How to Generate Worst-Case Scenarios When Testing Already Deployed Systems Against Unexpected Situations

Francisco Zapata[1], Ricardo Pineda[1], and Martine Ceberio[2]
[1]Research Institute for Manufacturing & Engineering Systems RIMES
[2]Department of Computer Science
University of Texas at El Paso, 500 West University Ave., El Paso, TX 79968, USA
fazapatagonzalez@utep.edu, rlpineda@utep.edu, mceberio@utep.edu

***Abstract*—Before a system of systems is deployed, it is tested – but it is tested against known operational mission, under several known operational scenarios. Once the system is deployed, new possible unexpected and/or uncertain operational scenarios emerge. It is desirable to develop methodologies to test the system against such scenarios. A possible methodology to test the system would be to generate the worst case scenario that we can think of – to understand, in principle, the behavior of the system. So, we face a question of generating such worst-case scenarios. In this paper, we provide some guidance on how to generate such worst-case scenarios.**

I. Formulation of the Problem

Traditional Systems Engineering approach. In the traditional Systems Engineering (SE) approach, a system of interest (SOI) is developed by eliciting requirements from the stakeholders. These requirements are analyzed and synthesized to build an architectural design that will drive the system development. Through an iterative process the system is constantly refined via elicitation and update of requirements, design, development, and testing until a final product is obtained [2]. In this approach, the development of the SOI is limited to the requirements specified by the stakeholders, and emergent behavior is not welcomed [8].

Systems of Systems. Since the 1990s, advances in Information and Communication Technologies (ICT) have enabled greater capabilities to exchange information between systems in near real-time or in real-time. The integration of these independently developed systems required communication interface standards, information models, and inter-operatibility standards. This integration need has given birth to a new kind of meta-systems called *Systems of Systems* (SoS) [8]. An SoS is a system of interest composed of a collection of large-scale, heterogenous systems that inter-operate to achieve a greater common objective. An SoS is characterized by the following attributes [8]:

- operational independence of the constituent systems,
- managerial independence of the constituent systems,
- SoS evolutionary development,
- SoS incremental functionality (knowledge domains),
- geographical distribution.

While for constituent systems, new behavior is still not welcomed, for the meta-system, some new emerging behavior may be welcomed.

Formulation of the problem. Before a system of systems is deployed, Integration, Verification, Validation, Test and Evaluation (IVVT&E) methodologies are applied to test the system against well defined operational mission under several known operational scenarios and Modes of Operation. Once the system is deployed, new possible scenarios emerge; see, e.g., [1], [3], [10] and references therein. It is desirable to develop methodologies to test a system against such emergent scenarios, to understand its behavior.

General examples:

- An Unmanned Aircraft System (UAS) encounters new scenarios that were not predicted when this system was deployed.
- A health care monitoring system encounters a new illness that was not known before system testing and deployment.

Specific example. In this paper, we start the analysis by considering the simplest example: an automatic system that helps prevent a car from getting too close to the walls of a freeway (or a tunnel).

At first glance, all we need for this is a sensoring system that measures a distance x from a car to the nearby wall – e.g., by emitting a sound wave and measuring the distance by the time it took the sound to bounce back. There are usually several distance sensors, and the system is set up to work well in the expected situation of a freeway or a tunnel.

The problem starts when we encounter a new unexpected situation, e.g., a hole in the wall or a rock that fell near a wall. In the case of a hole, some sensors measure the distance to a wall, while other sensors measure the distance to a next faraway wall which is located very far from the road. As a result, the existing algorithms may under-estimate the distance to the obstacle. So, even when the car is dangerously close to the wall, the system may operate under the false impression of safety.

Need for generating worst-case scenarios. Once the system designers realize that novel situations are possible, they can come up with ways to improve the system's performance on such non-standard situations. A usual way of testing a system is to test it on worst-case scenarios; see, e.g., [1], [3], [10]. So, we face a question of generating such worst-case scenarios.

