An Index for Operational Flexibility in Chemical Process Design

Part II: Computational Algorithms

Procedures for the numerical computation of an index for operational flexibility in chemical processes are considered. Two types of algorithms are proposed which rely on the assumption that critical points for feasible operation lie at vertices or extreme values of the uncertain parameters. The first algorithm is a direct search procedure that features a heuristic variant to avoid exhaustive enumeration of all vertices. The second algorithm employs an implicit enumeration scheme based on a lower bound for monotonic constraints. These algorithms are applied to several example problems to demonstrate both the use of the flexibility index in process design and the computational efficiency of the algorithms.

R. E. SWANEY and I. E. GROSSMANN

Department of Chemical Engineering
Carnegie-Mellon University
Pittsburgh, PA 15213

SCOPE

In Part I an index of flexibility was proposed to quantitatively characterize the flexibility of chemical processes. As was shown, this index gives a measure of the size of the region of feasible operation in the space of the uncertain parameters. Determination of this index for a design provides bounds for the uncertain parameters within which feasible operation can be guaranteed by proper manipulation of the control variables. Furthermore, identification of the critical points ("worst-case" conditions) which define the performance limits of the design is also provided.

Mathematical formulations were developed in Part I to serve as a basis for computing the index, and their properties were studied. Sufficient conditions for the nature of the constraint functions were established for which the solution is guaranteed to lie at a vertex; i.e., with each parameter assuming an extreme value. Furthermore, a parametric description of the feasible region that is convenient for numerical computation was developed. The corresponding formulation determines the direction of parameter deviations that yield the smallest scaled distance from the nominal parameter point to the constraint boundary.

This second part will deal with the problem of developing efficient computational algorithms to determine the flexibility index. These algorithms assume that critical points lie at vertices. The major challenge lies in how to determine the global solution without having to enumerate and analyze the 2^p vertices, where p is the number of uncertain parameters. The first algorithm that is proposed is based on the direct search of all vertices, and is suitable when the number of parameters is small (say $p \leq 4$). A heuristic variant of this algorithm, which can handle a large number of uncertain parameters effectively, avoids total enumeration of all of the vertices. The second algorithm also accomplishes this goal, but it uses a lower bound within an implicit enumeration scheme that is rigorous for monotonic constraints. It is shown that these algorithms can be extended to detect nonvertex critical points, and to perform sensitivity analyses which indicate the rate of change of the flexibility index with proposed changes in the design. Application of the algorithms is illustrated by evaluating the design flexibility of a refrigeration system, a reactor-recycle-compressor system, and a heat exchanger network.

CONCLUSIONS AND SIGNIFICANCE

Two algorithms for computing an index of flexibility have been presented. The direct search algorithm examines all vertices directly to determine the solution, and features a heuristic variant which allows an upper bound solution to be located in an expedient manner. The implicit enumeration algorithm relies on a computable lower bound for monotonic constraints that avoids an exhaustive search over all vertices. The performance of the algorithms has been illustrated with three example problems. These examples have demonstrated not only the fact that the algorithms can be applied to problems with a large number of uncertain parameters, but also that the algorithms provide useful tools for assessing flexibility in chemical plants, comparing alternative designs, and suggesting appropriate modifications to increase flexibility. The examples have also illustrated some other important points: that the location of critical points may not always be obvious; that nonvertex critical points can occur, though probably infrequently; and that the effect of design decisions (process structure and equipment sizing) on the flexibility of the resulting plant can be substantial.

Finally, an important point concerning the algorithms is that they are based on subproblems which are conventional nonlinear programs. Consequently, implementation of the algorithms to treat full-scale process design problems should be possible with existing flowsheet optimization techniques.

R. E. Swaney is currently with the Department of Chemical Engineering, University of Wisconsin, Madison, WI.

INTRODUCTION

In Part I it was shown that the flexibility index F for a given design d could be expressed as

$$F = \min_{\tilde{\theta} \in \tilde{T}} \delta^*(\tilde{\theta}) \tag{1}$$

$$\tilde{T} = \{\tilde{\theta} \mid -\Delta\theta^- \leq \tilde{\theta} \leq \Delta\theta^+\}$$

where the function $\delta^*(\tilde{\theta})$ is given by the nonlinear program (NLP)

$$\delta^*(\tilde{\theta}) = \max_{\delta,z} \delta \tag{2}$$

$$\text{s.t. } f(d,z,\theta) \leq 0$$

$$\theta = \theta^N + \delta\tilde{\theta}$$

$f(d,z,\theta)$ are the reduced inequality constraints of the process given in terms of the design variables d, the control variables z, and the uncertain prameters θ; $\tilde{\theta}$ is a displacement vector from the nominal point θ^N, and is bounded by the expected deviations $\Delta\theta^+, \Delta\theta^-$; δ is a scaled parameter deviation. The problem expressed in Eq. 1 consists then in selecting the displacement vector $\tilde{\theta}$ for which the maximum scaled deviation $\delta^*(\tilde{\theta})$ to the constraint boundary is the smallest.

The fundamental difficulty in solving Eq. 1 stems from the fact that the function $\delta^*(\tilde{\theta})$ usually has multiple local minima over the set $\tilde{T}$. Although it would not be overly difficult to determine a local minimum for $\delta^*(\tilde{\theta})$, solving for the flexibility index in Eq. 1 demands that the global minimum be determined, which is a much more difficult task. The fact that $\delta^*(\tilde{\theta})$ is defined implicitly and may be nondifferentiable further complicates matters.

Problems similar to Eq. 1 have been addressed before, particularly in the field of electrical engineering circuit design (Brayton et al., 1979; Polak and Sangiovanni, 1979). An approach proposed by Polak (1983) based on outer approximations to the feasible region could in principle be applied to this problem. However, a significant drawback in that algorithm is that it requires the determination of global minima for a sequence of nonlinear programming problems. The algorithms presented in this paper address the problem of global minimization in Eq. 1 directly.

The basic question that arises when seeking the global minimum in Eq. 1 concerns how to search among the infinite set of possible values $\tilde{\theta} \in \tilde{T}$ in order to locate the critical point solution $\tilde{\theta}^*$. Answers to that question generally involve discretization, whereby sets of discrete choices are made which will allow the problem to be tackled through a set of more tractable subproblems. The approach to be presented here employs direct discretization in the $\tilde{\theta}$ space.

The key assumption which will be made is that the solution $\tilde{\theta}^*$ will correspond to one of the vertices of the hyperrectangle $\tilde{T}$. Theorems 2 and 3 in Part I provide sufficient conditions under which a vertex solution can be guaranteed. Moreover, even when those conditions are not satisfied, there is still a strong tendency in engineering problems for the solution of Eq. 1 to lie at a vertex. Under this assumption, searching over the infinite set $\tilde{T}$ may be replaced with a search over the finite set of vertices $\tilde{\theta} \in V$, where $V = \{\tilde{\theta}^k, k \in K_v\}$ and $K_v = \{1, \ldots, 2^p\}$ is the index set of the vertices formed from the p uncertain parameters.

These discrete choices for $\tilde{\theta}$ may be represented as follows by introducing the 0,1 binary variables $y_j, j = 1, \ldots, p$:

$$\tilde{\theta}_j = (-\Delta\theta_j^-)(1 - y_j) + (\Delta\theta_j^+)y_j, \quad j = 1, \ldots, p. \tag{3}$$

Correspondingly, when $y_j = 0$ for a given j, component $\tilde{\theta}_j$ will imply a negative deviation for parameter θ_j, while for $y_j = 1$ a positive deviation will result. By considering $\tilde{\theta} \in V$ where $\tilde{\theta} = \tilde{\theta}(y)$ from Eq. 3, problem 1 reduces to the nonlinear integer program

$$F = \min_{y} \delta^*(\tilde{\theta}(y)), \tag{4}$$

$$y_j = 0 \text{ or } 1, \quad j = 1, \ldots, p$$

Two algorithms for solving Eq. 4 will be presented. The first involves a direct search over the set of vertices, with a heuristic variant that avoids enumeration of all vertices. The second algorithm is based on an implicit enumeration scheme that also avoids total enumeration, yet will still provide rigorous vertex solutions under conditions where the constraint functions are monotonic in θ.

