

# INTEGRATING COST, EFFECTIVENESS, AND STABILITY

*Dr. James P. Ignizio*

Various approaches, ranging from conventional cost-effectiveness ratios to newer multiple objective/multiple criteria models, have been used in an attempt to systematically factor cost and effectiveness into acquisition decisions. While this is laudable, we believe that one point has been overlooked: the critical nature of the inherent stability of the system to be developed and/or acquired. We often hear about cost overruns; but what we may not realize is that the underlying cause of such overruns may be something as simple as forgetting to consider the stability of the system. In this paper we attempt to indicate the nature and impact of stability in cost-effectiveness studies—and to propose topics for further investigation.

**W**hether funding an entirely new system (e.g., an air defense system) or including a new component into an existing system (e.g., inclusion of a new class of helicopter into a rapid deployment system), the goal is the same—procure a system that provides the “biggest bang per buck.” To accomplish this, we generally seek a solution that minimizes cost yet achieves certain target levels for a variety of multiple measures of system performance (e.g., range, accuracy, reliability, weight, volume, survivability, availability). Rarely, if ever, does one see the inclusion of system stability as a measure of system performance. And this may explain why program managers discover that their system has unexpectedly experienced a

significant cost overrun—as a consequence of its inherent instability.

Before proceeding further, let us define the notion of stability itself. First, stability analysis is not the same thing as sensitivity analysis, risk analysis, or reliability analysis. Conventional sensitivity analysis, risk analysis, or reliability analysis can be effective for dealing with systems of limited size and complexity, if sufficient data (e.g., probabilities of component failure) exists to support the needs of the analysis. However, in many of the real world systems under consideration today, problem size, system complexity and inherent errors (or gaps) in data invariably exist. Furthermore, these conventional methods merely estimate the likelihood of the failure (or degradation) for a given

system. As such, they result in estimates of such things as MTBF (mean time between failures), or the probability of a particular failure mode, or the impact of the variation of a given parameter on the performance of a system.

Stability analysis, on the other hand, addresses the topic of the “likely worst case performance” of a system—normally a system too large, complex, or messy (i.e., the typical real world system) to be dealt with effectively by more conventional means. Extensive experience (e.g., within the aerospace sector, the military sector, and even the financial sector) has shown that a system having a high degree of reliability can and often will fail as an unforeseen consequence of the failure of a certain combination of components. Usually, the only way to evaluate such potentially catastrophic consequences beforehand would be to run a brute force evaluation of all possible combinations of failures. For any real world system, such a process would take years, if not centuries, on even our most powerful computers.

As such, one can think of stability analysis as a systematic attempt to examine the likely worst-case performance of a complex system. In other words, it is an attempt to account for the consequences of “Murphy’s Third Law: Anything that can go wrong will.” In the following example, the consequences of limiting the

analysis of a system to just conventional performance measures is described—and this may be contrasted to the results actually exhibited once a prototype of that system was actually constructed and tested.

Some years ago a contractor for the Navy was tasked with the job of developing an acoustic array for torpedoes. After carefully determining the various measures of effectiveness (e.g., range, sensitivity, sidelobe suppression, etc.) and the cost elements, the contractor—aided by a group of academicians—tried to determine the “optimal” system, in terms of cost-effectiveness. They constructed a mathematical model of the system and then computed, by means of mathematical optimization tools, the solution: the precise amplitude and phase of the acoustic energy to be delivered to each transducer in the array—accurate to the fifth decimal point.

The design team went further in their analysis than is typical; performing a laborious sensitivity analysis of the array as a function of changes in the amplitudes and phases fed each array element. Since the mathematical model was nonlinear, and since the number of array elements were in the hundreds, they could—of course—only examine a finite and relatively limited number of combinations of perturbations in amplitude and phase.

**James P. Ignizio, Ph.D.**, is a professor of engineering at the University of Virginia and a visiting professor at the U.S. Army Logistics Management College, Fort Lee, VA. Dr. Ignizio is the author of seven books, more than two dozen monographs, and more than 250 technical papers, including more than 90 in various international, refereed, journals. He worked for seven years in the aerospace and military sector and has taught at Pennsylvania State University and the University of Houston. His primary areas of interest are intelligent decision systems, artificial intelligence, financial systems, and applied operations research. Dr. Ignizio is a Fellow of IIE and recipient of the First Hartford Prize.

