THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 50, No. 4, April, 1971
Printed in U.S.A.

Statistical Circuit Design:

The Optimum Assignment of Component Tolerances for Electrical Networks

By B. J. KARAFIN

(Manuscript received December 18, 1970)

The prediction of manufacturing yield of an electric circuit, given the tolerances and statistics of the components, is a straightforward procedure using Monte Carlo tolerance analysis. The inverse problem, namely, specifying the component tolerances that produce the cheapest network, is more difficult and has received little attention in the literature. In this paper an algorithm is presented that solves this problem for a significant class of networks.

The algorithm is limited to circuits for which one hundred percent yield is sought and whose components are statistically independent, i.e., discrete circuits. The one hundred percent yield assumption is used to reduce the number of possible tolerance choices. A variation of the branch and bound strategy that is shown to be particularly efficient is then used to make an optimum selection. Two-dimensional performance contours, worst-case, and Monte Carlo computations are also used as part of the procedure.

An example circuit is demonstrated for which the algorithm produced a tolerance assignment substantially cheaper than that projected by the designer.

I. INTRODUCTION

Monte Carlo tolerance analysis has proven to be a useful tool in evaluating the effects of component tolerances and environmental variations on electrical circuit performance. The method involves "constructing" samples of the circuit inside the computer using element values that obey the manufacturing statistics, analyzing these samples, and forming empirical distributions of performance. One common outcome of the process is the prediction of yield.

Monte Carlo tolerance analysis, henceforth referred to as TAP

(Tolerance Analysis Procedure), is an open-loop structure. If we examine the way in which it is used, we find that for discrete circuits the designer supplies the TAP program with a set of component tolerances he has chosen based on breadboard measurements, linear sensitivity or worst-case analysis, or some other approximate technique. At the conclusion of the TAP run, he observes yield and is faced with one of two situations:

(*i*) Yield is too low. With this result the designer knows he must change his tolerances. Unfortunately, TAP gives him little information as to which tolerances to change and by how much.

(*ii*) Yield is adequate. Here the designer may be satisfied by the design but he obtains little help in determining whether a cheaper (looser) set of tolerances might not give equally satisfactory yield.

TAP, then, is open-loop in that it is a tool for predicting yield given a set of tolerances; both the initial set of tolerances or any changes to that set must be provided by the designer without assistance from TAP. (It is interesting to note that for integrated circuits the designer has little freedom in specifying tolerances or tracking. In this case TAP is used to check that designs themselves are adequate, *given* the component tracking that can be achieved in integrated production.)

This paper discusses an algorithm for closing the TAP loop. The program embodying this algorithm will accept a circuit topology, nominal element values, a figure of merit for circuit performance and data relating element cost to element tolerance, and specify the set of tolerances that will produce the cheapest manufactured network that has 100 percent *a priori* yield. (By *a priori* yield we mean the percentage of those manufactured circuits all of whose components are within tolerance limits and which are wired correctly that meet specifications. 100 percent *a priori* yield implies that any manufactured circuit that fails to meet specifications has either a component outside of tolerance or a wiring error.) The techniques described are applicable at present only to discrete circuits, i.e., circuits where the parameters are statistically independent. Work is in progress to treat circuits with correlated parameters.

The method described below is a combination of two-dimensional performance contours,[1] a method the author has since learned is a variation on the branch and bound technique,[2] quasi-worst-case computations and repetitive TAP computations.

II. THE CHEAPEST NETWORK

In this paper the cheapest network is defined as the network with 100 percent *a priori* yield such that the sum of the costs of the components comprising that network is not greater than the sum of costs for every other network with 100 percent *a priori* yield. We are concerned with a network of fixed topology and fixed nominal element values for which we want to choose a tolerance vector, τ, each of whose elements, τ_i, is the tolerance chosen for circuit parameter, p_i. Associated with each parameter tolerance, τ_i, is a cost, $C_i(\tau_i)$, where $C_i(\cdot)$ is the cost function of the ith circuit parameter.

For each parameter the designer has a limited number of discrete tolerances from which to choose. Therefore, we will consider a finite universe of possible realizations, each realization being characterized by a unique tolerance vector. We will number the realizations in the universe 1, 2, $\cdots$, l, and refer to each realization by its associated tolerance vector τ^k. We next define the set,

$$\mathcal{A} = \{k \mid \text{realization } k \text{ has 100 percent } a \textit{ priori} \text{ yield}\}$$

and the associated set,

$$\mathcal{A}_\tau = \{\tau^k \mid k \in \mathcal{A}\}.$$

Then a cheapest network has tolerance vector τ^* satisfying the conditions:

(*i*) $\tau^* \in \mathcal{A}_\tau$,

(*ii*) $$\sum_{i=1}^{n} C_i(\tau_i^*) \leqq \sum_{i=1}^{n} C_i(\tau_i^k); \qquad \text{for all } k \in \mathcal{A},$$

where the circuit under study has n components.

One could imagine other criteria by which the cheapest network might be defined. In particular, one might find a network cost function that traded off yield against cost of repairing, component salvaging, or discarding networks. However, consideration of such tradeoffs raises serious questions about manufacturing practices, not to mention the question of allowing sub-marginal circuits to be installed in the field. Because of these questions, the following discussion is limited to circuits for which 100 percent *a priori* yield is required. As we shall see, this assumption is central to the proposed tolerance specification method.

