Estimating Probability of Failure of a Complex System based on Partial Information about Subsystems and Components, with Potential Applications to Aircraft Maintenance

Christelle Jacob[1], Didier Dubois[2], Janette Cardoso[1], Martine Ceberio[3], and Vladik Kreinovich[3]

[1]Institut Supérieur de l'Aéronautique et de l'Espace (ISAE)
Toulouse, France, jacob@isae.fr, cardoso@isae.fr
[2]Institut de Recherche en Informatique de Toulouse (IRIT)
Toulouse, France, dubois@irit.fr
[3]University of Texas at El Paso
El Paso, Texas, USA, mceberio@utep.edu, vladik@utep.edu

1. Formulation of the Problem

- Ideally, there should be no risk.
- In practice, it is not possible to completely eliminate the risk.
- It is thus necessary to estimate the probability of failure for complex systems.
- *Example:* aircraft design and maintenance.
- In *design*, we need to make sure that this probability of failure does not exceed the allowed probability p_0.
- On the *maintenance* stage, we need to decide whether a maintenance is needed:
  - if the probability of failure exceeds p_0,
  - this means that we need to perform maintenance to decrease this probability to the desired level $\leq p_0$.

2. Information Available for Estimating the System's Probability of Failure

- 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,
  - how reliable are these components and subsystems,
  - are the component failures independent events or they are caused by a common cause.
- Complex systems are usually built with redundancy.
- For each possible set of failed components, we usually know whether this set will lead to a system failure.
- For each component A, there is a probability $P(A)$ of its failure.
- For highly reliable components, there are few failures, so we only have a (confidence) interval $\mathbf{P}(A) = [\underline{P}(A), \overline{P}(A)]$.

3. Simplest Case: Component Failures Independent, Failure Probabilities $P(A)$ Known

- *Monte-Carlo simulations:* on each of N simulations,
  - for each component A, we simulate its failure with probability $P(A)$;
  - then we check whether, as a result of all these failures, the system fails or not.
- We take the proportion $p = f/N$ of failures as an estimate for the failure probability P.
- With prob. 99.9%, P is within the three-sigma interval $[p-3\sigma, p+3\sigma]$, where $\sigma = \sqrt{p\cdot(1-p)/N} \le 0.5/\sqrt{N}$.
- So, to get P with accuracy 10%, we need $N \ge 225$.
- *Important:* this number of iterations N is the same no matter how many components we have.

4. When Components are Very Reliable

- In this case, with high probability, in all simulations, all the components will be simulated as working properly.
- Hence, the above Monte-Carlo (MC) technique requires a large number of simulations.
- Idea: instead of the original values $P(A)$, we use re-scaled (larger) values $\lambda \cdot P(A)$ for some $\lambda \gg 1$.
- When $\lambda \cdot P(A)$ are small, the probability $P(\lambda)$ depends on λ as $P(\lambda) \approx \lambda^k \cdot P$ for some integer $k > 0$. So, we:
  - repeat MC procedure for two values $\lambda_1 \neq \lambda_2$, get $P(\lambda_1)$ and $P(\lambda_2)$,
  - then find k and P from equations $P(\lambda_1) \approx \lambda_1^k \cdot P$ and $P(\lambda_2) \approx \lambda_2^k \cdot P$.
- Here, $P(\lambda_1)/P(\lambda_2) \approx (\lambda_1/\lambda_2)^k$, so $k \approx \dfrac{\ln(P(\lambda_1)/P(\lambda_2))}{\ln(\lambda_1/\lambda_2)}$, and $P \approx P(\lambda_1)/\lambda_1^k$.

5. Interval Uncertainty

- In many cases, we only know the intervals $\mathbf{P}(A) = [\underline{P}(A), \overline{P}(A)]$ that contain the probabilities $P(A)$.
- In general different values $P(A) \in \mathbf{P}(A)$ lead to different probabilities of failure $P = f(P(A), P(B), \ldots)$.
- So, we want to know the range of possible values of 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 $\mathbf{P}$ is a particular case of interval computations:
  - 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 $\mathbf{x}_i$.

