Computer Science, University of Texas at El Paso, 2005.

[3] Chopra, S. Affine arithmetic-type techniques for handling uncertainty in expert systems, International *Journal of Intelligent Technologies and Applied Statistics*, V.1, N.1, 2008, pp.59-110.

[4] Dutuit, Y., Rauzy, A. Approximate estimation of system reliability via fault trees, *Reliability Engineering and System Safety*, V.87, N.2, 2005, pp. 163-172.

[5] Flage, R., et al. Handling of epistemic uncertainties in fault tree analysis: a comparison between probabilistic, possibilistic, and hybrid approaches, In: Bris, S., Guedes Sares, C., Martorell, S. (eds.), *Reliability, Risk and Safety: Theory and Applications, Proc. European Safety and Reliability Conf. ESREL'2009, Prague, September 7–10, 2009*, 2010.

[6] Guth, M. A. A probability foundation for vagueness and imprecision in fault tree analysis, *IEEE Transations on Reliability*, V.40, N.5, 1991, pp. 563-570.
[7] Interval computations website http://www.cs.utep.edu/interval-comp
[8] Jacob, C., Dubois, D., Cardoso, J., Ceberio, M., Kreinovich, K. Estimating probability of failure of a complex system based on partial information about subsystems and components, with potential applications to aircraft maintenance, *Proceedings of the International Workshop on Soft Computing Applications and Knowledge Discovery SCAKD'2011*, Moscow, Russia, June 25, 2011.
[9] Jacob, C., et al. Uncertainty handling in quantitative BDD-based fault-tree analysis by interval computation. *Proceedings of the 5th International Conference on Scalable Uncertainty Management SUM'2011, Dayton, OH, USA, October 10-13, 2011*, Springer Lecture Notes in Computer Science, V.6929, 2011.
[10] Jaksurat, P., Freudenthal, E., Ceberio, M., Kreinovich, V. Probabilistic approach to trust: ideas, algorithms, and simulations, *Proceedings of the Fifth International Conference on Intelligent Technologies InTech'04*, Houston, Texas, December 2–4, 2004.
[11] Jaulin, L., Kieffer, M., Didrit, O., Walter, E. *Applied Interval Analysis: With Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer, London, 2001.
[12] Klir, G., Yuan, B., *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.
[13] Kreinovich, V., Ferson, S. A new Cauchy-based black-box technique for uncertainty in risk analysis, *Reliability Engineering and Systems Safety*, V.85, N.1-3, P.267-279.
[14] Kreinovich, V., Jacob, C., Dubois, D., Cardoso, J., Ceberio, M., Batyrshin, I. Estimating Probability of Failure of a Complex System Based on Inexact Information about Subsystems and Components, with Potential Applications to Aircraft Maintenance, In: Batyrshin, I., Sidorov, G. (eds.), *Proceedings of the 10th Mexican International Conference on Artificial Intelligence MICAI'2011*, Puebla, Mexico, November 26 - December 4, 2011, Springer Lecture Notes in Artificial Intelligence, V.7905, pp.70-81.
[15] Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P. *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1997.
[16] Moore, R.E., Kearfott, R.B., Cloud, M.J. *Introduction to Interval Analysis*, SIAM Press, Philadelphia, Pennsylvania, 2009.
[17] Nguyen, H.T., Walker, E.A. *A First Course in Fuzzy Logic*, Chapman & Hall/CRC, Boca Raton, Florida, 2006.
[18] Papadimitriou, C. *Computational Complexity*, Addison Welsey, Reading, Massachusetts, 1994.
[19] Trejo, R., Kreinovich, V. Error estimations for indirect measurements: randomized vs. deterministic algorithms for 'black-box' programs, In: Rajasekaran, S., Pardalos, P., Reif, J., Rolim, J. (eds.), (eds.), *Handbook on Randomized Computing*, Kluwer, 2001, pp.673-729.
[20] Troffaes, M., Coolen, F. On the use of the imprecise Dirichlet model with fault trees, *Proceedings of the Mathematical Methods in Reliability Conference*, Glasgow, July 2007.
[21] Walley, P. *Statistical reasoning with imprecise probabilities*, Chapman & Hall, New York, 1991.