DIRECT SEARCH ALGORITHM

Given the assumption that the solution $\tilde{\theta}^*$ to Eq. 1 may be found within the vertex set $\{\tilde{\theta}^k, k \in K_v\}$, the simplest approach to solving Eq. 1 would be to compute the value of $\delta^*(\tilde{\theta}^k)$ for each vertex direction $\tilde{\theta}^k$, noting that each evaluation $\delta^*(\tilde{\theta}^k)$ involves the solution of the nonlinear programming problem in Eq. 2. The solution would then be given by

$$F = \min_{k \in K_v} \delta^*(\tilde{\theta}^k) \tag{5}$$

This procedure is direct, and for cases where the number of uncertain parameters p is small (say $p \leq 4$), it is probably the best way to solve the problem. The principal shortcoming of this approach becomes evident when the number of uncertain parameters increases. The number of vertices (and thus optimization subproblems) which must be evaluated is given by 2^p; when $p = 10$, 1,024 subproblems are required, whereas for $p = 20$, the number of subproblems grows to 1,048,576. Obviously, complete enumeration over all vertices becomes impractical for problems with a large or even moderate number of uncertain parameters.

In order to handle larger problems within a reasonable effort, it is possible to forego the rigorous examination of every vertex, and instead to employ a heuristic procedure to search for the critical vertex $\tilde{\theta}^*$. A heuristic search procedure is developed below which features the important characteristic that the number of candidate vertices which actually must be examined is usually small.

The heuristic procedure asks the question: Given a current estimate of the solution $\delta^*(\tilde{\theta}^k)$ at candidate vertex $\tilde{\theta}^k$, and given the fact that the value $\delta^*(\tilde{\theta}^k)$ for any vertex k provides an upper bound for the solution $F = \delta^*(\tilde{\theta}^*)$, what change $\tilde{\theta}^k \rightarrow \tilde{\theta}^\ell, \ell \in K_v$, is likely to cause a decrease in the value of $\delta^*(\tilde{\theta})$? Answers to that question are determined based on linearizing the functions $\delta^*(\tilde{\theta})$ and $f(d,z,\theta)$ at the current subproblem solution $\delta^*(\tilde{\theta}^k)$. There are two ways in which the change $\tilde{\theta}^k \rightarrow \tilde{\theta}^\ell$ could cause a decrease in $\delta^*(\tilde{\theta})$, depending on whether or not the set of active constraints in Eq. 2 changes. They will be presented in turn.

Assuming that the active set remains the same, linearizing the function $\delta^*(\tilde{\theta})$ at the current point $\tilde{\theta}^k$ predicts

$$\delta^*(\tilde{\theta}^\ell) \approx \delta^*(\tilde{\theta}^k)[1 + v^T(\tilde{\theta}^\ell - \tilde{\theta}^k)] \tag{6}$$

where v^T represents the gradient in θ-space of the function $\delta^*(\tilde{\theta})$, and is the vector of multipliers $v^T = -\lambda^T(\partial f/\partial\theta^T)$ determined in Eq. 19c of Part I at the solution to the subproblem $\delta^*(\tilde{\theta}^k)$. In order to decrease $\delta^*(\tilde{\theta})$, this linearization suggests the choice of $\tilde{\theta}^\ell$ such that $v^T\tilde{\theta}^\ell < 0$, i.e.

$$\text{sign}(\tilde{\theta}_j^\ell) = -\text{sign}(v_j) \quad j = 1, \ldots, p \tag{7}$$

This procedure corresponds to projecting the gradient v onto the set of vertices. If $\tilde{\theta}^\ell \neq \tilde{\theta}^k$, then there must exist some $\tilde{\theta}'$ in the direction $(-v)$ such that $\delta^*(\tilde{\theta}') \leq \delta^*(\tilde{\theta}^k)$, and vertex $\tilde{\theta}^\ell$ becomes a candidate for examination. However, the situation $\tilde{\theta}^\ell = \tilde{\theta}^k$ may occur, indicating that no local descent direction exists, and that therefore $\tilde{\theta}^k$ is a local minimizer for Eq. 1. In this case a second prediction strategy may be invoked.

Although there may be directions $\tilde{\theta}^\ell$ for which the prediction in Eq. 6 based on a constant active set indicates an increase for $\delta^*(\tilde{\theta})$, in actuality the change $\tilde{\theta}^k \rightarrow \tilde{\theta}^\ell$ could cause the active set in Eq. 2 to change and therefore result in a decrease for $\delta^*(\tilde{\theta})$. Thus we seek a predictor to indicate the likelihood that a constraint f_i which is inactive at the current point $\tilde{\theta}^k$ will become active at a proposed point $\tilde{\theta}^\ell$. Linearizing each constraint for changes in $\tilde{\theta}$ provides

$$\Delta f_i \approx \bar{\delta} \frac{\partial f_i}{\partial \theta^T} (\theta^\ell - \theta^k) \tag{8}$$

where $\bar{\delta}$ is the value for the current upper-bound estimate of the solution $\delta^*(\theta^*)$. Using Eq. 8, the local gradient information $\partial f_i / \partial \theta^T$ may be used to estimate for each constraint f_i the vertex $\theta^{*,i}$ which will cause the greatest increase Δf_i:

$$\theta_j^{*,i} = \begin{cases} \Delta\theta_j^+ & \dfrac{\partial f_i}{\partial \theta_j} \geq 0 \\ -\Delta\theta_j^- & \dfrac{\partial f_i}{\partial \theta_j} < 0 \end{cases} \qquad j = 1, \ldots, p \tag{9}$$

Thus we may define the nonnegative variables

$$w_i = \frac{\partial f_i}{\partial \theta^T} (\theta^{*,i} - \theta^k) \tag{10}$$

and estimate the increases for each constraint by

$$\Delta f_i \approx \bar{\delta} w_i \tag{11}$$

The proposed predictor is simply

$$\rho_i = \frac{\Delta f_i}{-f_i} = \frac{\bar{\delta} w_i}{-f_i} \tag{12}$$

where $(-f_i)$ is the current "slack" for each constraint. The value ρ_i computed for each inactive constraint ($\lambda_i = 0$ in Eq. 2) gives the ratio of estimated possible increase to available slack. Values $\rho_i \gg 1$ suggest that constraint i is likely to be violated at the vertex $\theta^{*,i}$, requiring a change in the active set that may produce a lower value for $\delta^*(\theta)$; values $\rho_i \ll 1$ suggest that this will be unlikely. The prediction strategy is then to compare values ρ_i with a chosen threshold value ρ^{MAX}; vertices $\theta^{*,i}$ corresponding to values $\rho_i \geq \rho^{MAX}$ then become candidates for examination.

The resulting heuristic search procedure based on these predictors performs a limited enumeration of the vertices by at each step examining a new candidate vertex until no new candidates are predicted. The algorithm is:

Step 0. Select starting vertex θ^k; initialize set of examined vertices $V' = \phi$; set upper bound $\bar{\delta} = \infty$.

Step 1. Examine candidate vertex θ^k by solving Eq. 2; set $V' = V' \cup \{\theta^k\}$. If $\delta^*(\theta^k) < \bar{\delta}$, set current solution estimates $\theta^{est} = \theta^k$, $\bar{\delta} = \delta^*(\theta^k)$, and proceed to Step 2. Otherwise, go to Step 3.

Step 2. Project θ-gradient to determine θ^ℓ as in Eq. 7. If $\theta^\ell = \theta^k$, proceed to Step 3. Otherwise, set $\theta^k = \theta^\ell$ and go to Step 1.

Step 3. At the solution to Eq. 2 for the current estimate θ^{est}, compute $\rho_i = \bar{\delta} w_i / -f_i$ for each inactive constraint $f_i < 0$. Select the largest ρ_i such that $\theta^{*,i}$ has not been examined previously; i.e., identify m such that $\rho_m = \max\{\rho_i \mid i \in I, \theta^{*,i} \notin V'\}$. If $\rho_m < \rho^{MAX}$, or no unexamined $\theta^{*,i}$ remain, go to Step 4. Otherwise set $\theta^k = \theta^{*,m}$ and go to Step 1.

Step 4. Stop. Set $\theta^* = \theta^{est}$, $F = \bar{\delta} = \delta^*(\theta^*)$.

The basic idea in this algorithm is that each time the upper bound $\bar{\delta}$ is updated, a linear analysis is performed at the corresponding vertex to identify local descent directions leading to other vertices. When no such directions exist, or when the vertex under examination in Step 1 does not produce a lower value for $\bar{\delta}$, Step 3 is used to identify vertices that may produce a decrease in the bound by requiring a change in the active constraint set. This is accomplished by comparing an estimate of the potential increase to the available slack for each inactive constraint.

Choice of the parameter ρ^{MAX} allows some control over the thoroughness of the search. A value of $\rho^{MAX} < 0$ provides a conservative approach, since in this limit the algorithm becomes equivalent to total vertex enumeration. A value such as $\rho^{MAX} = 1$ will yield a more expedient search, since this threshold value corresponds to the case where the linearized increase Δf_i just equals the available slack $(-f_i)$. The advantage of using a value such as $\rho^{MAX} = 1$ is that the algorithm will usually terminate while the set of examined vertices V' is still small in comparison to V, the entire set of 2^p vertices.