III. REGIONS OF ACCEPTABILITY AND POSSIBILITY

It is assumed that the circuits with which we are dealing have a scalar performance, J, that is a computable function of the circuit parameter values; a nominal design with performance measure, J_0; and an allowable degradation from nominal, ϵ, such that $(J_0 + \epsilon)$ marks a boundary separating acceptable circuits from unacceptable ones.*

We now examine the n-dimensional parameter deviation space, i.e., a space whose origin is defined as the nominal parameter vector and each of whose axes is the percent deviation of one parameter from its nominal value. We postulate the existence of a connected region, including the origin, such that all circuits built with parameter values represented by points inside the region are acceptable and those outside the region are unacceptable. We call this the Region of Acceptability, R_A.[1] We now notice that in this same space, each tolerance vector, τ^k, is associated with an n-dimensional parallelepiped that we call the Region of Possibility.† In other words, all circuits built of components with tolerances specified by τ^k have parameter deviation vectors inside the parallelepiped we are calling the Region of Possibility, R_P.

To clarify these ideas we extend an example presented in Ref. 1. Figure 1a shows a simple voltage divider. Its transfer function is given by $T = 1/(1 + R_1/R_2)$, and its input resistance is $R = R_1 + R_2$. With nominal one-ohm resistors, the nominal transfer function and input resistance are 0.5 and 2.0 respectively. Suppose the design specifications call for $0.46 \leqq T \leqq 0.53$ and $1.85 \leqq R \leqq 2.15$. Then the Region of Acceptability (R_A) shown in Fig. 1b is bounded by four lines, each of which is the locus of all points producing networks exactly satisfying one of the four specification limits. Figure 1b also shows the Region of Possibility (R_P) when the voltage divider is built with 5 percent components. Notice that R_P intersects the parameter deviation axes at ± 5 percent. Notice further that all possible networks are acceptable, i.e., R_P is totally contained within R_A. In Fig. 1c, a new R_P that is the result of using a 10 percent resistor for R_1 and a 3 percent resistor

* See Ref. 1 for examples of transcribing classical loss, phase, frequency specifications into J, ϵ representation.

† Here we are assuming that all components are 100 percent tested, i.e., a batch of 15 percent components has *all* components within ± 15 percent of nominal value. We are further assuming that tolerances place symmetric limits on component values. The asymmetric case can be handled but it makes the argument obtuse. Lastly, we are assuming all components are statistically independent.

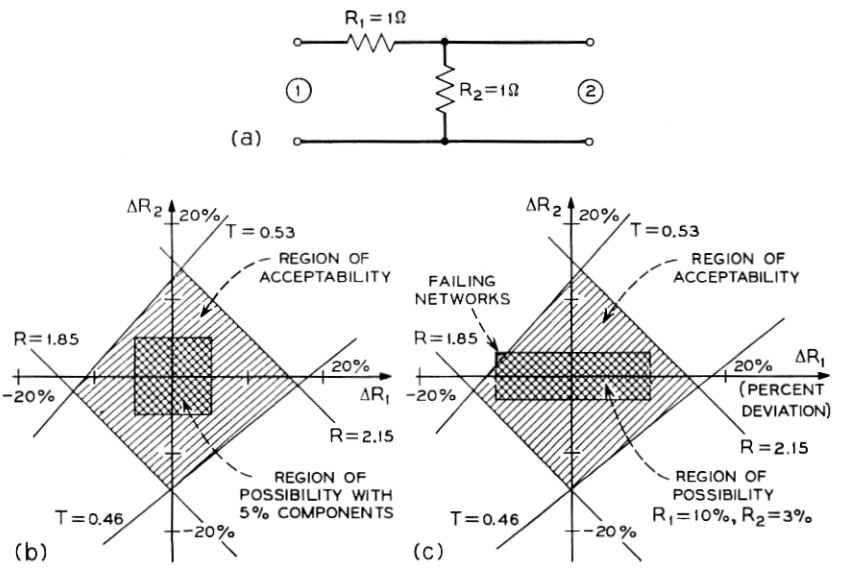


Fig. 1a—Voltage divider example.
Fig. 1b—Regions of acceptability and possibility for voltage divider (5 percent components).
Fig. 1c—Regions of acceptability and possibility for voltage divider (R_1 a 10 percent resistor, R_2 a 3 percent resistor).

for R_2 is superimposed over R_A. Notice that some networks in the second quadrant fail, i.e., they are inside R_P but outside R_A. Hence, the tolerance vector (10, 3) is not a member of $\mathfrak{A}_\tau$ and cannot be used for the example network.

The conclusion we may now reach is that in order for a network to have 100 percent *a priori* yield we must have $R_P \subseteq R_A$. Further, because we are restricting our attention to networks that have 100 percent *a priori* yield, our task is to choose the cheapest tolerance vector that will satisfy the condition, $R_P \subseteq R_A$.