Vladik Kreinovich - received his M.Sc. in Mathematics and Computer Science from St. Petersburg University, Russia, in 1974, and Ph.D. from the Institute of Mathematics, Soviet Academy of Sciences, Novosibirsk, in 1979. In 1975-80, he worked with the Soviet Academy of Sciences, in particular, in 1978-80, with the Special Astrophysical Observatory (representation and processing of uncertainty in radioastronomy). In 1982-89, he worked on error estimation and intelligent information processing for the National Institute for Electrical Measuring Instruments, Russia. In 1989, he was a Visiting Scholar at Stanford University. Since 1990, he is with the Department of Computer Science, University of Texas at El Paso. Also, served as an invited professor in Paris (University of Paris VI), Hong Kong, St. Petersburg, Russia, and Brazil.
Main interests: representation and processing of uncertainty, especially interval computations and intelligent control. Published 3 books, 6 edited books, and more than 800 papers. Member of the editorial board of the international journal *Reliable Computing* (formerly, *Interval Computations*), and several other journals. Co-maintainer of the international website on interval computations http://www.cs.utep.edu/interval-comp

Christelle Jacob - A graduate student from Université Paul-Sabatier (France) in 2006. She obtained a Master degree in Applied Mathematics from Université Paul-Sabatier (France) in 2008, and a Specialized Master in Embedded Systems from Institut Supérieur de l'Aéronautique et de l'Espace (ISAE, France) in 2009. Presently, she is preparing a Ph.D degree on uncertainty management for preventive maintenance of aircrafts in the project *@MOST* of Airbus company.

Didier Dubois is a Research Advisor at IRIT, the Computer Science Department of Paul Sabatier University in Toulouse, France and belongs to the French National Centre for Scientific Resarch (CNRS). He holds a Doctorate in Engineering from ENSAE, Toulouse (1977), a Doctorat d'Etat from Grenoble University (1983) and an Honorary Doctorate from the Facult Polytechnique de Mons, Belgium (1997). He is the coauthor, with Henri Prade, of two books on fuzzy sets and possibility theory, and 15 edited volumes on uncertain reasoning and fuzzy sets. Also with Henri Prade, he coordinated the HANDBOOK of FUZZY SETS series published by Kluwer (7 volumes, 1998-2000) including the book Fundamentals of Fuzzy Sets. He has contributed about 200 technical journal papers on uncertainty theories and applications. He is Editor- in -Chief of the journal Fuzzy Sets and Systems, an Advisory Editor of the IEEE Transactions on Fuzzy Systems, and a member of the Editorial Board of several technical journals, such as the International Journals on Approximate Reasoning, General Systems, Applied Logic, and Information Sciences among others. He is a former president of the International Fuzzy Systems Association (1995-1997). He received the 2002 Pioneer Award of the IEEE Neural Network Society, and the 2005 IEEE TFS Outstanding Paper Award. His topics of interest range from Artificial Intelligence to Operations Research and Decision Sciences, with emphasis on the modelling, representation and processing of imprecise and uncertain information in reasoning and problem-solving tasks.

Janette Cardoso - graduated from the Universidade Federal de Santa Catarina, Brazil. She received her PhD degree in Computer Science from the University of Toulouse, France, in 1990 and her *Habilitation* degree in 2007. Presently, she is Professor at ISAE - Institut Supérieur de l'Aéronautique et de l'Espace, Toulouse, France. Her research interests are focused on Possibility theory, Petri nets and formal methods for embedded systems.

Martine Ceberio is an assistant professor in Computer Science at the University of Texas at El Paso (UTEP). She received her PhD in Computer Science in 2003 from the University of Nantes, France and joined UTEP in fall 2003. Her research is in numerical constraint solving, global optimization, multi-criteria decision making, and their applications. She has been engaged in activities related to the participation of women in computing since 2008. She has implemented projects with an all-girls high school of El Paso and she coordinates the local affiliate of a national competition of the National Center for Women in Information and Technology (NCWIT) that recognizes high-school girls with interest and experience in computer science.

Ildar Batyrshin - received Diploma of Moscow Physical-Technical Institute in 1975; Ph.D. from Moscow Power Engineering Institute in 1983; Dr.Sci. (Habilitation) from Higher Attestation Committee of Russian Federation in 1996. Since 1975 he has been with the Department of Informatics and Applied Mathematics of Kazan State Technological University, Russia, (Department head in 1997-2003). Since 1999, he has been also with the Institute of Problems of Informatics of Academy of Sciences of the Republic of Tatarstan, Russia as a leading researcher. He is now with the Research Program of Applied Mathematics and Computations of Mexican Petroleum Institute as a leading researcher and head of the project. He is also a visiting professor of the Center of Computing Research of the National Polytechnic Institute of Mexico. He is a Past President of Russian Association for Fuzzy Systems and Soft Computing, member of the International Fuzzy Systems Association (IFSA) Council, member of the Council of the Mexican Society of Artificial Intelligence, IEEE member, Honorary Professor of Budapest Tech and Honorary Researcher of the Republic of Tatarstan, Russia. He is a coauthor of 5 books. His areas of research activity are Fuzzy Logic and Fuzzy Reasoning, Hardware Implementation of Fuzzy Systems, Computational Intelligence, Expert Systems, Decision Making, Cluster Analysis, Time Series Data Mining and Data Visualization.