On paper the solution looked fabulous. The laboratory consisted of a water tunnel in which the array could be tested in what were close to actual conditions. The results for the prototype array that had been fabricated for testing weren't nearly so good. In fact, they were awful. The problem? A slight change in some combination of seemingly insignificant changes in two or more attributes could result in a significant degradation of the array's performance. If this array design were actually deployed, it would require the development of either super-sensitive receivers or a major breakthrough in manufacturing fabrication tolerances. Either alternative would add literally millions of dollars to the cost of the total system (translation: a cost overrun).

Our experience, over three decades of system design and cost-effectiveness analysis, indicates that systems designed to be "optimal" are surprisingly often more likely to be unstable than systems that are more conservative (i.e., less "effective") in design—at least on paper.

"Optimal" acquisition decisions, no matter how cost-effective in the conventional sense, may not be a "good thing."

### **GRAPHICAL ILLUSTRATIONS OF INSTABILITY**

The reason system stability is so important, and why it is so easily overlooked, may be explained graphically. Consider a very simple problem in which the performance of a system is a (nonlinear) function of a single variable. The output of this system, as a function of some variable  $x$ , is depicted in Figure 1. The possible values of  $x$  range from 0.00 to 0.20. The system performance has two main peaks.

The highest is centered at about 0.02 while another peak (somewhat lower but much broader) is centered at about 0.16.

Conventional methods of analysis are optimal seeking. As such, they would determine that the optimal value of  $x$  is found at  $x = 0.02$  (for a performance value—shown on the vertical axis—of 9 units). However, should the performance function be off as little as 0.01 units (i.e., imagine a horizontal shift in the nonlinear function some 0.01 units to the left or right), the system performance drops from 9 units to 0.1 units—a fall of roughly 99 percent!

Yet, had we selected a less than optimal solution, at  $x = 0.16$ , the performance of the system would only vary between 8.4 and 8.5 units. In fact, for a solution at  $x = 0.16$ , the performance function could shift by 0.02 units (twice that which virtually destroyed the "optimal" system), and still result in a performance level between 8 and 8.5 units.

In a problem this simple, conventional sensitivity analysis (e.g., perturbations of variables and, possibly, the mapping of the response surface) would likely suffice. However, for larger, more realistic problems (i.e., those having numerous variables and in which we have little or no idea as to the shape of the functions involved), such an approach might not be practical.

"Our experience... indicates that systems that are designed to be "optimal" are surprisingly often more likely to be unstable than systems that are more conservative (i.e., less "effective") in design—at least on paper."