If we could characterize the Region of Acceptability, the task of choosing a tolerance vector that produces the cheapest network would be reduced to the problem of finding the "largest" parallelepiped (R_P) that is contained within R_A. (Notice that our assumptions of 100 percent *a priori* yield and 100 percent component testing obviate the need to consider component distributions. All we are concerned with are the limits of component deviation.) Unfortunately, characterization of R_A is an impractical task; we are trapped by "the curse of dimensionality." Even if R_A were as simple a region as a hypercube, it would, for a net-

work of say 15 parameters, have over 32,000 vertices. The number of function evaluations required to locate these vertices would be exceedingly large. When one considers that the $(n - 1)$ dimensional surfaces bounding R_A are likely to be quite complex, the magnitude of the problem becomes staggering.

The algorithm that is used to select the tolerance vector that will produce the cheapest network, τ^*, begins with a consideration of two-dimensional subspaces of R_A.[1] We will require that two-dimensional subspaces of R_P be inside the corresponding subspaces of R_A. We will then use a series of techniques to insure that $R_P \subseteq R_A$.

IV. FINDING THE CHEAPEST NETWORK

The algorithm for finding a cheapest network is broadly outlined in Fig. 2. Below we take up each of the major blocks in detail.

4.1 *Pairwise Constraints*

The first step in obtaining pairwise constraints on component tolerances is to compute the minimum intercept[1] for each parameter. Call this L_i for the ith parameter. Briefly, this is the minimum deviation each parameter can undergo, keeping all other parameters fixed at nominal value, before the network specifications are violated.

Next we consider the $\binom{n}{2}$ performance contours[1] for the n parameter network. These contours are the boundaries of the two-dimensional subspaces of R_A, holding $(n - 2)$ parameters fixed at nominal. Each contour determines what tolerance pairs are allowable for the variable parameters. In short, each tolerance pair defines a rectangle. The pair is allowable if that rectangle is within the performance contour.

The designer is presumed to have specified a discrete set of obtainable tolerances for each component. Call this set S_i for the ith component. We will number the members of S_i in descending order and refer to them as $\tau_{i1}, \tau_{i2}, \cdots, \tau_{im_i}$, where m_i is the number of obtainable tolerances for the ith component. We will form a table for each pair of parameters, say (r, s), of the form:

p_r	p_s
$\tau_{r,i}$	$\tau_{s,\alpha}$
⋮	
τ_{r,m_r}	$\tau_{s,\gamma}$.

For ease of subsequent computation, the parameter on the left will be the one with the smaller minimum intercept. Not all m_r entries neces-

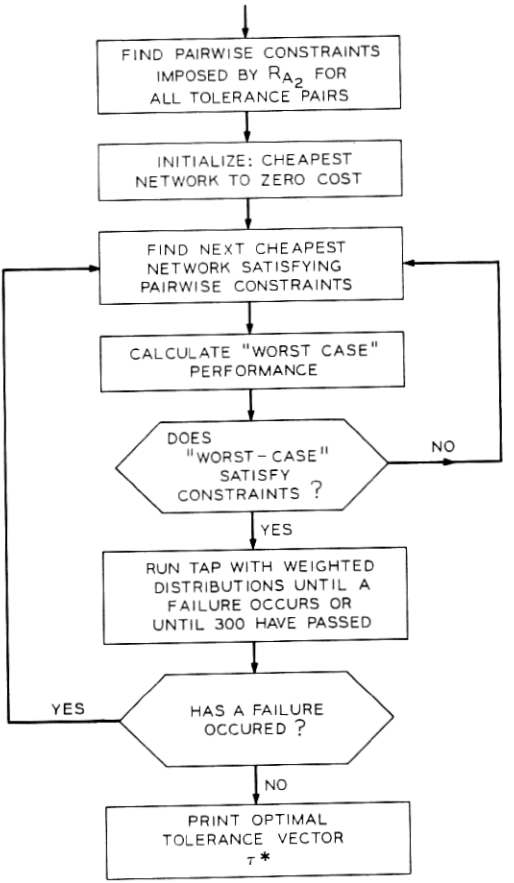


Fig. 2—Tolerance specification algorithm.

sarily exist in the table. First we eliminate those tolerances that are larger than the minimum intercept, since they will clearly produce rectangles not contained within the (r,s) contour $(\Gamma_{r,s})$. Then for the first $\tau_{r,j} \leqq |L_r|$, we find the largest element of S_s that produces a rectangle inside $\Gamma_{r,s}$, and this tolerance is entered into the table opposite $\tau_{r,j}$. If a line of the table reads

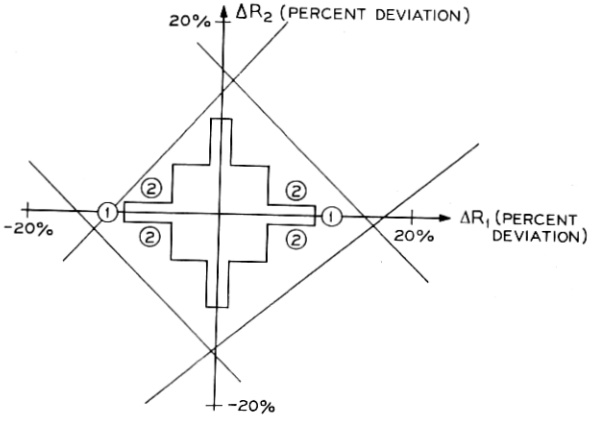
$$\tau_{r,j} \qquad \tau_{s,k}$$

and if parameter r has tolerance $\tau_{r,j}$, parameter s may have $\tau_{s,k}$ or any smaller tolerance. We proceed through the elements of S_r finding the corresponding largest allowable tolerance within S_s until we have entered τ_{r,m_r} and its companion in the table.