What we do in this paper. In this paper, on the example of the above car problem, we explore the ways of generating worst-case scenarios to validate system behavior under unexpected scenarios. We start with a simplified (crisp) case when we assume to know the exact range of each variable and when we do not have any information about which combinations of values of these variables are more probable and which are less probable. Then, we show how we can take into account (inevitably imprecise, uncertain) information about which values (and which situations) are more probable and which are less probable.

While our preliminary results are encouraging, the main objective of this paper is not so much to present new results, but rather to introduce an important class of problems and to solicit help of the fuzzy research community in solving these problems.

II. Case Study: Precise Formulation of the Problem (Crisp Case)

How the distance-measuring system is set up now: a simplified description. The distance-measuring system usually involve several sensors, to make sure that the system remains operational when one of the sensors fails.

Each of the sensors produces a measurement results, so we need to estimate the actual distance d based on these measurement results $x_1, \ldots, x_n$. Because of the measurement noise, for each distance d, we get different values $x_i \approx d$ with different probabilities. In many cases, the measurement error is normally distributed, with a standard deviation σ. In other words, for each result x_i, we have a probability distribution with the probability density

$$\rho_{d,i}(x_i) = \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(-\frac{(x_i-d)^2}{2\sigma^2}\right).$$

Measurement errors corresponding to different measurements are usually independent; so the probability density $\rho_d(x)$ for the vector $x = (x_1, \ldots, x_n)$ of measurement results can be described as a product of the probability density functions corresponding to different sensors:

$$\rho_d(x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\cdot\sigma}\cdot\exp\left(-\frac{(x_i-d)^2}{2\sigma^2}\right).$$

As a desired estimate d for the distance, it is reasonable to select the most probable value d, i.e., the value d for which, for the actual measurement results $x_1, \ldots, x_n$, the probability $\rho_d(x_1, \ldots, x_n)$ is the largest possible.

Differentiating this expression with respect to d and equating the derivative to 0, we conclude that the most probable value d is equal to the arithmetic average

$$\overline{x} = \frac{x_1 + \ldots + x_n}{n}$$

of the measurement results $x_1, \ldots, x_n$.

Criterion for selecting a worst-case scenario. The above analysis shows that a reasonable way to set up a system for measuring distance is to estimate this distance based on the arithmetic average of measured values $x_1, \ldots, x_n$.

This average works well in standard situations, but in non-standard situations, we should raise an alert when at least one of the measured distances x_i becomes dangerously small, i.e., equivalently, when the smallest of the three distances is dangerously small. In other words, ideally, we should make decisions based not on the average but on the minimum $m \stackrel{\text{def}}{=} \min(x_1, \ldots, x_n)$.

When the minimum m is close to the average $\overline{x}$, the situation is not so bad. The worst-case scenario is when there is a drastic difference between $\overline{x}$ and m. In other words, the worst-case scenario is, in this case, a scenario for which the difference

$$\overline{x} - m = \frac{x_1 + \ldots + x_n}{n} - \min(x_1, \ldots, x_n) \qquad (1)$$

attains the largest possible value.

Crisp case. First, we consider the crisp case, when each distance x_i can take arbitrary value from the interval $[0, D]$, for some constant D. In this case, we need to maximize the objective function (1) under the constraints that $0 \le x_i \le D$.

III. Case Study: Algorithmic Analysis

In general, exact optimization is difficult. In general, the problem of exactly maximizing a given function is computationally difficult (NP-hard); see, e.g., [5], [7]. Crudely speaking, this means that unless P = NP (which most computer scientists believe to be false), we cannot have an efficient (feasible) algorithm that always provides an exact solution to the optimization problem.

Let us select an approximate algorithm. Since *exact* optimization is difficult, we need to use *approximate* optimization algorithms. Most known optimization algorithms – such as gradient descent and its versions – use derivatives of the objective function. In our case, the objective function is not differentiable when two of the values x_i coincide. Indeed, even for two values x_1 and x_2, the function $\min(x_1, x_2)$ is equal to x_1 when $x_1 \le x_2$ and to x_2 when $x_1 \ge x_2$; thus, this function is not differentiable at the point $x_1 = x_2$.