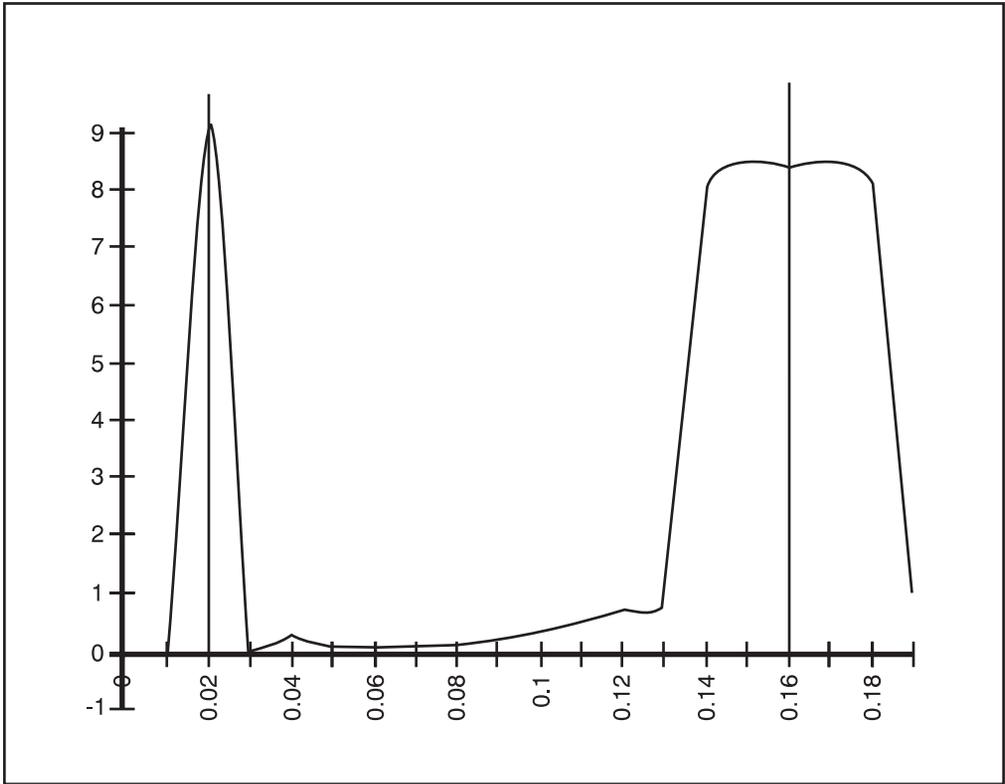


Figure 1. Nonlinear Response Function

In the next section we present a brief overview of existing methods for stability analysis. In the section following that, a proposal for an unconventional approach to stability is presented.

### **INCLUDING STABILITY IN COST-EFFECTIVENESS ANALYSIS**

Some would claim that all one has to do to consider system stability is to perform a sensitivity analysis on the model. The problem with this assumption is twofold. First, conventional sensitivity analyses are intended, for the most part, for strictly linear, continuous, problems—and

the systems that are to be considered for real world procurement or development are invariably (highly) nonlinear. More specifically, they are problems of combinatorial complexity. Second, even if the problem is linear (or a linear approximation seems reasonable), sensitivity analysis is not the same thing as stability analysis.

Sensitivity analysis tells us only the range over which some solution remains optimal (e.g., changes in the basis of linear programming models), or that range of model coefficient values over which the solution still satisfies all constraints (Ignizio and Cavalier, 1994). That type of information, in itself, tells us little about the inherent stability of a given solution.

Other analysts have suggested that we “simply include solution stability” as another objective or constraint—that is, as yet another measure of effectiveness. That would be fine if we could determine a way in which to capture solution stability as a mathematical function; but there is no effective way of doing this.

Others would argue that we use “perturbation analysis.” In perturbation analysis we perturb the values of various variables and coefficients in a model, singly and in combination, and observe the effect (Fiacco and Ishizuka, 1990; Perlis and Ignizio, 1980).

For relatively small problems this may provide satisfactory results. But for problems more typical of those faced in the real world, the amount of effort required would be truly enormous. Consider, for example, just a modestly sized problem involving 10 constraint functions, 5 objective (measure of performance) functions, and 20 decision variables (e.g., the value for each of 20 rapid deployment force weapon types). In this problem there would be some 310 model coefficients and right-hand side values. If we restricted our perturbation analysis to just the evaluation of changes in each coefficient and each pair of coefficients, we would be required to perform nearly 48,000 evaluations—each of which would require a large number of sub-evaluations (e.g., analysis of the increments in change of each coefficient or pair of coefficients). Even then we will have ignored those combinations taken three or more at a time. As such, even this modest problem could not be adequately analyzed by means of perturbation analysis.

## **STABILITY ANALYSIS VIA GENETIC ALGORITHMS**

---

If conventional methods cannot assist us in an evaluation of stability analysis, then unconventional approaches should be considered. Recall the acoustic array design problem discussed in an earlier section. The “optimal” design was unstable—too unstable to even consider. However, a “less-than-optimal” design achieved the desired stability without an excessive degradation in the performance measures that were promised, on paper, by the optimal system.

Less-than-optimal conservative designs would appear to have a greater likelihood of remaining stable. Instead of being located on boundaries of the solution space, or at the extreme points of that space, they tend to be found in the interior of the space, and are thus dominated, or inefficient. Consequently, any changes in the system, or any errors in the data should have less impact on these solutions.

A preliminary exploration of an “evolutionary” approach to stability analysis provides a practical means to systematically analyze both the effectiveness and the stability of a solution. In this approach either genetic or evolutionary programming is employed to produce a population of final solutions.

Genetic algorithms (GA) pursue a “survival of the fittest” search for optimality

“ If conventional methods cannot be relied upon to assist us in an evaluation of stability analysis, then unconventional approaches should, we believe, be considered.”