An example should clarify these ideas. R_A for the voltage divider of Fig. la is reproduced in Fig. 3. Since the voltage divider has only two parameters, only one performance contour exists and this is identical to R_A. Suppose that R_1 and R_2 have the same set of obtainable tolerances,

$$S_1 = S_2 = \{15, 10, 5, 3, 1\}.$$

Figure 3 indicates $L_1 \simeq -11$ percent $L_2 \simeq 13$ percent. We will therefore consider p_1 to be the left side of the table. To begin to fill the table we notice $\tau_{11} = 15$ percent $> |L_1|$ and is therefore eliminated from consideration. $\tau_{12} = 10$ percent is entered as the first entry in the left-hand column. Starting at the points in Fig. 3 (10 percent, 0) and (−10 percent, 0), the vertical directions are explored to determine how far R_2 may be varied before specifications are exceeded. In the second quadrant, specifications are exceeded before $\Delta R_2 = 2$ percent. Therefore, $\tau_{2,5} = 1$ percent is made the first right-hand entry in the table. Computationally we alternate between $\Delta R_1 = 10$ percent and $\Delta R_1 = -10$ percent and take small steps in ΔR_2 both positively and negatively until the specifications are violated (shown as step 1 on Fig. 3). The horizontal lines of the (10, 1) rectangle are then explored until ΔR_1 has been reduced to $\tau_{1,3} = 5$ percent. Since no specification failures are detected, the entry (10,1) is certified (step 2 in Fig. 3). (Notice we are assuming that performance contours are simply connected.) We then explore vertically from the four points on which step 2 ended and find that the next entry in the table is (5, 5). We continue in this way until



p_1	p_2
10%	1%
5%	5%
3%	5%
1%	10%

Fig. 3—Allowable tolerance rectangles for voltage divider.

the table shown in Fig. 3 is complete, and we have explored the combined perimeter of all allowable tolerance rectangles. Notice that tightening ΔR_1 from 5 percent to 3 percent resulted in no increased allowable tolerance for R_2.

This process is repeated for each of the $\binom{n}{2}$ parameter pairs. The table for each parameter pair (r, s) is denoted $T_{r,s}$. Each table gives us a pairwise constraint on tolerances. The next step in our effort to find τ^* is to find the cheapest network satisfying the pairwise constraints, i.e., to find a tolerance vector, $\tau^\#$, whose elements satisfy the table constraints pairwise and is the cheapest of all such tolerance vectors. Having found $\tau^\#$, we will then try to discover if $\tau^\# \in \mathfrak{A}_\tau$, i.e., if $\tau^\# = \tau^*$. If $\tau^\# \notin \mathfrak{A}_\tau$, we will find the next cheapest tolerance vector satisfying the pairwise constraints, etc., until we find one that is a member of $\mathfrak{A}_\tau$. If we define a set,

$$\mathfrak{B}_\tau = \{\tau \mid \tau \text{ satisfies pairwise constraints}\},$$

then it is clear that $\mathfrak{B}_\tau \supseteq \mathfrak{A}_\tau$ and by choosing elements of $\mathfrak{B}_\tau$ in order of increasing cost, the first chosen element of $\mathfrak{B}_\tau$ that is also an element of $\mathfrak{A}_\tau$ is in fact τ^*.

4.2 *Branch and Bound Technique for Finding the Cheapest Network*

In this section we describe a variation on the branch and bound technique that is used to find the cheapest element of $\mathfrak{B}_\tau$, $\tau^\#$. Consider the multi-root tree structure shown in Fig. 4. The nodes in the first column represent the elements of S_1, the obtainable tolerances for p_1. From each node in the first column we have branches leading to all the elements of S_2, the obtainable tolerances for p_2, etc. Finally, the last column has all the elements of S_n repeated for every choice of the $(n - 1)$ preceding element tolerances. There are $\prod_{i=1}^{n} m_i$ nodes in the last column and an identical number of paths from the left side of the tree to the right. Each path is associated with a choice of tolerance vector and hence has an associated cost, $C = \sum_{i=1}^{n} C_i(\tau_i^k)$. The task of finding $\tau^\#$ is then the task of finding a path through the tree from left to right such that:

(*i*) The tolerance pair associated with every node pair of the path is consistent with the tabled constraints.

(*ii*) C is minimized.

With each node we can associate a partial cost. For a node selected in column k, we have a partial cost $C^k = \sum_{i=1}^{k} C_i(\tau_i)$ where τ_k is the tolerance selected in column k and τ_i, $i < k$, are the tolerances selected for the initial part of the path.