Two minor modifications to the above procedure are recommended. First, the simple constraint $\delta \leq \bar{\delta}$ should be included when solving the subproblems of Eq. 2 in Step 1. The constraint will eliminate any unboundedness, and often will reduce the computational effort required to solve the subproblem for those vertices where $\delta^*(\theta^k) > \bar{\delta}$. Although for those vertices the true value for $\delta^*(\theta^k)$ will not be computed since the simple constraint will become active, this is of no concern because the objective is only to identify vertices for which $\delta^*(\theta^k) < \bar{\delta}$.

The second modification applies to Step 3, and is included simply to improve the power of the predictors ρ_i in cases where there are degrees of freedom remaining in the control variables at the solution to subproblem 2 in Step 1. This situation corresponds to having fewer than $n_z + 1$ active constraints with strictly positive multipliers. A slight variation of the feasibility subproblem of Eq. 14 in Part I may be solved prior to computing the ρ_i:

$$\min_{z,u} u \tag{13}$$
$$\text{s.t. } f(d,z,\theta) - uw \leq 0$$
$$\theta = \theta^N + \delta\theta^k$$

Since the weighting vector w given in Eq. 10 has nonnegative elements, a feasible point is determined, and any remaining degrees of freedom are used to maximize the slacks $(-f_i)$ in proportion to each w_i. This potentially will improve the prediction ability of the ρ_i. Solving Eq. 13 requires a minimal effort when the solution to the Step 1 subproblem is taken as a starting point.

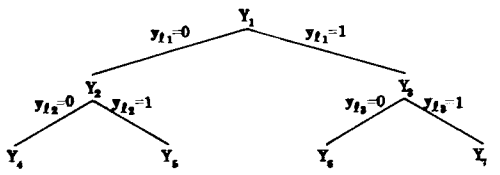
Summarizing, the heuristic procedure provides an efficient means of searching the set of vertices for a solution to Eq. 1, with reasonable likelihood that the solution found will in fact be a global minimizer. Rigorous solution guarantees are not possible with this algorithm, however, since it is based on local linearizations, and it does not take into account the simultaneous interactions of the separate inequalities in Step 3. Thus a vertex which is infeasible at $\bar{\delta}$ could possibly be overlooked. For this reason, the only rigorous result from this procedure is that $\bar{\delta}$ provides a valid upper bound for the flexibility index.

IMPLICIT ENUMERATION ALGORITHM

When using the direct search procedure of the preceding section, exhaustive enumeration of all 2^p vertices will be required to rigorously guarantee that the global minimum to Eq. 1 has been determined. An alternative approach based on implicit enumeration is presented in this section. The object in implicit enumeration is to arrive at the global minimum without explicitly examining every possible solution (i.e., vertex), yet at the same time guaranteeing that the true global solution is not overlooked. The branch-and-bound procedure developed below will accomplish this with rigorous guarantee of the solution when the constraint functions $f(d,z,\theta)$ are monotonic in θ and the solution lies at a vertex.

It will be convenient to consider the 0–1 integer programming formulation of the problem as given in Eq. 4, where each vertex is designated by a unique 0-1 integer vector y. The set of possible solutions, i.e., the set of vertices V, may be represented by the terminal nodes of a binary tree as illustrated in Figure 1. Each node branch corresponds to the choice of either 0 or 1 for a particular component y_j; at the terminal nodes, all components y_j, $j = 1, \ldots, p$, are completely assigned and thus these nodes correspond to the actual individual vertices. Intermediate nodes, on the other hand, correspond to partial assignments of the y_j, i.e., for a given node some of the parameters are assigned fixed values of either 0 or 1, while the remaining parameters are free to assume either value.

The tree representation provides a method of partitioning the vertices into subsets. Each node represents the particular vertex subset whose members all share the same values for those parameters y_j that have been assigned fixed values at previous branches. The branches at each node partition the vertices in that subset into two disjoint subsets based on whether the value 0 or 1 is assumed by the designated parameter y_ℓ being assigned at that node.



$$V = \{\ \bar{\theta}(y) \mid y \in Y\ \}$$

$$Y_2 = \{\ y \mid y \in Y_1,\ y_{t_1}=0\ \};\quad Y_3 = \{\ y \mid y \in Y_1,\ y_{t_1}=1\ \}$$

$$Y_4 = \{\ y \mid y \in Y_2,\ y_{t_2}=0\ \};\quad Y_5 = \{\ y \mid y \in Y_2,\ y_{t_2}=1\ \}$$

$$Y_6 = \{\ y \mid y \in Y_3,\ y_{t_3}=0\ \};\quad Y_7 = \{\ y \mid y \in Y_3,\ y_{t_3}=1\ \}$$

Figure 1. Partial assignment tree for vertex partitioning.

The motivation for this partitioning by stages lies in the possibility of eliminating entire subsets of vertices from consideration during the search for the minimizing vertex. Subset elimination becomes possible when a value L can be computed which will be a valid lower bound for the minimum of $\delta^*[\bar{\theta}(y)]$ over the subset. The essence of the implicit enumeration scheme is to exploit the fact that a vertex subset whose lower bound L exceeds any known vertex solution $\bar{\delta} = \delta^*[\bar{\theta}(\hat{y})]$ cannot contain the global solution, and therefore may be safely ignored. On the other hand, if the entire subset cannot be eliminated in this way, it may be partitioned further. The resulting subsets will themselves have lower bounds which are at least as large as, and hopefully greater than, the bound for the parent subset. By continuing in this manner, the global solution is isolated, eliminating vertex subsets whenever possible.

The key requirement for the above scheme is the availability of a lower bound which is both computable and tight enough to allow effective subset elimination. In Appendix I it is shown that if the problem in Eq. 1 is restated as a min-max problem, a lower bound for the flexibility index can be developed. The resulting lower bound $L(S_m)$ for any vertex subset S_m is obtained by solving the following nonlinear program:

$$L(S_m) = \max_{\delta,z} \delta \tag{14}$$

$$\text{s.t. } f_i(d,z,\theta^{(i)}) \le 0,\ i \in I$$

$$\theta^{(i)} = \theta^N + \delta\bar{\theta}^{*,i},\ i \in I$$

where, under the assumption that the constraints $f_i(d,z,\theta)$ are monotonic in each component of θ, the values of the $\bar{\theta}^{*,i}$ may be determined directly from the signs of the gradients as in Eq. 9, subject to the restriction $\bar{\theta}^{*,i} \in S_m$. Consequently, since the vertex subset S_m for any given node m in the enumeration tree is given by $S_m = \{\bar{\theta} | \bar{\theta}_j$ corresponds to the partial assignment of the binaries y_j for components j assigned at node $m; \bar{\theta}_j = -\Delta\theta_j^-$ or $\Delta\theta_j^+$ for all other $j\}$ the vector $\bar{\theta}^{*,i}$ for each constraint may be obtained by fixing those components specified by the partial assignment at the node, and by setting the remaining components equal to either $-\Delta\theta_j^-$ or $\Delta\theta_j^+$ as indicated by Eq. 9.

The nonlinear program in Eq. 14 provides an efficient means of computing the lower bound $L(S)$ for a chosen vertex subset S, and will serve as the basis for the proposed implicit enumeration algorithm. The advantage of having the $\bar{\theta}^{*,i}$ as constants, which are readily determined by analyzing the constraint gradients, is that the constraints in Eq. 14 are then simple functions which may be evaluated directly.

Branch-and-Bound Procedure

From the min-max inequality, Eq. A5 in Appendix I, it follows that

$$L(V) = \max_z \min_{\bar{\theta} \in V} \delta'(\bar{\theta},z) \le \min_{\bar{\theta} \in V} \max_z \delta'(\bar{\theta},z) = F \tag{15}$$

under the assumption that the solution to Eq. 1 is contained in the set of all vertices $V \subset \bar{T}$. Cases for which Eq. 15 is satisfied as an equality (the saddlepoint condition) permit immediate solution of the index F by computation of $L(V)$. Typically, though, the saddlepoint condition does not hold, and a gap will exist between $L(V)$ and F. (One may note the close analogy here with the dual gap of nonlinear programming.)