6. Fuzzy Uncertainty

- Sometimes, we only know a fuzzy number $\mathcal{P}(A)$ that describes the probability $P(A)$.
- A natural way to define the membership function $\mu(P)$ for P is Zadeh's extension principle:

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

- It is known that this formula can be reduced to interval computations for α-cuts $\mathcal{X}(\alpha) \stackrel{\text{def}}{=} \{x : \mu(x) \geq \alpha\}$:

$$\mathcal{P}(\alpha) = f(\mathcal{P}(A, \alpha), \mathcal{P}(B, \alpha), \ldots); \text{ so:}$$

  - once we know how to solve our problem under interval uncertainty,
  - we can also solve it under fuzzy uncertainty, by taking, e.g., $\alpha = 0, 0.1, \ldots, 0.9, 1.0$.
- In view of this reduction, in the following text, we will only consider the case of interval uncertainty.

7. Interval Uncertainty: Monotonicity Case

- Let us assume that if a system fails, it will still fail if we add components to the list of failed ones.
- In this case, the smaller the probability of failure $P(A)$ for each A, the smaller the failure probability P.
- Thus, to compute the smallest possible value $\underline{P}$ of the failure probability, it is sufficient to consider $\underline{P}(A)$.
- So, to compute the largest possible value $\overline{P}$ of the failure probability, it is sufficient to consider $\overline{P}(A)$. Hence:
  - 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 $\overline{P}(A)$.
- Each of these two computations can be performed by using efficient Monte-Carlo techniques.

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

- To increase reliability, systems include duplicate subsystems that take charge if the main subsystem fails.
- E.g., if $X^2=\sum_{i=1}^{n}\frac{(x_i-m_i)^2}{\sigma_i^2}$ exceeds a χ^2-threshold, this means that some sensors are malfunctioning.
- One malfunctioning sensor may not increase X^2 too high.
- So, its malfunctioning will not be detected, and the system will fail.
- If all n sensors fail, the system will detect the malfunctioning, and the duplicate subsystem will be activated.
- This is exactly the case of non-monotonicity.

9. Case of Few Non-Monotonic Components

- For all monotonic B, we take $P(B) = \overline{P}(B)$ when computing $\overline{P}$, and $P(B) = \underline{P}(B)$ when computing $\underline{P}$.
- For non-monotonic components A, we get:

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

- A linear function of $P(A)$ attains it minimum and maximum at the endpoints.
- Thus, to find $\underline{P}$ and $\overline{P}$, it is sufficient to only consider two values: $P(A) = \underline{P}(A)$ and $P(A) = \overline{P}(A)$.
- For k non-monotonic components $A_1, \ldots, A_k$, it is sufficient to take 2^k combinations of $\underline{P}(A_i)$ and $\overline{P}(A_i)$.
- When k is small, this is doable – and, as our preliminary experiments show, works very well.
- When k is large, the needed computation time becomes unrealistically large.

10. The Problem is NP-Hard, and Even Checking Whether a Component Is Monotonic Is NP-hard

- *Reminder:* the above algorithm requires unrealistic exponential time.
- *Natural question:* is it because our algorithm is inefficient or is it because the problem itself is difficult?
- *Claim:* the problem if NP-hard.
- *Proof:* reduction from SAT: for $[\underline{P}(A), \overline{P}(A)] = [0, 1]$:
  - for a satisfiable propositional formula F, $\overline{P} = 1$;
  - for a non-satisfiable F, we have $\overline{P} = 0$.
- *Auxiliary result:* checking monotonicity is NP-hard.
- *Proof:* for each $F(A_1, \ldots, A_n)$, let $F' \stackrel{\text{def}}{=} F(A_1, \ldots, A_n) \,\&\, \neg A_0$.
- Then, F' is monotonic w.r.t. $A_0 \Leftrightarrow F$ is satisfiable.

11. Case of Narrow Intervals: Cauchy Deviate Method

- In many practical situations, intervals are narrow.
- In this case, we can use an efficient Cauchy deviates method to find the range of the resulting probability P.
- General problem:
  - we want to find the value $y = f(x_1, \ldots, x_n)$;
  - we compute $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ based on the estimates $\widetilde{x}_i \approx x_i$.
- Since $\widetilde{x}_i \neq x_i$, the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is different from the actual value $y = f(x_1, \ldots, x_n)$.
- *Problem:* estimate the uncertainty $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$.