A flowchart for selecting the cheapest path whose associated $\tau \in \mathcal{B}_\tau$ is shown in Fig. 5. In words, we begin at the left and choose for τ_1 the largest (cheapest) tolerance not greater than L_1. We then move across the tree picking the kth element's tolerance using

$$\tau_k = \operatorname*{Min}_{i<k} T_{i,k}(\tau_i),$$

where the function $T_{i,k}(\tau_i)$ picks the right-hand entry opposite τ_i in the table, $T_{i,k}$. After each such selection we compute a partial cost, C^k, for the portion of the path chosen so far. Two occurrences will allow us to eliminate that part of the tree with initial path $\tau_1, \cdots, \tau_{k-1}$, viz.:

(*i*) At column k, one or more $T_{i,k}(\tau_i)$ does not exist. This indicates that any τ with the first $(k-1)$ components, $\tau_1, \cdots, \tau_{k-1}$ is not a member of $\mathcal{B}_\tau$.

(*ii*) The partial cost $C^k \geqq C^T$, a target cost. The target cost, initially set at ∞, is replaced by the cost of the first complete path found, and is updated as cheaper tolerance vectors emerge from the process.

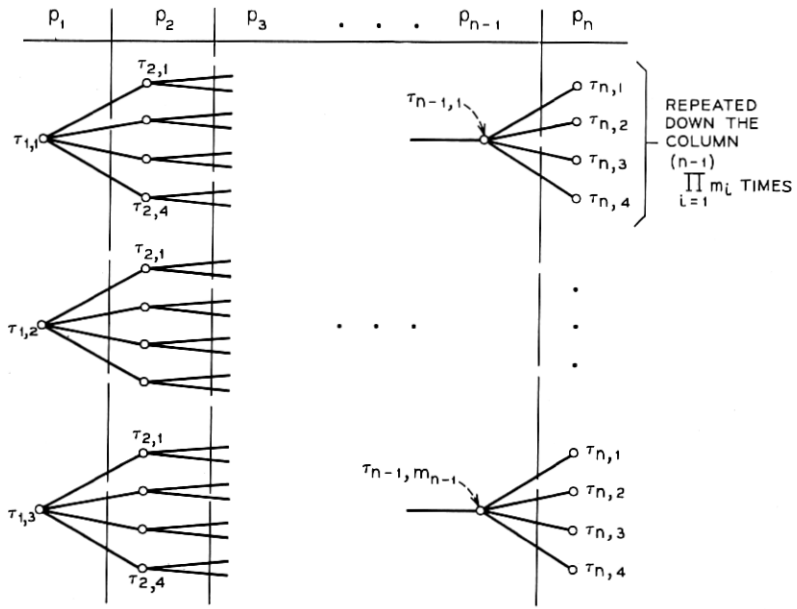


Fig. 4—Tolerance choice tree.

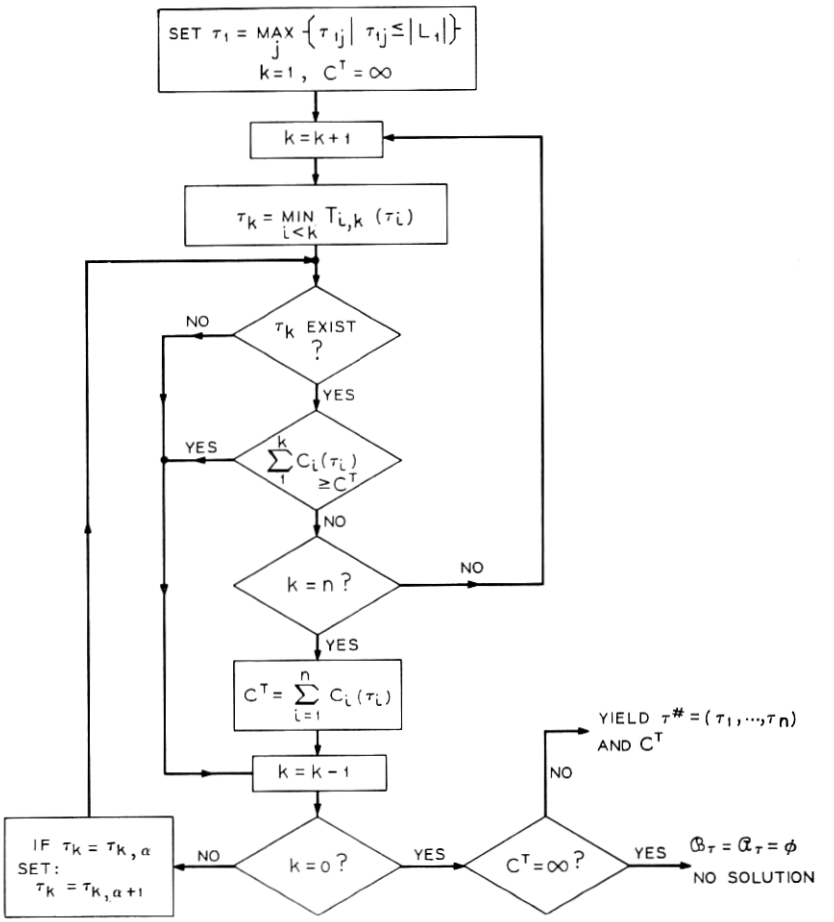


Fig. 5—Algorithm to find $\tau^\#$.