A branch-and-bound strategy will be used to close this gap by employing an improved lower bound $\bar{L}$, which will be successively increased during the branching procedure. The basic idea is to progressively partition the vertices into a collection of subsets with individual lower bounds. Since the global lower bound $\bar{L}$ is given by the minimum of the individual bounds, and the individual subset bounds may be increased by further partitioning, the global bound will increase accordingly. In the following, let $L_m = L(S_m)$, where S_m is the set of vertices associated with node m. These vertices correspond to the set of y vectors Y_m (i.e., $S_m = \{\bar{\theta}(y) | y \in Y_m\}$), with the subsets Y_m partitioning the vertices as depicted in Figure 1. For any $S' \subset S$, the following condition

$$\max_z \min_{\bar{\theta} \in S'} \delta'(\bar{\theta},z) \ge \max_z \min_{\bar{\theta} \in S} \delta'(\bar{\theta},z) \tag{16}$$

must hold, since the lefthand side minimization is more constrained than that on the right. As a consequence of Eq. 16, for any branch where vertex set S_{m_0} is partitioned into subsets S_{m_1} and S_{m_2}, the following relation is true for the lower bounds,

$$\min\{L_{m_1},L_{m_2}\} \ge L_{m_0} \tag{17}$$

This property shows that by branching, any bound L_{m_0} may be replaced by $\min\{L_{m_1},L_{m_2}\}$, where L_{m_1} and L_{m_2} are the bounds computed for each descendant. Additionally, the revised bound will quite possibly be higher, in accordance with Eq. 17, implying a monotonic increase in the bounds with successive branching.

Since any individual bound may be replaced with the pair of its descendant bounds, it is possible to maintain a global lower bound $\bar{L}$ of the form

$$\bar{L} = \min_{m \in M} \{L_m\} \tag{18}$$

by starting with $M = \{0\}$, i.e., $\bar{L} = L_0 = L(V)$, and successively replacing selected bounds in the set with their descendant pairs. The index set M represents the set of open nodes in the tree at any stage. In the enumeration strategy employed below, at each stage the least lower bound L_{m_0}, $m_0 \in M$, is selected. The two bounds L_{m_1} and L_{m_2} corresponding to branching at node m_0 are computed and used to replace L_{m_0} in the bound set. This process is repeated until no further increases in $\bar{L}$ are possible, at which point the condition $\bar{L} = F$ will hold and the solution has been reached. Using this priority strategy, the only additional requirement is a decision rule for choosing the particular binary variable y_ℓ to branch upon at each stage.

By utilizing the multipliers $\hat{\lambda}$ for the inequality constraints in Eq. 14 obtained at the solution for node m_0, the following linearizations may be used to predict the potential increases $L_{m_0} \to L_{m_1}$ and $L_{m_0} \to L_{m_2}$ obtainable by branching on variable y_j:

$$\gamma_j^+ = \sum_{i \in I} \hat{\lambda}_i \frac{\partial f_i}{\partial \theta_j}(\Delta\theta_j^+ - \bar{\theta}_j^{*,i}) \tag{19}$$

$$\gamma_j^- = \sum_{i \in I} \hat{\lambda}_i \frac{\partial f_i}{\partial \theta_j}(-\Delta\theta_j^- - \bar{\theta}_j^{*,i}) \tag{20}$$

Defining

$$\gamma_j = \min\{\gamma_j^+,\gamma_j^-\} \tag{21}$$

the resulting selection rule is then to choose y_ℓ such that

$$\gamma_\ell = \max_j \gamma_j \tag{22}$$

With this choice, the branch at node m_0 will partition the vertex subset S_{m_0} in a manner which (one hopes) will allow $\bar{L}$ to increase as rapidly as possible. When $\gamma_\ell = 0$, no available increase in $\bar{\delta}$ is apparent, so this may be used as a stopping criterion. Based on the above, the algorithm may be stated as follows:

Step 0. Start with $M = \{0\}$, $S_0 = V$. Determine $\bar{\theta}^{*,i} \in V$ for $i \in$

I from the signs of the constraint gradients and solve Eq. 14 for $L_0 = L(V)$.

Step 1. Set $\overline{L} = \min_{m \in M}\{L_m\}$.

Step 2. Determine node m_0 such that $L_{m_0} = \min_{m \in M}\{L_m\}$. (In case of ties, choose one of the deepest nodes.)

Step 3. Compute γ_j^+ and γ_j^-, $j = 1, \ldots, p$, from Eqs. 19 and 20 and determine y_ℓ and γ_ℓ as in Eq. 22 to determine the branching variable.

Step 4. If $\gamma_\ell = 0$, stop ($F = \overline{L}$). Otherwise proceed to Step 5.

Step 5. Branch at node m_0:

5.1) Partition S_{m_0} into S_{m_1} and S_{m_2} based on y_ℓ.

5.2) Determine $\tilde{\theta}^{*,i} \in S_{m_1}$ and solve Eq. 14 for L_{m_1}.

5.3) Determine $\tilde{\theta}^{*,i} \in S_{m_2}$ and solve Eq. 14 for L_{m_2}.

5.4) In set M, replace m_0 with m_1 and m_2. Go to Step 1.

The algorithm shown above is equivalent to a branch-and-bound procedure where the node expanded at each stage is the one having the lowest lower bound (a priority strategy; see Murty, 1976). It is thus neither a depth-first nor a breadth-first tree enumeration, but one that avoids solving those node subproblems with lower bounds that are greater than the optimal solution. Although an upper bound is not used explicitly, the bounding is implicit at termination due to the condition $F = \overline{L}$. From Eq. 18, the lower bounds for all nodes that are open at the stopping point must be at least as great as F. Therefore, all open nodes are eliminated at termination using F as an upper bound.

Three practical considerations should be mentioned. The first concerns the stopping criterion in Step 4. Frequently, constraint sparsity in $\partial f/\partial \theta^T$ will allow the algorithm to terminate with a vertex subset S that contains more than a single vertex. Normally this is quite acceptable, and there will exist a critical vertex $\tilde{\theta}^* \in S$ such that $\delta^*(\tilde{\theta}^*) = L(S)$. However, for special cases with linearly dependent gradients $\partial f_i/\partial \theta^T$, it can result that $\gamma^\ell = 0$ when in fact $\delta^*(\tilde{\theta}^*) > L(S)$. This behavior may be detected and remedied as follows. Assuming normal behavior, a critical vertex $\tilde{\theta}^{\text{est}}$ may be identified within the subset S by taking

$$\tilde{\theta}^{\text{est}} = \begin{cases} \Delta\theta_j^+, \gamma_j^+ = 0 \\ -\Delta\theta_j^-, \gamma_j^- = 0 \end{cases} \quad j = 1, \ldots, p \tag{23}$$

After computing $\delta^*(\tilde{\theta}^{\text{est}})$, if $\delta^*(\tilde{\theta}^{\text{est}}) = L(S)$ then $\tilde{\theta}^{\text{est}} = \tilde{\theta}^*$ and the solution has been found. If a gap remains ($\delta^*(\tilde{\theta}^{\text{est}}) > \overline{L}$), branching must continue. Subsequent branching parameters y_ℓ should be chosen such that

$$\sum_{i \in I} \hat{\lambda}_i \left| \frac{\partial f_i}{\partial \theta_\ell} \right| > 0 \tag{24}$$

until the linearly-dependent cancellation is broken, whereupon either γ_ℓ will again be nonzero, or no available branch y_ℓ can be chosen to satisfy Eq. 24. In the unlikely event that $\partial f_i/\partial \theta_\ell = 0$ for $\hat{\lambda}_i \neq 0$ and a gap still remains, arbitrary choice of the branching parameter y_ℓ may be employed until fathoming occurs.

The second consideration applies if the algorithm is to be used on problems with constraints which in fact are not monotonic in θ. Under these circumstances, the validity of the lower bound or the global solution property cannot be rigorously guaranteed, but the algorithm may still be employed to obtain a solution which is likely to be global. This requires supplementation of Eq. 22 as follows:

$$\gamma_\ell = \begin{cases} \max\limits_j \gamma_j \text{ if } \gamma_j \geq 0 \; \forall j \\ \min\limits_j \gamma_j \text{ otherwise} \end{cases} \tag{25}$$

An occurrence of $\gamma_\ell < 0$ (potential decrease of the lower bound) implies nonmonotonic behavior, and this branching rule attempts to restore the validity of the lower bound in such cases.

A final practicality concerns computational requirements for solution of the lower bound in Eq. 14. It is desirable to employ the smallest possible number of distinct vertices $\tilde{\theta}^{*,i}$, since each distinct vertex implies solution for a separate set of state variables x in order to obtain the reduced functions $f_i(d,z,\theta^{(i)})$. In principle it could be possible for each individual constraint i to require a different vertex $\tilde{\theta}^{*,i}$. However, advantage may be taken of both sparsity and natural coincidences in the gradients $\partial f_i/\partial \theta^T$ so as to merge the possible choices for the $\tilde{\theta}^{*,i}$ and arrive at a minimal distinct set. Valid choices for the individual vertices $\tilde{\theta}^{*,i}$ may be taken from within that set, thus minimizing the required computing effort. A set-covering procedure for determination of the merged vertex set may be found in Grossmann and Sargent (1978). Often, though, the desired set may be deduced by inspection.