Since our objective function is not differentiable, we need to use optimization algorithms which do not require derivatives. One of the simplest algorithms of this type is *component-wise optimization*. In this algorithm, to find the values $x_1^{\max}, \ldots, x_n^{\max}$ at which a given function $f(x_1, \ldots, x_n)$ attains its maximum, we start with some initial values $x_1^{(0)}, \ldots, x_n^{(0)}$. Then, we do the following:

- First, we fix all the values but x_1, i.e., we take

$$x_2 = x_2^{(0)}, \ldots, x_n = x_n^{(0)},$$

and find the value $x_1^{(1)}$ for which the expression $f\left(x_1, x_2^{(0)}, \ldots, x_n^{(0)}\right)$ is the largest possible.

- Then, we fix all the values but x_2, i.e., we take

$$x_1 = x_1^{(1)}, x_3 = x_3^{(0)}, \ldots, x_n = x_n^{(0)},$$

and we find the value $x_2^{(1)}$ for which the expression $f\left(x_1^{(1)}, x_2, x_3^{(0)} \ldots, x_n^{(0)}\right)$ is the largest possible.

- Then, we repeat the same to find new values of x_3, x_4, …, x_n.
- Once the new values $x_1^{(1)}$, …, $x_n^{(1)}$ of all the variables $x_1, \ldots, x_n$ are found, we again fix the values of all the variables but x_1, find the new value $x_1^{(2)}$ of x_1, etc.
- We stop when we do not get any improvement, i.e., when the values on the next iteration are equal to (or close to) values on the previous iteration.

In more formal terms, we can describe this algorithm as follows. In this algorithm, we start with the initial values $x_1^{(0)}, \ldots, x_n^{(0)}$, and then we perform iterations. On each iteration, we start with the values $x_1^{(k)}, \ldots, x_n^{(k)}$ obtained on the previous iteration (or, for the first iteration, with the initial values). Each iteration consists of n computational stages. On each stage $i = 1, \ldots, n$, we find the next value $x_i^{(k+1)}$ in the following way:

- we fix the previously obtained values of all the variables except for x_i, i.e., we take

$$x_1 = x_1^{(k+1)}, \ldots, x_{i-1} = x_{i-1}^{(k+1)},$$

$$x_{i+1} = x_{i+1}^{(k)}, \ldots, x_n = x_n^{(k)};$$

- we then find a value x_i for which the function $f\left(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i, x_{i+1}^{(k)}, \ldots, x_n^{(k)}\right)$ attains the largest value;
- this value x_i is then taken as the new value $x_i^{(k+1)}$.

We stop when

$$\left|x_i^{(k+1)} - x_i^{(k)}\right| \le \varepsilon$$

for all i, for some appropriate $\varepsilon > 0$.

Let us apply the selected algorithm to our problem. Since in the normal case, the measurement results x_i come from measuring the same distance, it is reasonable to start with equal values $x_1^{(0)} = \ldots = x_n^{(0)} = d_0$, for some appropriate distance d_0. Let us now implement the first iteration.

First stage: general description. At the first stage, according to the general algorithm, we select a value x_1 for which the difference

$$\frac{x_1 + d_0 + \ldots + d_0}{n} - \min(x_1, d_0, \ldots, d_0)$$

is the largest possible. The first term in this expression has the form

$$\frac{x_1 + (n-1) \cdot d_0}{n}.$$

The value of the second (minimum) term depends on whether $x_1 \le d_0$ or $x_1 \ge d_0$. Let us consider these two cases one by one.