12. Propagation of Probabilistic Uncertainty

- *Fact:* often, we know the probabilities of different values of Δx_i.
- *Example:* Δx_i are independent normally distributed with mean 0 and known st. dev. σ_i.
- *Monte-Carlo approach:*
  - For $k = 1, \ldots, N$ times, we:
    - simulate the values $\Delta x_i^{(k)}$ according to the known probability distributions for x_i;
    - find $x_i^{(k)} = \widetilde{x}_i - \Delta x_i^{(k)}$;
    - find $y^{(k)} = f(x_1^{(k)}, \ldots, x_n^{(k)})$;
    - estimate $\Delta y^{(k)} = y^{(k)} - \widetilde{y}$.
  - Based on the sample $\Delta y^{(1)}, \ldots, \Delta y^{(N)}$, we estimate the statistical characteristics of Δy.

13. Propagation of Interval Uncertainty

- *In practice:* we often do not know the probabilities.
- *What we know:* the upper bounds Δ_i on the approximation errors Δx_i: $|\Delta x_i| \le \Delta_i$.
- *Enter intervals:* once we know $\widetilde{x}_i$, we conclude that the actual (unknown) x_i is in the interval

$$\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i].$$

- *Problem:* find the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of possible values of y when $x_i \in \mathbf{x}_i$:

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=}$$

$$\{f(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

- *Fact:* this *interval computation* problem is, in general, NP-hard.

14. Linearization

- In many practical situations, the errors Δx_i are small, so we can ignore quadratic terms:

$$\Delta y = \widetilde{y} - y = f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(x_1, \ldots, x_n) =$$
$$f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) \approx$$
$$c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n,$$

  where $c_i \stackrel{\text{def}}{=} \dfrac{\partial f}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.

- For a linear function, the largest Δy is obtained when each term $c_i \cdot \Delta x_i$ is the largest:

$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n.$$

- Due to the linearization assumption, we can estimate each partial derivative c_i as

$$c_i \approx \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h_i}.$$

15. Linearization: Algorithm

To compute the range $\mathbf{y}$ of y, we do the following.

- First, we apply the algorithm f to the original estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$, resulting in the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- Second, for all i from 1 to n,
  - we compute $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$ for some small h_i and then
  - we compute

$$c_i = \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h_i}.$$

- Finally, we compute $\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n$ and the desired range $\mathbf{y} = [\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.
- *Problem:* we need $n+1$ calls to f, and this is often too long.

16. Cauchy Deviate Method: Idea

- For large n, we can further reduce the number of calls to f if we Cauchy distributions, w/pdf

$$\rho(z) = \frac{\Delta}{\pi \cdot (z^2 + \Delta^2)}.$$

- Known property of Cauchy transforms:
  - if $z_1, \ldots, z_n$ are independent Cauchy random variables w/parameters $\Delta_1, \ldots, \Delta_n$,
  - then $z = c_1 \cdot z_1 + \ldots + c_n \cdot z_n$ is also Cauchy distributed, w/parameter

$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n.$$

- This is exactly what we need to estimate interval uncertainty!

17. Cauchy Deviate Method: Towards Implementation

- To implement the Cauchy idea, we must answer the following questions:
  - how to simulate the Cauchy distribution; and
  - 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:

$$\delta_i = \Delta_i \cdot \tan(\pi \cdot (r_i - 0.5)), \text{ where } r_i \sim U([0,1]).$$

- To estimate Δ, we can apply the Maximum Likelihood Method $\rho(\delta^{(1)}) \cdot \rho(\delta^{(2)}) \cdot \ldots \cdot \rho(\delta^{(N)}) \to \max$, i.e., solve

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

18. Cauchy Deviates Method: Algorithm

- Apply f to $\widetilde{x}_i$; we get $\widetilde{y} := f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- For $k = 1, 2, \ldots, N$, repeat the following:
  - use the standard RNG to draw $r_i^{(k)} \sim U([0,1])$, $i = 1, 2, \ldots, n$;
  - compute Cauchy distributed values $c_i^{(k)} := \tan(\pi \cdot (r_i^{(k)} - 0.5))$;
  - compute $K := \max_i |c_i^{(k)}|$ and normalized errors $\delta_i^{(k)} := \Delta_i \cdot c_i^{(k)} / K$;
  - compute the simulated "actual values" $x_i^{(k)} := \widetilde{x}_i - \delta_i^{(k)}$;
  - compute simulated errors of indirect measurement: $\delta^{(k)} := K \cdot \left(\widetilde{y} - f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)\right)$;
- Compute Δ by applying the bisection method to solve the Maximum Likelihood equation.