EXTENSIONS

There are two extensions worthy of note which may be employed after a vertex solution is determined using either of the two algorithms presented above. The first concerns problems for which the true critical point does not lie along a vertex direction. It is frequently possible to detect the presence of a nonextreme critical point (if one exists) by examining the θ-space gradient of $\delta^*(\tilde{\theta})$ at the solution vertex $\tilde{\theta}^{\text{est}}$ determined by the vertex search. If the gradient projection as in Eq. 7 indicates a direction of descent at $\tilde{\theta}^{\text{est}}$, then there exists a (nonvertex) direction $\tilde{\theta}$ in the vicinity of $\tilde{\theta}^{\text{est}}$ with a lower value of $\delta^*(\tilde{\theta})$. Upon detecting such a condition, the solution $\tilde{\theta}^*$ could be located by computing the local minimum for $\delta^*(\tilde{\theta})$ in the vicinity of $\tilde{\theta}^{\text{est}}$. One possible procedure for this is presented in Appendix II.

The second extension addresses the question of how the flexibility index, once determined, will be affected by changes in the design variables d. The sensitivities of F with respect to changes in each design parameter d_j could be used as guidelines to modify a design to increase its flexibility. Sensitivity coefficients may be obtained through conventional multiplier analysis at the solutions to the critical point subproblems. For a single critical point, when strict complementary slackness holds with unique multipliers λ at the solution of Eq. 2 or 14 (i.e., nonzero multipliers for all active constraints), the sensitivity coefficients σ_j may be computed from

$$\sigma_j = \frac{\partial F}{\partial (d_j)} = -\lambda^T \frac{\partial f}{\partial (d_j)} \tag{26}$$

If zero multipliers result for any of the active constraints, the sensitivity values may depend on whether the designated design variable d_j will be increased or decreased. In such cases, subgradient analysis may be employed and the desired sensitivities computed by solving the following linear programs (LP's):

$$\sigma_j^{\pm} = \max_{p_\delta, p_z} p_\delta$$
$$\text{s.t. } \frac{\partial f_i}{\partial (\delta)} p_\delta + \frac{\partial f_i}{\partial z^T} p_z \pm \frac{\partial f_i}{\partial (d_j)} \leq 0 \;\; \forall_i \text{ s.t. } f_i = 0 \tag{27}$$

where σ_j^+ and σ_j^- correspond to increasing and decreasing d_j respectively.

For problems with multiple distinct critical points, the overall sensitivities σ_j^+, σ_j^- may be obtained from the values determined separately for each critical point:

$$\sigma_j^+ = \min\{(\alpha_j^+)^k\}; \; \alpha_j^- = \min\{(\alpha_j^+)^k\}$$
$$k \in \{\text{distinct critical points}\} \tag{28}$$

Here the term "distinct critical point" is employed simply to avoid the unnecessary consideration of the duplicate critical points which frequently accompany sparseness in the active constraint set. Distinct critical points represent distinct operating limitations, and will feature distinct active constraint sets (excepting some special cases that contain nonmonotonic constraints).

NUMERICAL EXAMPLES

Four process examples are presented here to illustrate the computational algorithms developed in this paper. These examples, chosen to exhibit a variety of common process operating characteristics, will be described first along with the flexibility analysis results. Discussion and comparison of algorithm performance follow in a separate section.

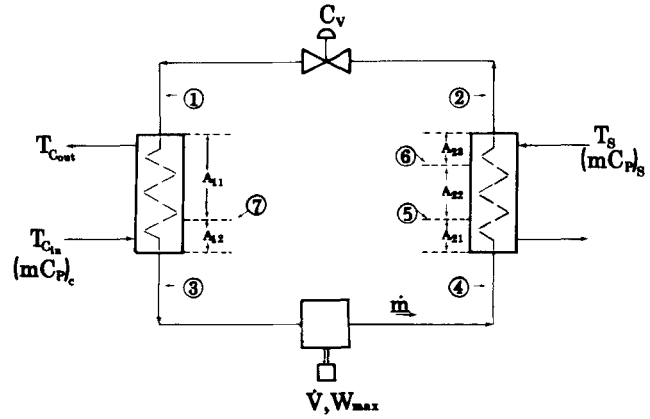


Figure 2. Refrigeration cycle example (REEF).

Refrigeration Cycle (REEF)

This system consists of the simplified refrigeration cycle depicted in Figure 2. The simple expansion valve arrangement with a positive-displacement compressor provides the processing stream with cooling from T_{cin} to T_{cout}, discharging heat to a sink medium entering at T_s. The evaporator and condenser exchangers are modeled in two and three sections, respectively, to account for the pronounced nonlinearities in the refrigerant temperature-enthalpy relationship accompanying the phase changes. Points 1–7 in Figure 2 locate the various thermodynamic states of refrigerant. The system of 57 equations and corresponding state variables describing process behavior may be found in Swaney (1983).

Three parameters are taken as uncertain: T_s, the sink medium inlet temperature, and T_{cin} and $(mC_p)_c$, the inlet temperature and heat capacity-flowrate for the stream to be cooled. Nominal values and expected deviations for these parameters are shown in Table 1. Control is provided by varying the expansion valve opening, C_v.

The performance constraint inequalities are as follows:

$$\begin{aligned}
&1: \ W - W_{max} \le 0 \\
&2: \ T_{cout} - 286\ \text{K} \le 0 \\
&3: \ 280\ \text{K} - T_{cout} \le 0 \\
&4: \ -T_1 + 274\ \text{K} \le 0 \\
&5: \ t_3^L \le 0
\end{aligned} \tag{29}$$

Constraint 1 imposes the compressor driver power limit, while constraints 2 and 3 bound the allowable delivery temperature of the processing stream. Constraint 4 is included to prevent frost formation at the inlet to the evaporator, and constraint 5 disallows the presence of any liquid at the compressor suction. (Variable t_3^L, when positive, denotes the liquid fraction at point 3.)

The set of design variables consists of the two exchanger areas A_1 and A_2, compressor volumetric capacity $\dot{V}$ and driver size W_{max}, the heat sink flowrate $(mC_p)_s$, and the total mass of refrigerant charge in the system m_{tot}. Values were chosen (see Table 1) to meet the nominal operation with no specific overdesign other than a slight oversizing of the compressor drive. The resulting flexibility index is $F = 0.274$, which implies the following parameter ranges for guaranteed feasibility: 301.6 K $\le T_s \le$ 304.4 K, 294.9 K $\le T_{cin} \le$ 297.1 K, and 0.95 kW/K $\le (mC_p)_c \le$ 1.06 kW/K. The critical point corresponds to conditions of high heat sink temperature (304.4 K) with high inlet temperature (297.1 K) and flowrate (1.06 kW/K) for the processing stream. Table 4 lists the limiting constraints which become active at the critical point for this refrigeration system. Thus, as can be seen with this example, determination of the index of flexibility provides a number of useful pieces of information to assess the performance of a design.

TABLE 1. DATA FOR REFRIGERATION CYCLE EXAMPLE REEF
Nominal Parameters and Expected Deviations

j	θ	θ^N	$\Delta\theta^+$	$-\Delta\theta^-$
1	T_s, K	303	+5	−5
2	T_{cin}, K	296	+4	−4
3	$(mC_p)_c$, kW/K	1.0	+0.2	−0.2

Design Variables

j	d	d^1
1	A_1, m^2	3.43
2	A_2, m^2	7.49
3	$\dot{V}$, m^3/s	3.07×10^{-3}
4	W_{max}, kW	3.5
5	$(mC_p)_s$, kW/K	1.0
6	m_{tot}, kg	4.60

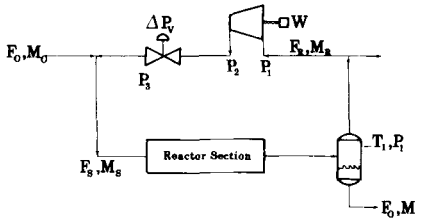


Figure 3. Reactor-recycle system (COMP).

Reactor-Recycle System (COMP)

Figure 3 depicts this simplified version of a common reactor-recycle compressor process rearrangement. For the purposes of this example, the reactor section is treated simply as a pressure drop, but with variations in by-product formation (e.g., due to catalyst aging) that introduce significant variations in the recycle gas molecular weight M_R. The set of 11 equations and state variables chosen to describe this system may be found in Swaney (1983).