In either case, we return to the $(k-1)$st column and choose the next tighter allowable tolerance. If we are already at the tightest allowable tolerance in the $(k-1)$st column, we return to the $(k-2)$nd column, etc. If neither event occurs, a path reaches the right side of the tree. The associated cost of this path then becomes the target cost. We then back up to the $(n-1)$st column and choose the next tighter tolerance, etc.

The efficiency of the algorithm derives from the fact that the vast majority of paths are never explored to completion, but rather are eliminated by one of the two terminating rules.

The algorithm terminates when all paths have either been explored or terminated. The final target cost is the cost associated with $\tau^{\#}$. Having found $\tau^{\#}$, we must now determine whether $\tau^{\#} \in \mathcal{A}_\tau$.

4.3 *Yield Checks Using Worst-Case and Tolerance Analysis*

A circuit built with the tolerance vector $\tau^{\#}$ determined by the branch and bound optimization has the property that if any two parameters are allowed to vary within their tolerance limits while all other parameters are held constant at their nominal values, design specifications will be satisfied. In other words, this circuit is the cheapest realization that meets all two-at-a-time constraints. Having determined that $\tau^{\#} \in \mathcal{B}_\tau$, we must now verify that $\tau^{\#} \in \mathcal{A}_\tau$, or that the circuit with tolerance vector $\tau^{\#}$ has 100 percent *a priori* yield.

If it happens that $\tau^{\#} \notin \mathcal{A}_\tau$, we will return to the branch and bound algorithm, delete $\tau^{\#}$ from the tree, and find the next cheapest network. We can thus view the branch and bound as a technique for ordering the elements of $\mathcal{B}_\tau$ by cost. At each call to the algorithm we obtain the next element in the nondecreasing cost sequence, $\tau^{\#}, \tau^{\#2}, \cdots, \tau^{\#q}, \cdots$. The first element of this sequence that is an element of $\mathcal{A}_\tau$ is τ^*.

Tolerance vectors that satisfy the two-at-a-time constraints but that have less than 100 percent yield are easy to visualize. Figure 6 is a

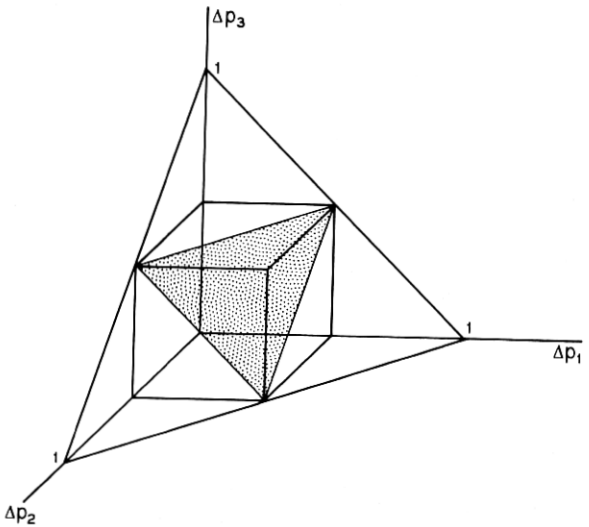


Fig. 6—R_A and R_P for a circuit that meets pairwise constraints but has less than 100 percent *a priori* yield.

three-dimensional perspective of a circuit that meets the two-at-a-time constraints but has less than 100 percent yield. Considering only the positive octant, we see that each of the three performance contours is a triangle. After inscribing rectangles in each triangle, we have a cubic R_P in three dimensions. Now if R_A is a tetrahedron, we see that the shaded plane of intersection of R_A and R_P in Fig. 6 separates the possible circuits into acceptable and unacceptable classes, and despite the fact that the pairwise constraints are met in each of the planes that includes the origin, we have a tetrahedron of failing circuits extending out of the drawing.

Since we have not characterized R_A, we go through two approximate steps to determine if $\tau^* \in \mathcal{A}_\tau$. The first step is a quasi-worst-case computation. We use the signs of L_i to extend each parameter to the limit defined by τ_i. In other words, we compute the circuit's performance with parameters

$$p_i = p_{i0}[1 + \tau_i \operatorname{sgn}(L_i)]$$

where p_{i0} is the nominal value of p_i. If this performance does not meet specifications, we know that the realization with tolerance vector τ^* cannot have 100 percent yield. If the worst-case circuit fails, we eliminate τ^* from the tree, go back to the branch and bound algorithm, and find the next cheapest circuit.

Since even a linear circuit normally has a performance measure that is a nonlinear function of its parameters, the true worst-case performance need not occur at an extreme point of R_P. Hence neither the preceding nor any other classical "worst-case" computation is sufficient to determine if $R_P \subseteq R_A$. The technique that has been chosen is to run Monte Carlo samples of the circuit using the tolerances of τ^* until either a sample fails, in which case we again return to branch and bound for the next cheapest network, or until 300 sample networks have passed specification. The probability distributions used for the Monte Carlo analysis are the triangular shapes shown in Fig. 7. This shape is chosen to emphasize the regions near the parameter extremes; satisfaction of the specifications when the parameters are near nominal has been assured by compliance with the pairwise constraints.