(Davis, 1991; Ignizio and Cavalier, 1994; Zalzalá and Fleming, 1997). Typically, they utilize a Boolean coding of the solution variables—and the result is termed a “chromosome.” Evolutionary programming is essentially the same thing, but uses real numbers in coding. For example, the chromosome  $x = (1\ 0\ 1\ 1\ 0\ 0)$  may mean that we fund project 1, 3, and 4—and do not fund projects 2, 5, and 6. We see then that a “1” in the  $x$ -vector indicates that the associated project is funded, while a “0” signifies an unfunded effort.

Rather than starting from a single solution point and conducting an iterative search, and then moving in the direction of local optimality, genetic or evolutionary programming begins with a population of solutions

“Inherently unstable solutions de-evolve (the reverse of evolution) to poor solutions faster than stable solutions.”

(tens, or even hundreds per “generation”). These are evaluated for their “fitness” (i.e., the correspond-

ing objective function, or functions are evaluated) and the “fittest” of the bunch are considered for “mating” and “reproduction.” Choices are made stochastically, with preference given to the most fit. Mated pairs may reproduce, via an exchange of “chromosomes.” However, since reproduction is also stochastic, not all mated pairs will reproduce. Yet another stochastic element is introduced in the reproduction process; some chromosomes may be mutated. These operations result in the population of the next generation. And the procedure continues until some termination criterion is reached. The references provide further information on these notions. However, a detailed under-

standing of GA is not needed to appreciate the proposed procedure.

While global optimality cannot be guaranteed, genetic and evolutionary algorithms are effective in finding “very good” solutions. In addition, the approach is inherently parallel, and thus computational effort is minimized.

Our proposal is simple and straightforward. Use a genetic or evolutionary programming algorithm to solve the problem under consideration. During the processing of the algorithm, maintain a file of the top 10 or 20 solutions generated by the algorithm. At convergence, proceed to examine this list of solutions for stability.

The question that remains, of course, is just how to evaluate—in a practical manner—the stability of a given solution. Our approach is based upon the following hypothesis: Inherently unstable solutions de-evolve (the reverse of evolution) to poor solutions faster than stable solutions.

Furthermore, if a genetic algorithm is effective in evolving a population of solutions toward a high level of fitness, it would seem reasonable to believe that it would be equally effective in de-evolving optimal or near-optimal solutions to those that are of poor quality. In other words, we will use the GA on good solutions in an attempt to see how long (how many generations) it takes to de-evolve.

The approach that we are investigating may be summarized via the following steps:

- *Step 1:* Model the problem under consideration in a format compatible with GA (e.g., code the solution representation, determine crossover, mutation, and reproduction operators, and pro-

vide a means for the evaluation of the fitness of each solution).

- *Step 2:* Use a GA to solve the problem; maintain a list of the 10 or 20 best solutions generated by the GA.
- *Step 3:* For each of the solutions in the list of the 10 or so best solutions, develop a cluster of solutions about that point by means of perturbation.
- *Step 4:* For each of the clusters in Step 3, use a GA to search for the least fit solution; and record the number of generations required to de-evolve to a sufficiently poor solution.
- *Step 5:* Solutions (i.e., the original solutions used to develop the clusters of Step 3) that de-evolve the slowest are assumed to be the most stable. Those that de-evolve fastest are assumed to be least stable.

We should elaborate further on Steps 3 and 4. In Step 3 we generate a tight cluster, about each point, by means of random perturbation of the coordinates of that point—where each perturbation is held to a small size. In Step 4 we could easily de-evolve to a poor solution by moving across a constraint into an infeasible region. However, the GA should be set so as to disallow such moves (e.g., assign any infeasible point some very high value). In this way the de-evolution takes place entirely within the (assumed) solution space.

Our next step is to perform small perturbations about each of these solutions so as to develop a “population” of solutions all in close proximity to the original solution point.

Following the development of the clusters we would use a GA, on one cluster at

a time, to generate poor solutions. Our interest is centered about the number of generations required to de-evolve to some predetermined level representing a poor solution. Those

clusters taking longest (i.e., in terms of number of generations) to reach that level are assumed to be the most stable. For example, if it took 50 generations for the cluster about one particular solution (X) to de-evolve, and only 10 for that about another (Y), the first solution (i.e., X) would be considered a far more stable solution.