The set of uncertain parameters adopted for this example are F_O, the feed throughput; M_O, the molecular weight of the feed; M_R, the recycle gas molecular weight; and K_f, the pressure-drop resistance of the reactor section. Their nominal values and expected ranges are given in Table 2. Control of the recycle flow is provided by adjusting the pressure drop ΔP_v of the throttling valve at the compressor discharge.

The constraints defining acceptable operation are as follows:

$$\begin{aligned}
&1: \ -\Delta P_v + 34\ \text{kPa} \le 0 \\
&2: \ W - \hat{W} \le 0 \\
&3: \ \tfrac{1}{2}Q^* - Q \le 0 \ (\tfrac{1}{2}Q^* \approx Q_{surge}) \\
&4: \ 235\,F_O - (31.4 - M_R)F_R \le 0 \\
&5: \ (31.4 - M_R)F_R - 294\,F_O \le 0
\end{aligned} \tag{30}$$

Constraint 1 reflects the full-open limit of the control valve, constraint 2 the compressor drive power limit. Constraint 3 *is a result* of the surge limit restriction for the compressor (Figure 4), while constraints 4 and 5 place required stoichiometric bounds on the recycle ratio.

To a major extent the operational behavior of this system is a consequence of the operating characteristics of the centrifugal recycle compressor. Figure 4 illustrates typical developed-head versus volumetric-throughput relationships which govern the

TABLE 2. DATA FOR REACTOR-RECYCLE SYSTEM COMP
Nominal Parameters and Expected Deviations

j	θ	θ^N	$\Delta\theta^+$	$-\Delta\theta^-$
1	F_O, kmol/s	0.45	+0.05	−0.08
2	M_O, kg/kmol	125	+10	−10
3	M_R, kg/kmol	5	+6	−1
4	K_f	1.2×10^4	$+0.1 \times 10^4$	-0.2×10^4

Design Variables

j	d	d^1	d^2
1	$\tilde{W}$, kW	15,400	21,060
2	H^*, kJ/kg	192.8	303.1
3	Q^*, m^3/s	4.432	3.460

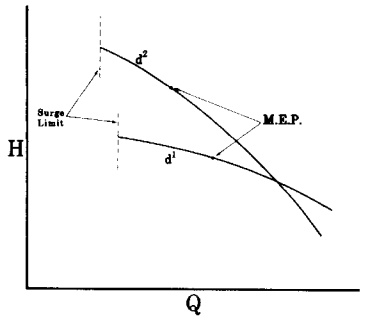


Figure 4. Recycle compressor head vs. volume relations.

performance of a fixed design. A particular compressor design implies a given choice for the head H^* and throughput Q^* corresponding to the maximum efficiency point on the characteristic curve. In this example, therefore, the design variables are taken as H^* and Q^*, and the compressor drive power limit $\hat{W}$. Two different designs are considered, d^1 and d^2 as described below. The two corresponding compressor characteristics may be compared in Figure 4.

For design d^1 (see Table 2), Q^* was assigned to the nominal throughput, while H^* and $\hat{W}$ were selected to accommodate the expected deviations for the operation corresponding to maximum feed rate, feed molecular weight, recycle molecular weight, and reactor pressure drop. This operation might intuitively appear to be the "worst case," since it presents the highest loop pressure drop and maximum throughput for the compressor. However, the flexibility index achieved by design d^1 has a value of only $F = 0.366$. The critical point for this design in fact lies at the condition of lowest recycle molecular weight M_R (see Table 4). This behavior results because, even though the loop pressure drop does decrease with lower M_R, the compressor outlet pressure decreases more rapidly (since P_2/P_1 is nearly proportional to M_R). Consequently, the recycle flowrate needed for the desired stoichiometry cannot be maintained.

The second design, d^2, was selected to accommodate the expected deviations for both the high and low cases of M_R. The computed flexibility index for d^2 assuming a vertex solution is $F = 1.00$, with three critical points covering both high and low cases M_R as listed in Table 4. However, employing the test in Eq. 7 and the procedure in Appendix II reveals the presence of a nonvertex critical point in the vicinity of the vertex solutions yielding a flexibility index of $F = 0.962$.

This nonvertex critical point exists because the compressor power requirement $W = (M_R/\eta_p)F_R H$ goes through a maximum as throughput is varied over the operating range, and the driver power capacity becomes limiting. This maximum is due to the fact that as the recycle flow F_R increases with higher feed rate F_O, the head H in the compressor decreases. Thus, the critical point for design d^2 does not occur at either of the predicted extremes (0.373, 0.498) for the feed rate (with $F = 0.962$), but it instead occurs at an intermediate value of $F_O = 0.399$ kmol/s. It may be noted that, at least for this example, the error between the true value for F and the value computed assuming a vertex solution is not large.

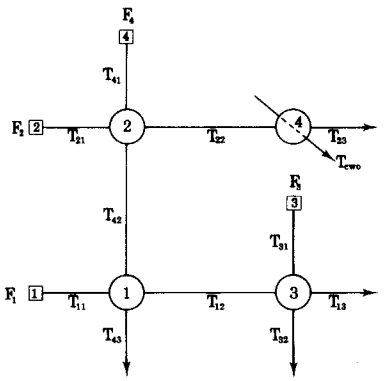


Figure 5. Example HX1.

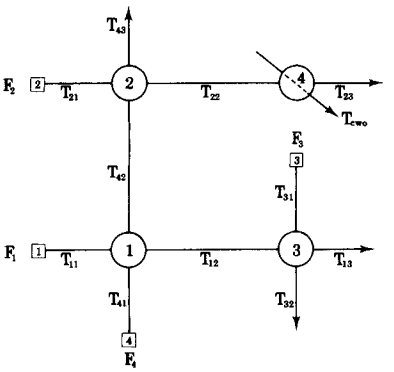


Figure 6. Example HX2.

Heat Exchanger Networks (HX1, HX2)

These two examples consist of the alternative heat exchanger networks shown in Figures 5 and 6. The problem might correspond to a redesign or retrofit situation, with four existing exchangers of given areas being available (see Table 3), and with specified heat exchange requirements that need to be satisfied for each of the four process streams. HX1 and HX2 represent two possible network configurations to be considered.

The four process streams are treated with constant heat capacities. Variable bypass capability is assumed for each exchanger and the outlet temperatures are specified by inequalities. The corresponding system of 28 equalities and state variables is given in Swaney (1983). The individual heat transfer coefficients are treated as functions of the stream flowrates:

$$\begin{aligned} R_1 &= 0.184\,F_4^{-0.8} + 0.167\,F_1^{-0.6} + r_{f1} \\ R_2 &= 0.184\,F_4^{-0.8} + 0.744\,F_2^{-0.6} + r_{f2} \\ R_3 &= 0.106\,F_3^{-0.8} + 0.167\,F_1^{-0.6} + r_{f3} \\ R_4 &= 1.114\,F_{cw}^{-0.8} + 0.744\,F_2^{0.6} + r_{f4} \end{aligned} \tag{31}$$

where $R = U^{-1}$ is the inverse of the overall transfer coefficient and r_f is the fouling factor.

Both the inlet temperatures and flowrates for each of the four streams are treated as uncertain parameters, along with the individual fouling resistances in the exchangers. Table 3 gives the nominal values and expected deviations for each of the twelve uncertain parameters.

TABLE 3. DATA FOR HEAT EXCHANGER NETWORKS HX1 and HX2
Nominal Parameters and Expected Deviations

j	θ	θ^N	$\Delta\theta$	$-\Delta\theta^-$
1	F_1, kW/K	10	+1	−1
2	F_2, kW/K	20	+2	−2
3	F_3, kW/K	30	+3	−8
4	F_4, kW/K	20	+2	−2
5	T_{11}, K	583	+5	−5
6	T_{21}, K	723	+5	−5
7	T_{31}, K	313	+5	−5
8	T_{41}, K	388	+5	−5
9	r_{f1}, m²·K/kW	0.5	+0.5	−0.5
10	r_{f2}, m²·K/kW	0.5	+0.5	−0.5
11	r_{f3}, m²·K/kW	0.5	+0.5	−0.5
12	r_{f4}, m²·K/kW	0.5	+0.5	−0.5

Design Variables

j	d	d^1
1	A_1, m²	40
2	A_2, m²	63
3	A_3, m²	120
4	A_4, m²	12
5	F_{cw}^{MAX}, kW/K	150