After successful completion of 300 Monte Carlo samples, we claim that $\tau^{*a} \in \mathcal{A}_\tau$. We then replace the distributions of Fig. 7 with the actual distributions to be expected in manufacture and continue to run Monte Carlo samples as a further check. The latter process is continued to whatever extent is desired. It has been found in practice that some failing samples show up if the Monte Carlo analysis is run long enough. In this event we content ourselves with the notion that 99+ percent is an acceptable yield.

V. AN EXAMPLE

The techniques described above were tested on the bandpass filter whose schematic and requirements appear as Figs. 8a and b respectively. In our computation, 30 discrete frequencies were used to define the specification. The network designer's first guess at a tolerance assignment used all 2 percent components. In fact, such a choice does insure 100 percent yield, but it is not the optimum choice.

The set of obtainable tolerances were given as:

$$S_i = \{20, 15, 10, 5, 3, 2, 1, \tfrac{1}{2}\}; \; i = 1, \cdots, 8.$$

Absolute costs, of course, are of no importance to the algorithm; all computations are performed on the basis of relative cost. Therefore, for this example we used

$$C_i(\tau_i) = 1/\tau_i; \; i = 1, \cdots, 8.$$

It is interesting to consider the dimensionality of this example. We have eight parameters, each with eight obtainable tolerances. Hence, we have $8^8 = 16{,}777{,}216$ conceivable tolerance vectors. Many of these vectors can be eliminated by not considering tolerances greater than the minimum intercept of the associated parameters. For example, the inductor labeled p_5 had $L_5 = -5.7$ percent. If this is done, the number of conceivable tolerance vectors falls to just under 3×10^6 —still a rather formidable number.

Two examples of the tolerance-pair tables are shown below:

p_5	p_6
5%	½%
3%	3%
2%	5%
1%	5%
½%	5%

p_1	p_4
20%	½%
15%	5%
10%	10%
5%	15%
3%	15%
2%	15%
1%	20%
½%	20%

The cheapest network that the branch and bound algorithm located, i.e., that satisfied the pairwise constraints, was

$$\tau^* = \{10, 5, 5, 10, 3, 3, 10, 5\},$$

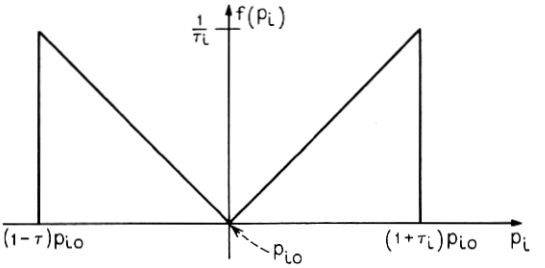


Fig. 7—Probability density function used in Monte Carlo analysis.

with a cost, $C(\tau^*) = 1.57$. The efficiency of branch and bound is demonstrated by the fact that although there are almost 3×10^6 paths through the tree, τ^* is found in 200 ms of computer time on a CDC 3500. To understand this efficiency we note that the number of tolerance vectors satisfying the pairwise constraints is under 700,000. This is 4 percent of the original sixteen million realizations and less than a quarter of those remaining after each S_i is diminished by $\{\tau_i \mid \tau_i > \mid L_i \mid\}$. Perhaps more important to note is that the optimum realization has cost 1.5

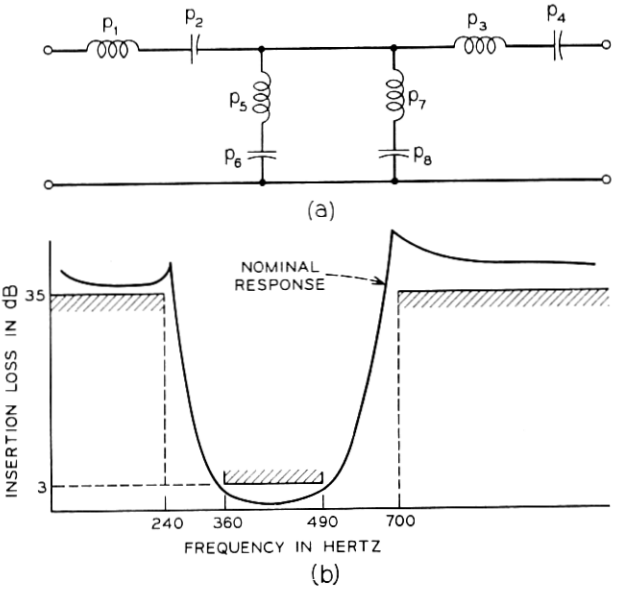


Fig. 8a—Example bandpass filter.
Fig. 8b—Example circuit specifications.

while any path that reaches a node of tolerance $\frac{1}{2}$ percent has a cost greater than 2. Clearly, a great number of paths are eliminated quickly because of the cost constraint.