How well does this approach work? The only proper way to evaluate the method is to apply it to real world problems with real world data. Obviously, that is time-consuming. Thus far, we have used the approach on a handful of actual problems and found the results, in each instance, to be very useful in predicting solution stability.

---

## ILLUSTRATION

To illustrate the concept, consider a more tangible example: the procurement of an air defense system. Such a system is composed of a variety of subsystems and components, and its actual design is a problem of combinatorics. These consist of such things as the choices of missiles,

“ Our next step is to perform small perturbations about each of these solutions so as to develop a “population” of solutions (all in close proximity to the original solution point).”

missile warheads, radar types, support vehicle types, and so on. Typically, the number of possible combinations is massive.

A genetic algorithm is a highly effective approach to problems of combinatorics. As such, we could use such an algorithm to generate not just the “best solution,” but also a list of the top 10, 20, or so best solutions (i.e., combinations of subsystems and components forming the air defense system). We performed such an analysis for the design of a number of possible air defense systems, each differing according to the combination of subsystems and components that make up the system. Using simulation, we then determined the top 10 candidates (according to their effectiveness to cost ratios), as listed in Table 1.

Our next step was to form tight clusters about each candidate solution. This was achieved by perturbing each solution ever so slightly: that is, exchange one or two components in the present solution for components not in solution. This exchange is repeated until we have a population of

solutions (i.e., clusters) centered about each candidate solution.

We then apply a GA to each of the clusters (one at a time) and record the number of generations needed to de-evolve. The cluster taking the longest is considered the most stable population, and the original solution from which that cluster was generated is selected as the most stable solution.

What we discovered was that candidate system A may have had the “optimal” effectiveness to cost ratio, but it was extremely unstable. Candidate system F, on the other hand was, far and away, the most stable of the 10 prototype air defense systems. While its effectiveness to cost ratio may be somewhat lower than that of A, it likely makes up for that deficiency in terms of its superior stability. After all, would the military prefer an air defense system that is optimal on paper, yet whose performance can radically be degraded by the chance combination of failures for a number of components?

Table 1: Candidate Air Defense Systems

Candidate	Cost	Effectiveness	Effectiveness/ Cost Ratio
A	50	80.00	1.6
B	48	76.32	1.59
C	52	84.42	1.585
D	54	84.78	1.57
E	49	76.44	1.56
F	53	81.62	1.54
G	56	84.00	1.5
H	57	79.80	1.4
I	56	67.20	1.2
J	59	64.90	1.1

## **SUMMARY**

---

In this paper we have explored the notion that cost-effectiveness analyses, in support of the acquisition process, may be ignoring an extremely important, if not critical factor: the inherent stability of the system. In such problems, stability may be far more important than such conventional, and comfortable, notions as optimality (i.e., the “best” cost-effectiveness). Yet, while debate rages as to what tool or tools are needed to develop optimal designs, relatively little attention is paid to the stability of solutions in what

will always be imperfect mathematical models.

We propose that consideration be given to the concept of considering solution stability by means of evolutionary algorithms. While our hypothesis (i.e., that inherently unstable solutions de-evolve to poor solutions much faster than inherently stable results) has only been explored at a preliminary level, the results thus far seem to support the thesis. Hopefully, this paper will encourage further efforts in this area.

### **ACKNOWLEDGMENT**

This paper was supported in part through an Intergovernmental Personnel Act between the University of Virginia and the U.S. Army Logistics Management College, Fort Lee, VA. This paper represents the views of the author and does not necessarily reflect the official opinion of the Army Logistics Management College.

## REFERENCES

---

- Davis, L. (Ed.). (1991). *Handbook of genetic algorithms*. New York: Van Nostrand Reinhold.
- Fiacco, A. V. & Ishizuka, Y. (1990). Sensitivity and stability analysis for nonlinear programming. *Annals of Operations Research*, 27, pp. 215–235.
- Ignizio, J. P. & Cavalier, T. M. (1994). *Linear programming*. Englewood Cliffs, NJ: Prentice-Hall.
- Perlis, J. H. & Ignizio, J. P. (1980). Stability analysis: An approach to the evaluation of system design. *Cybernetics and Systems*, 11, pp. 87–103.
- Zalzala, A. M. S. & Fleming, P. J. (Eds.). (1997). *Genetic algorithms in engineering systems*. London: The Institution of Electrical Engineers.