TABLE 4. SOLUTION VALUES AND CRITICAL POINT LOCATIONS

Example	Design	F	Active Constraints	Uncertain Parameters θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7	θ_8	θ_9	θ_{10}	θ_{11}	θ_{12}
REEF	d^1	0.274	1,5	H	H	H									
COMP	d^1	0.366	1,4	H	H	L	H								
	d^2	1.00	2,4	L	–	H	–								
			1,4	H	H	L	H								
			1,4	H	H	H	H								
		0.962	2,5	*	–	H	–								
HX1	d^1	0.060	1,9,11,12	H	–	L	H	L	–	H	–	H	–	–	–
HX2	d^1	0.816	1,5,9	H	–	L	L	H	–	H	–	H	–	H	–

H = High value, positive deviation
L = Low value, negative deviation
– = Value does not affect critical point
* = Intermediate value due to non-vertex critical point

The inequality constraints defining feasibility are as follows:

$$
\begin{aligned}
&1: R_1 z_1 - A_1 \le 0\\
&2: -z_1 \le 0\\
&3: R_2 z_2 - A_2 \le 0\\
&4: -z_2 \le 0\\
&5: R_3 z_3 - A_3 \le 0\\
&6: -z_3 \le 0\\
&7: R_4 z_4 - A_4 \le 0\\
&8: -z_4 \le 0\\
&9: T_{13} - 323\ \text{K} \le 0\\
&10: T_{23} - 553\ \text{K} \le 0\\
&11: T_{32} - 393\ \text{K} \le 0\\
&12: 563\ \text{K} - T_{43} \le 0\\
&13: T_{cwo} - 319\ \text{K} \le 0\\
&14: F_{cw} - F_{cw}^{\text{MAX}} \le 0
\end{aligned}
\tag{32}
$$

Constraints 1–8 reflect the bypassing limits such that the control variables z_1, z_2, z_3, and z_4, representing the effective conduction (UA) for each exchanger, may vary only between 0 and the upper limits imposed by the fixed exchanger areas. Constraints 9–12 are single inequalities reflecting the target temperature requirements for each process stream. Constraint 13 limits the cooling water exit temperature, while constraint 14 corresponds to the maximum cooling water flow capacity in exchanger 4.

With the exchanger sizes given in Table 3, both HX1 and HX2 will satisfy the nominal point operating condition. However, the flexibility index for configuration HX1 is a meager $F = 0.060$. The HX2 configuration, on the other hand, yields a flexibility index of $F = 0.816$ using the existing exchangers, and is obviously to be preferred. This substantial difference reflects the fact that in configuration HX1 the temperature crossover condition is more easily approached in exchanger 1 due to its smaller temperature driving force. The critical point operations for each case are given in Table 4.

In addition, the sensitivity analysis as in Eq. 26 may be used to indicate where additional exchanger area should be added in order to increase flexibility. For HX2, only A_1 and A_3 have nonzero sensitivities, with values of 0.003986/m^2 and 0.008296/m^2 respectively. The answer is therefore to add the extra area to either exchanger 1 or 3, or to both. The relative sensitivities suggest that adding area to exchanger 3 will be more effective. This is born out by more detailed analysis, which indicates that the desired flexibilty $F = 1.0$ will require a value for A_1 of 144 m^2 (an area increase of 104 m^2), whereas the same flexibility may be obtained with a value for A_3 of 146 m^2 (an area increase of only 26 m^2). The sensitivity analysis thus provides a quantitative indication of potential flexibility increases, as well as indicating the required direction(s) for expansion for bottlenecking.

Algorithm Performance

Solutions to the nonlinear programming subproblems were computed using a reduced-space variant (Edahl et al., 1983) of the successive quadratic programming approach developed by Wilson (1963), Han (1977), and Powell (1977). This approach determines the optimal values for the decision variables δ,z while simultaneously solving the system equations $h(\delta,z,x) = 0$ for the state variables x. Analytical derivatives were employed; the quadratic programming solutions were determined using SOL/QPSOL (Gill et al., 1982), while operations involving the sparse Jacobian $\partial h/\partial x^T$ were performed using MA28 (Harwell, 1977). A summary of this procedure is presented in Swaney (1983).

Table 5a lists the computation statistics for the direct vertex search algorithms, where each subproblem corresponds to solving

TABLE 5. ALGORITHM PERFORMANCE

(*a*) *Direct Search Algorithms*

Example	Design	No. of Variables θ	z	x	Total Enumeration Subproblems	Total Enumeration CPU seconds*	Heuristic Search Subproblems	Heuristic Search CPU seconds*
REEF	d^1	3	1	57	8	43.5	2	11.6
COMP	d^1	4	1	11	16	27.5	4	6.6
	d^2				16	35.4	4	10.8
HX1	d^1	12	5	28	4,096	–	2	10.0
HX2	d^1	12	5	28	4,096	–	3	23.4

(*b*) *Implicit Enumeration Algorithm*

Example	Design	Total Possible Nodes	Nodes Evaluated	Equivalent Subproblems	CPU seconds*
REEF	d^1	15	7	12	75.1
COMP	d^1	31	11	23	20.5
	d^2	31	17	31	44.8
HX1	d^1	8,191	95	121	237.0
HX2	d^1	8,191	3	9	28.1

* DEC-20.

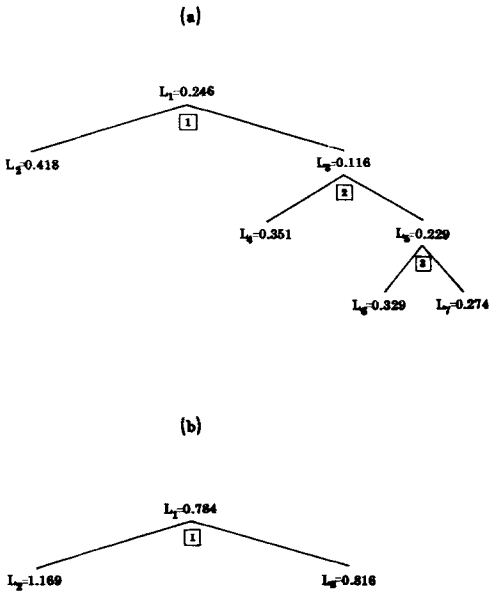


Figure 7. Enumeration trees for refrigeration cycle example and heat exchanger network HX2. (a) Example REEF; (b) example HX2.

Eq. 2 for an individual vertex direction. The nominal point values $x^N, z^N, \delta = 0$ were used as starting values; in the heuristic search, a value of 1.0 was chosen for the parameter ρ^{MAX}. For all of the cases investigated, the heuristic search was able to locate the correct solution. Whereas complete enumeration of all vertices requires 2^p subproblems, the heuristic procedure required the evaluation of only a few vertices, allowing a substantial reduction in CPU time. This advantage is particularly noticeable in examples HX1 and HX2, for which complete enumeration is not practical. (Since the vertex subproblems in those examples take about 5 CPU-seconds each, more than 5 CPU-hours would be required for total vertex enumeration.) For each case, the initial vertex direction was taken as the one with positive deviations for all uncertain parameters. Experiments with more involved schemes for selecting the initial vertex showed algorithm performance to be relatively insensitive to the starting choice.

In Table 5b the results of the implicit enumeration algorithm are presented. Again, the correct vertex solutions were found in each of the examples. The actual number of node evalutions required by the algorithm may be compared to the $(2^{p+1} - 1)$ total possible nodes contained in the binary tree. For the examples investigated, the lower bound was relatively effective in reducing the required search effort, especially for the problems HX1 and HX2 with many uncertain parameters.

The entries labeled "equivalent subproblems" are presented to allow comparison with the results from the direct search algorithm. A node evaluation, i.e., the lower bound computation in Eq. 14, usually requires more effort than solving a vertex evaluation subproblem as in Eq. 2. The reduced constraints in Eq. 14 must be evaluated for each of the vertex directions $\bar{\theta}^{*,i}$, and this requires the determination of a separate set of state variables x for each discrete vertex. Since, for a full-scale flowsheet optimization calculation the computing requirements will be dominated by the work required to solve the large-scale system $h(\delta,z,x) = 0$ for x, a good indicator of overall effort is the number of times (in series or in parallel) that the large system of equalities must be solved. Correspondingly, the number of equivalent subproblems listed is the sum of the number of discrete vertices $\bar{\theta}^{*,i}$ in Eq. 14 taken over all nodes evaluated. Because of sparsity in $\partial f/\partial \theta^T$, the number of discrete vertices required was quite modest, starting at a maximum of 3 for COMP, HX1, and HX2; for REEF; and decreasing at deeper levels in the trees.