Examination of the algorithm's operation reveals that τ^* with cost 1.57 failed the "worst-case" test. The branch and bound routine was then re-entered 2167 times to produce tolerance vectors of 25 different costs each of which failed the "worst-case" test. At a cost level of 2.47, 362 tolerance vectors were produced of which three passed worst-case. However, each of these three failed during the TAP analysis. Costs of 2.50, 2.53, and 2.57 were then obtained with a total of 1124 vectors of which 12 passed worst-case but again failed during TAP. Finally, the tenth vector with a cost of 2.60 passed both worst-case, the initial 300 TAP samples and a subsequent 500 TAP samples. The solution vector was:

$$\tau^* = \{3, 3, 5, 3, 2, 2, 5, 5\}.$$

The final cost of 2.60 compares favorably with the designer's first assignment, using 2 percent components, that has a cost of 4.0.

The total time to execute the entire algorithm was 20 minutes on a CDC 3500. There were 8124 circuit evaluations: 121 to compute the L_i, 5659 to form the tables $T_{r,s}$, and 2024 total TAP samples.

VI. CONCLUSION

In this paper we have described an algorithm that will assign network element tolerances in a manner that minimizes network cost. The tolerance for each element is chosen from a list of obtainable values, which, together with associated costs, is supplied by the designer. The set of tolerances chosen by the algorithm satisfies two conditions, namely:

(*i*) All sample networks built with components whose values are within the limits imposed by the tolerances meet circuit performance specifications.

(*ii*) The sum of element costs associated with the chosen tolerances is a minimum.

The number of circuit evaluations required by the algorithm is relatively large and can be expected to grow as the square of the number of elements. An example circuit with eight parameters required just over eight thousand circuit evaluations. The feasibility of the algorithm is thus tied to the efficiency of our circuit analysis techniques. Linear networks, nonlinear static networks,[3] and a limited class of

nonlinear dynamic circuits are presently analyzed in a few seconds or less and hence are acceptable candidates for tolerance prediction. On the other hand, it is not unusual for the time-domain analysis of a nonlinear circuit to require several minutes of large scale computer time. The application of the tolerance prediction algorithm to such circuits must await algorithmic or hardware advances that bring the cost of circuit analysis down by at least an order of magnitude.

The efficiency of the techniques is also influenced by the nature of the Region of Acceptability, R_A, the shape of which is determined by the equations describing the network and by the performance objectives. More specifically, the algorithm depends on the adequacy of the characterization of R_A by the performance contours. For the example circuit we noted that there were approximately 700,000 tolerance vectors satisfying the pairwise constraints. However, only 3663 of these vectors had to be tested before one was found that was a member of $\mathfrak{A}_\tau$, i.e., before a vector guaranteeing 100 percent *a priori* yield was found. We deduce then that the performance contours were, for our purposes, a good characterization of R_A. It is possible to imagine an R_A for which this is not the case. In such a situation, where the number of vectors that are not members of $\mathfrak{A}_\tau$ but which still satisfy the pairwise constraints is very large, the number of circuit evaluations for both "worst-case" and TAP computations can become large enough to render the entire technique unfeasible. Fortunately, our experience indicates that in a large class of circuits, element pairs, e.g., LC and RC products and quotients, are dominant influences on circuit response, and that for this class of circuits the performance contours are an acceptable characterization of R_A.

All of this is to say that application of the tolerance specification algorithm is not cheap, and moreover may become quite expensive in some pathological situations. Hence, the algorithm should only be applied to networks whose anticipated production rate is large. In other words, one must balance out expected savings resulting from optimum tolerance assignment against the computational cost of achieving such an assignment. In any case, the variation of component cost with tolerance selection is seldom an issue for limited production networks.

One final point should be mentioned, namely, the designer's ability to obtain accurate component cost data is essential to the success of the process. The generation of such data has often proved more difficult than it might seem. For example, choice of a component of given tolerance for a large production circuit may in itself cause a pricing

change for that component. Further, when one is dealing with adjustable elements for which an adjustment accuracy is sought from the algorithm, one is faced with obtaining costs on a process before it is implemented. The solution of this problem, of course, lies in the close cooperation between design and production engineering.

It should be clear that the algorithm depends strongly on the assumption that only tolerance choices that allow 100 percent yield are acceptable. This assumption eliminates consideration of component probability distributions and the necessity of performing multi-dimensional integration. It allowed us to use the performance contours, "worst-case," and TAP computations to eliminate large numbers of tolerance vectors and bring the problem within manageable proportions.

The assumption of statistical independence of the components is less vital. In the algorithm as presented above, it meant only that the Regions of Possibility in two dimensions were assumed to be rectangular. Techniques that allow component correlation are subjects of future work.

VII. ACKNOWLEDGMENT

The author is indebted to E. M. Butler whose work formed a basis for much of the above, to Mrs. J. M. Schilling who designed and wrote the tolerance specification programs, and to J. Chernak who suggested the project.

REFERENCES

1. Butler, E. M., "Large Change Sensitivities for Statistical Design," B.S.T.J., this issue, pp. 1209–1224.
2. Lawler, E. L., and Wood, D. E., "Branch and Bound Methods: A Survey," J. Operations Res., *14*, No. 4 (July–August 1966), pp. 699–719.
3. Cermak, I. A., and Kirby, Mrs. D. B., "Nonlinear Circuits and Statistical Design," B.S.T.J., this issue, pp. 1173–1195.