19. Important Comment

- To avoid confusion, we should emphasize that:
  - in contrast to the Monte-Carlo solution for the probabilistic case,
  - the use of Cauchy distribution in the interval case is a computational trick,
  - it is *not* a truthful simulation of the actual measurement error Δx_i.
- Indeed:
  - we know that the actual value of Δx_i is always inside the interval $[-\Delta_i, \Delta_i]$, but
  - a Cauchy distributed random attains values outside this interval as well.

20. What If We Do Not Assume Independence: Exact Methods

- It is sufficient to describe 2^n prob. of *atomic* statements $A_1^{\varepsilon_1} \,\&\, \ldots \,\&\, A_n^{\varepsilon_n}$, w/$\varepsilon_i \in \{-,+\}$, $A^+ \stackrel{\text{def}}{=} A$, $A^- \stackrel{\text{def}}{=} \neg A$.
- These probabilities add up to 1.
- Each value $P(A_i)$ and the desired value 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* (LP) problems:
  - find min (max) of a linear function
  - under linear constraints.
- For $n = 2$, we have $P_{++} = P(A_1 \,\&\, A_2)$, $P_{+-} = P(A_1 \,\&\, \neg A_2)$, $P_{-+} = P(\neg A_1 \,\&\, A_2)$, $P_{--} = P(A_1 \,\&\, A_2)$.
- They satisfy constraints $P_{++} + P_{+-} + P_{-+} + P_{--} = 1$, $P_{++} + P_{+-} = P(A_1)$, and $P_{++} + P_{-+} = P(A_2)$.

21. Heuristic Approximate Methods

- For large n, this method requires an unrealistically long time $\sim 2^n$.
- In principle, we can use technique similar to straightforward interval computations:
  - we *parse* the expression F, i.e., represent it as a sequence of simple boolean operations, and
  - replace each computation step with corresponding operations with probability ranges.
- At each stage, we keep the bounds on the probabilities $P(F_j)$ of the corresponding intermediate formulas.
- The problem: 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 an interval containing the correct value 1.

22. Heuristic Approximate Methods (cont-d)

- A more accurate algorithm: at each stage,
  - besides the bounds on the values $P(F_j)$,
  - we also compute the bounds for the probabilities $P(F_j \,\&\, F_k)$, $P(F_j \,\&\, \neg F_k)$, $P(\neg F_j \,\&\, F_k)$, $P(\neg F_j \,\&\, \neg F_k)$.
- For each new intermediate result F_a (e.g., $F_a = F_b \,\&\, F_c$), we add bounds for all combinations containing F_a:

$$P(F_a \,\&\, F_k), P(F_a \,\&\, \neg F_k), P(\neg F_a \,\&\, F_k), P(\neg F_a \,\&\, \neg F_k).$$

- 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,
  - get the desired bounds on the combinations of F_a and F_k by solving the corresponding LP problem.
- In this linear programming problem, we consider, as variables, $2^3 = 8$ atomic probabilities $F_b^{\varepsilon_b} \,\&\, F_c^{\varepsilon_c} \,\&\, F_k^{\varepsilon_k}$.

23. Heuristic Approximate Methods (cont-d)

- At the end, we get an interval $\widetilde{P} \supseteq [\underline{P}, \overline{P}]$.
- For s intermediate steps, we now need s^2 estimations instead of s.
- 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, we can compute probabilities of all possible combinations of three results:

$$F_i, \quad F_j, \text{ and } F_k.$$

- 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.

24. Acknowledgments

C. Jacob was supported by a grant from @MOST Prototype, a join project of

- Airbus,
- Institut de Recherche en Informatique de Toulouse (IRIT),
- Laboratoire d'Analyse et d'Architecture des Systémes (LAAS),
- ONERA, and
- Institut Supérieur de l'Aéronautique et de l'Espace (ISAE).

V. Kreinovich was supported in part:

- by the National Science Foundation grants HRD-0734825 and DUE-0926721, and
- by Grant 1 T36 GM078000-01 from the National Institutes of Health.