First stage: case when $x_1 \le d_0$. In this case, the minimum term $\min(x_1, \ldots, x_n)$ is equal to x_1 and thus, the maximized function is equal to

$$\frac{x_1 + (n-1) \cdot d_0}{n} - x_1 = \frac{n-1}{n} \cdot d_0 - \frac{n-1}{n} \cdot x_1.$$

This function decreases with x_1, so for values $x_1 \in [0, d_0]$, the largest possible value of the objective function is attained when x_1 attains its smallest possible value $x_1 = 0$. For this value $x_1 = 0$, the above objective function takes the form

$$\frac{n-1}{n} \cdot d_0. \tag{2}$$

First stage: case when $x_1 \ge d_0$. In this case, the minimum term $\min(x_1, \ldots, x_n)$ is equal to d_0 and thus, the maximized function is equal to

$$\frac{x_1 + (n-1) \cdot d_0}{n} - d_0.$$

This function increases with x_1, so for values $x_1 \in [d_0, D]$, the largest possible value of the objective function is attained when x_1 attains its largest possible value $x_1 = D$. For this value $x_1 = D$, the above objective function takes the form

$$\frac{n-1}{n} \cdot d_0 + \frac{1}{n} \cdot D - d_0 = \frac{1}{n} \cdot (D - d_0). \tag{3}$$

First stage: final result. By considering the above two cases, we conclude that the maximum of our function is attained either for $x_1 = 0$, or for $x_1 = D$. To find the result $x_1^{(1)}$ of the first stage, we need to check which of the values is larger. By comparing the results (2) and (3) corresponding to $x_1 = 0$ and $x_1 = D$, we conclude that the value $x_1 = 0$ leads to the larger value of the objective function if and only if

$$\frac{n-1}{n} \cdot d_0 \ge \frac{n-1}{n} \cdot d_0 + \frac{1}{n} \cdot D - d_0,$$

i.e., if and only if $D \le d_0 \cdot n$. So, we arrive at the following conclusion:

- if $D \le d_0 \cdot n$, then we take

$$x_1^{(1)} = 0;$$

- if $D \ge d_0 \cdot n$, then we take

$$x_1^{(1)} = D.$$

In the analysis of the second stage, we will consider both choices.

Second stage: case when $D \le d_0 \cdot n$ and $x_1^{(1)} = 0$. In this case, the objective function takes the form

$$\frac{0 + x_2 + (n-2) \cdot d_0}{n} - 0 = \frac{1}{n} \cdot x_2 + \frac{n-1}{n} \cdot d_0.$$

This function increases with x_2, so for values $x_2 \in [0, D]$, the largest possible value of the objective function is attained when x_2 attains its largest possible value $x_2 = D$. Thus, in this case, we take $x_2^{(1)} = D$.

Second stage: case when $D \geq d_0 \cdot n$ and $x_1^{(1)} = D$. The situation in this case depends on whether $n = 2$ or $n > 2$.

If $n = 2$, then the objective function takes the form

$$\frac{D + x_2}{2} - \min(D, x_2) = \frac{D + x_2}{2} - x_2 = \frac{1}{2} \cdot D - \frac{1}{2} \cdot x_2.$$

This function decreases with x_2, so for values $x_2 \in [0, D]$, the largest possible value of the objective function is attained when x_2 attains its smallest possible value $x_2 = 0$. Combining with the previous case, we conclude that for $n = 2$, one of the two values x_i should be equal to 0, and another one to D.

If $n > 2$, the objective function takes the form

$$\frac{D + x_2 + (n-2) \cdot d_0}{n} - \min(D, x_2, d_0, \ldots, d_0).$$

The value of the minimum term depends on whether $x_2 \leq d_0$ or $x_2 \geq d_0$. Let us consider these two cases one by one.

If $x_2 \leq d_0$, then the objective function takes the form

$$\frac{D + x_2 + (n-2) \cdot d_0}{n} - x_2 = \frac{D + (n-2) \cdot d_0}{n} - \frac{n-1}{n} \cdot x_2.$$

This function decreases with x_2, so for values $x_2 \in [0, d_0]$, the largest possible value of the objective function is attained when x_2 attains its smallest possible value $x_2 = 0$. The corresponding value of the objective function is equal to

$$\frac{D + (n-2) \cdot d_0}{n}. \tag{5}$$

If $x_2 \geq d_0$, then the objective function takes the form

$$\frac{D + x_2 + (n-2) \cdot d_0}{n} - d_0.$$

This function increases with x_2, so for values $x_2 \in [d_0, D]$, the largest possible value of the objective function is attained when x_2 attains its largest possible value $x_2 = D$. The corresponding value of the objective function is equal to

$$\frac{D + D + (n-2) \cdot d_0}{n} - d_0 = \frac{D + (n-2) \cdot d_0}{n} + \frac{D}{n} - d_0. \tag{6}$$

Since we consider the case when $D \geq n \cdot d_0$, the expression (6) corresponding to $x_2 = D$ is larger than the expression (5) corresponding to $x_2 = 0$. Thus, we take $x_2^{(1)} = D$.

So, we arrive at the following conclusions:

Second stage: final result for $n = 2$.

- If $D \leq d_0 \cdot n$, then we take
$$x_1^{(1)} = 0 \text{ and } x_2^{(1)} = D;$$
- if $D \geq d_0 \cdot n$, then we take
$$x_1^{(1)} = D \text{ and } x_2^{(1)} = 0.$$

Second stage: final result for $n > 2$. We always take $x_1^{(2)} = D$. In other words:

- if $D \leq d_0 \cdot n$, then we take
$$x_1^{(1)} = 0 \text{ and } x_2^{(1)} = D;$$
- if $D \geq d_0 \cdot n$, then we take
$$x_1^{(1)} = D \text{ and } x_2^{(1)} = D.$$

k-th stage: a natural hypothesis and its induction-based proof. Let us prove, by induction, that on each stage k, if $k < n$, then:

- if $D \leq d_0 \cdot n$, then we take
$$x_1^{(1)} = 0 \text{ and } x_2^{(1)} = \ldots = x_k^{(1)} = D;$$
- if $D \geq d_0 \cdot n$, then we take
$$x_1^{(1)} = x_2^{(1)} = \ldots = x_k^{(1)} = D.$$

If $k = n$, then:

- if $D \leq d_0 \cdot n$, then we take
$$x_1^{(1)} = 0 \text{ and } x_1^{(1)} = \ldots = x_k^{(1)} = D;$$
- if $D \geq d_0 \cdot n$, then we take
$$x_1^{(1)} = x_2^{(1)} = \ldots = x_{k-1}^{(1)} = D \text{ and } x_k^{(1)} = 0.$$

We know that the above is true for $k = 1$ and $k = 2$. Let us assume that this is true for selecting the values $x_1^{(1)}, x_2^{(1)}, \ldots, x_{k-1}^{(1)}$, and let us prove that it is true for selecting the next value $x_k^{(1)}$ as well.

k-th stage: case when $D \leq d_0 \cdot n$, $x_1^{(1)} = 0$, and $x_1^{(1)} = \ldots = x_{k-1}^{(1)} = D$. In this case, the objective function takes the form

$$\frac{0 + (k-2) \cdot D + x_k + (n-k) \cdot d_0}{n} - 0.$$

This function increases with x_k, so for values $x_k \in [0, D]$, the largest possible value of the objective function is attained when x_k attains its largest possible value $x_k = D$. Thus, in this case, we indeed take $x_k^{(1)} = D$.

k-th stage: case when $D \geq d_0 \cdot n$ and $x_1^{(1)} = \ldots = x_{k-1}^{(1)} = D$. The situation in this case depends on whether $n = k$ or $n > k$.

If $n = k$, then the objective function takes the form

$$\frac{(n-1) \cdot D + x_n}{n} - \min(D, \ldots, D, x_n) = \frac{(n-1) \cdot D + x_n}{n} - x_n = \frac{n-1}{n} \cdot D - \frac{n-1}{n} \cdot x_n.$$

This function decreases with x_n, so for values $x_n \in [0, D]$, the largest possible value of the objective function is attained when x_n attains its smallest possible value $x_n = 0$. So, in this case, we select $x_k^{(1)} = 0$.

If $n > k$, the objective function takes the form

$$\frac{(k-1) \cdot D + x_k + (n-k) \cdot d_0}{n} - \min(D, \ldots, D, x_k, d_0, \ldots, d_0).$$

The value of the minimum term depends on whether $x_k \leq d_0$ or $x_k \geq d_0$. Let us consider these two cases one by one.

If $x_k \le d_0$, then the objective function takes the form

$$\frac{(k-1)\cdot D + x_k + (n-k)\cdot d_0}{n} - x_k =$$

$$\frac{(k-1)\cdot D + (n-k)\cdot d_0}{n} - \frac{n-1}{n}\cdot x_k.$$

This function decreases with x_k, so for values $x_k \in [0, d_0]$, the largest possible value of the objective function is attained when x_k attains its smallest possible value $x_k = 0$. The corresponding value of the objective function is equal to

$$\frac{(k-1)\cdot D + (n-k)\cdot d_0}{n}. \tag{7}$$

If $x_k \ge d_0$, then the objective function takes the form

$$\frac{(k-1)\cdot D + x_k + (n-k)\cdot d_0}{n} - d_0.$$

This function increases with x_k, so for values $x_k \in [d_0, D]$, the largest possible value of the objective function is attained when x_k attains its largest possible value $x_k = D$. The corresponding value of the objective function is equal to

$$\frac{(k-1)\cdot D + D + (n-k)\cdot d_0}{n} - d_0 =$$

$$\frac{(k-1)\cdot D + (n-k)\cdot d_0}{n} + \frac{D}{n} - d_0. \tag{8}$$

Since we consider the case when $D \ge n \cdot d_0$, the expression (8) corresponding to $x_k = D$ is larger than the expression (7) corresponding to $x_k = 0$. Thus, we take $x_k^{(1)} = D$.

The statement is proven.

The result of the first iteration is actually the global maximum. So far, we have traced only one iteration of the component-wise optimization algorithm. Let us show that the second iteration will not improve the value of the objective function and thus, the first-iteration vector is what our algorithm will return.

For that, let us show that the vector obtained on the first iteration – in which one component is 0 and all other components are equal to D – actually corresponds to the largest possible value of the objective function (1).

Indeed, one can easily check that the objective function (1) is a convex function; see, e.g., [9]. Thus, due to the known properties of convex functions (see, e.g., [9]), the maximum of the objective function (1) on a box

$$[0, D] \times \ldots \times [0, D]$$

is attained at one of the vertices, i.e., when some of the values x_i are equal to 0 and some to D. The value of the objective function does not change if we swap some values x_i, it only depends on how many values x_i are equal to 0 and how many to D. Without losing generality, we can therefore take $x_1 = \ldots = x_z = D$ and $x_{z+1} = \ldots = x_n = 0$, for some index z.

When $z = 0$, all the values x_i are equal to 0. In this case, both the mean $\overline{x}$ and the minimum term m are equal to 0, so the difference (1) is also equal to 0.

Similarly, when $z = n$, all the values x_i are equal to D. In this case, both the mean $\overline{x}$ and the minimum term m are equal to D, so the difference (1) is equal to 0.

Since the difference $\overline{x} - m$ is always non-negative, its maximum should be found when $z = 1, \ldots, n-1$. In this case,

$$\frac{x_1 + \ldots + x_z + x_{z+1} + \ldots + x_n}{n} -$$

$$\min(x_1, \ldots, x_z, x_{z+1}, \ldots, x_n) =$$

$$\frac{z\cdot D + (n-z)\cdot 0}{n} - \min(D, \ldots, D, 0, \ldots, 0) =$$

$$\frac{z\cdot D}{n} - \min(0, D) = \frac{z\cdot D}{n}.$$

This expression increases with z, so for values

$$z \in \{1, \ldots, n-1\},$$

the largest possible value of the objective function is attained when z attains its largest possible value $z = n-1$, i.e., when $n-1$ values x_i are equal to D and one value is equal to 0.

This is exactly what we got after the first iteration. So, the result of the first iteration is indeed the global maximum.

IV. General Recommendation

Based on this positive computational experience, we can recommend to use *component-wise optimization* as a general technique for finding the parameters corresponding to the worst-case scenario.

V. From Crisp Case to a More Realistic Case of Soft Constraints

In the above text, we assumed:

- that we know the exact bound D on the possible distances x_i, and
- that we have no information about which combinations $x = (x_1, \ldots, x_n)$ are more probable and which are less probable.

In practice, we only know such bounds with uncertainty, but we also have some information about which combinations are more probable and which are less probable. This information is usually described in imprecise terms, by using words from a natural language. It is therefore reasonable to use fuzzy techniques (see, e.g., [4], [6], [12]) to describe this information. In the fuzzy approach, we assign, to every vector x, a degree $\mu(x)$ to which the corresponding combination of values is probable.

Then, to find the worst-case scenario, instead of optimizing an objective function $f(x)$ under crisp constrains, we need to optimize the objective function under such *soft* (fuzzy) constraints; for this optimization, we can use known techniques of optimizing a (crisp) function over fuzzy sets (see, e.g., [4], [6]).

For example, we can use Bellman-Zadeh techniques in which we maximize the expression

$$g(x) \stackrel{\text{def}}{=} \min\left(\frac{f(x) - \underline{y}}{\overline{y} - \underline{y}}, \mu(x)\right),$$

where $\underline{y}$ and $\overline{y}$ are the minimum and maximum of the function $f(x)$ over the entire domain, the ratio

$$\frac{f(x) - \underline{y}}{\overline{y} - \underline{y}}$$

describes to what extent the vector x is optimal, and the optimized expression $g(x)$ reflects the requirement that the desired vector x must be optimal *and* it must satisfy the given constraints – with $\min$ corresponding to "and".

ACKNOWLEDGMENTS

The authors are thankful to the anonymous referees for valuable suggestions.

REFERENCES

[1] R. T. Brooks and A. P. Sage, "System of systems integration and test", *Information Knowledge Systems Management*, 2005/2006, Vol. 5, pp. 261–280.

[2] C. Haskins (ed.), *Systems Engineering Handbook: A Guide for Systems Life Cycle Processes and Activities*, Version 3.2.2, International Council on Systems Engineering (INCOSE) Document INCOSE-TP-2003-002-03.2.2, San Diego, CA, 2011.

[3] C. B. Keating, "Emergence in System of Systems", in: M. Jamshidi (ed.), *System of Systems Engineering: Innovations for the 21st Century*, John Wiley and Sons, 2009, pp. 169–190.

[4] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[5] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1997.

[6] H. T. Nguyen and E. A. Walker, *First Course In Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[7] P. M. Pardalos, *Complexity in Numerical Optimization*, World Scientific, Singapore, 1993.

[8] R. L. Pineda, "Understanding Engineered Complex Systems of Systems (CSoS), *Proceedings of The Fourth General Assembly of the Cartagena Network of Engineering CNE'2010*, Metz, France, September 21–24, 2010.

[9] R. T. Rockafeller, *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1970.

[10] B. Rogers and E. Gilbert, "Identifying arhitectural modularity in the smart grid: an application of design structure matrix methodology", *Proceedings of the Grid-Interop Forum'2011 "Implementing Interoperability Advancing Smart Grid Standards, Architecture and Community"*, Phoenix, Arizona, December 5–8, 2011.

[11] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, Boca Raton, Florida, 2007.

[12] L. A. Zadeh, "Fuzzy sets", *Information and control*, 1965, Vol. 8, pp. 338–353.