Although constraint monotonicity in θ is required for the algorithm to be rigorous, its execution does not appear to be hindered by nonmonotonicity. Negative values for γ_ℓ were observed during the solution of both REEF and COMP, Figure 7a, indicating nonmonotonic behavior. Although the computed bound $\overline{L}$ sometimes actually decreased instead of monotonically increasing, the algorithm eventually terminated with the correct solutions, at least for the cases observed here. It is important to point out that for examples HX1 and HX2, constraint monotonicity in θ is guaranteed. In Swaney (1983) this is shown to be a general property for heat exchanger networks when bypassing is available and the target temperatures are specified by inequality constraints. As a result, the solutions for these problems obtained through the branch-and-bound procedure are rigorous (assuming a vertex solution), and the lower bounds increase monotonically, Figure 7b.

DISCUSSION

For problems that involve a small number of uncertain parameters, the total vertex enumeration search procedure is clearly a simple and direct way of determining the flexibility index. For cases with large numbers of uncertain parameters the implicit enumeration procedure seems to be greatly superior to total vertex enumeration, although more expensive than the nonrigorous heuristic procedure. The heuristic procedure, in addition to being very efficient appears to be quite reliable, based on the limited number of example problems where it has been tested. The vertex solutions obtained with any of these algorithms can be tested for the possible existence of nonvertex critical points, in which case the procedure suggested in Appendix II can be applied.

It should be noted that implementation of the proposed algorithms to solve full-scale process design problems should not be difficult. Any of the techniques or software packages developed to perform flowsheet optimization could be adapted to solve the nonlinear programming subproblems required to determine the flexibility index. For the heuristic procedure in particular, the computer time requirements for a large chemical process should not be excessive since typically only a few optimization subproblems would have to be solved. Additional information regarding the sensitivities of the flexibility index to changes in the design variables may be easily obtained from the solutions provided by any of the proposed algorithms.

Finally, it is worth noting that the algorithms developed in this paper could be applied within the algorithm by Halemane and Grossmann (1983) for optimal design under uncertainty. That procedure requires that feasibility be ensured at the vertices of a set of parameters with fixed lower and upper bounds. The algorithms presented here could be adapted to their feasibility calculation with virtually no modification. Another possibility however, is to replace their feasibility test with the flexibility index itself, where the feasibility condition would correspond to $F \geq 1$. The advantage of this option is that the results yielding $F \geq 1$, or $F < 1$, have a quantitative interpretation, whereas the feasibility test only indicates feasibility or infeasibility.

ACKNOWLEDGMENT

The authors would like to acknowledge financial support provided by the National Science Foundation under grant CPE-8121665, and the Exxon Industrial Fellowship.

APPENDIX I

Derivation of Lower Bound

A useful interpretation of the problem in Eq. 1 results when the function $\delta^*(\bar{\theta})$ is decomposed as follows. By defining the following nonlinear program (NLP) for any specified z

$$\delta'(\bar{\theta},z) = \max_{\delta'} \delta'$$

$$\text{s.t. } f(d,z\theta) \le 0$$

$$\theta = \theta^N + \delta'\tilde{\theta}$$

$$\delta' \ge 0$$

$$\{\exists z' | f(d,z',(\theta^N + \delta\tilde{\theta})) \le 0\} \forall \delta \in [0,\delta'] \tag{A1}$$

a function $\delta'(\tilde{\theta},z)$ may be constructed so that $\delta^*(\tilde{\theta})$ may be expressed as

$$\delta^*(\tilde{\theta}) = \max_z \delta'(\tilde{\theta},z) \tag{A2}$$

A suitable definition for $\delta'(\tilde{\theta},z)$ is

$$\delta'(\tilde{\theta},z) = \begin{Bmatrix}\text{Solution to Eq. A1 if feasible for } (\tilde{\theta},z)\\ -\infty \text{ if Eq. A1 is infeasible for } (\tilde{\theta},z)\end{Bmatrix}. \tag{A3}$$

$\delta'(\tilde{\theta},z)$ therefore represents the maximum feasible deviation in direction $\tilde{\theta}$ for a given choice of the controls z. (The last line in Eq. A1 is simply the analog of Assumption 1 in Part I included formally as a constraint.)

Though it is not intended that $\delta'(\tilde{\theta},z)$ ever actually be computed, its use permits Eq. 1 to be expressed as

$$F = \min_{\tilde{\theta}\in\tilde{T}} \max_z \delta'(\tilde{\theta},z) \tag{A4}$$

The interpretation of the flexibility index is then direct: Determine the direction of parameter deviations $\tilde{\theta}$ for which the greatest feasible deviation $\delta'(\tilde{\theta},z)$ is minimized, given that for each choice of $\tilde{\theta}$ the controls z will be chosen to maximize the permissible deviation.

The importance of the min-max formulation lies in the following inequality (see Avriel, 1976), which may be applied to any subset S of the vertices in V:

$$\max_z \min_{\tilde{\theta}\in S} \delta'(\tilde{\theta},z) \le \min_{\tilde{\theta}\in S} \max_z \delta'(\tilde{\theta},z) \tag{A5}$$

The lefthand side of Eq. A5 will provide the lower bound which is needed for the implicit enumeration scheme. This bound may be expressed in computable form as shown below.

From Eq. A5, the lower bound for any vertex subset S is given by

$$L(S) = \max_z \min_{\tilde{\theta}\in S} \delta'(\tilde{\theta},z). \tag{A6}$$

In order to develop a practical means of computing $L(S)$, it is convenient to decompose the function $\delta'(\tilde{\theta},z)$ into components $\delta_i'(\tilde{\theta},z)$ as follows. For each constraint function $f_i, i \in I$, define

$$\delta_i'(\tilde{\theta},z) = \begin{Bmatrix}\text{Solution to Eq. A8 if feasible for } (\tilde{\theta},z)\\ -\infty \text{ if Eq. A8 is infeasible for } (\tilde{\theta},z)\end{Bmatrix}. \tag{A7}$$

based on the NLP

$$\delta_i'(\tilde{\theta},z) = \max_{\delta_i'} \delta_i'$$

$$\text{s.t. } f_i(d,z\theta) \le 0$$

$$\theta = \theta^N + \delta_i'\tilde{\theta}$$

$$\delta_i' \ge 0$$

$$\{\exists z' | f_i(d,z',(\theta^N + \delta\tilde{\theta})) \le 0\} \forall \delta \in [0,\delta_i'] \tag{A8}$$

in direct analogy with the definition of $\delta'(\tilde{\theta},z)$ in Eqs. A1 and A3. The function $\delta_i'(\tilde{\theta},z)$ represents the value for $\delta'(\tilde{\theta},z)$ that would result if $f_i(d,z,\theta)$ were the only constraint. It then follows that

$$\delta'(\tilde{\theta},z) = \min_{i\in I} \delta_i'(\tilde{\theta},z) \tag{A9}$$

since Eq. A3 defines $\delta'(\tilde{\theta},z)$ to be the greatest feasible deviation subject to all of the constraints $i \in I$. Employing Eq. A9 in Eq. 6 leads to the following:

$$L(S) = \max_z \min_{\tilde{\theta}\in S} \delta'(\tilde{\theta},z) \tag{A6}$$

$$= \max_z \min_{\tilde{\theta}\in S} \min_{i\in I} \delta_i'(\tilde{\theta},z) \tag{A10}$$

$$= \max_z \min_{i\in I} \left[\min_{\tilde{\theta}\in S} \delta_i'(\tilde{\theta},z)\right] \tag{A11}$$

$$= \max_z \min_{i\in I} \delta_i'(\tilde{\theta}^{*,i},z) \tag{A12}$$

where $\tilde{\theta}^{*,i}$ is defined to be the solution to

$$\min_{\tilde{\theta}\in S} \delta_i'(\tilde{\theta},z). \tag{A13}$$

At this point it would be possible to retain generality and to treat $\tilde{\theta}^{*,i}$ as a function of z, computed from Eq. A13. In the interest of efficient implementation, however, it will be assumed that the vertices $\tilde{\theta}^{*,i}$ which minimize $\delta_i'(\tilde{\theta},z)$ (or, in other words, which maximize the constraints $f_i(d,z,\theta)$ individually in Eq. A8) do not change with variations in the controls z, i.e., that they are constants. A sufficient condition for this behavior is for the constraints $f(d,z,\theta)$ to be monotonic in each component of θ. Under the monotonicity assumption, the values of the $\tilde{\theta}^{*,i}$ may be determined directly from the signs of the gradients as in Eq. 9, subject to the restriction $\tilde{\theta}^{*,i} \in S$.

Finally, a formulation convenient for computing the lower bound can be obtained by noting that Eq. A12 is equivalent to

$$L(S) = \max_{\delta,z} \delta$$

$$\text{s.t. } \delta \le \delta_i'(\tilde{\theta}^{*,i},z), i \in I \tag{A14}$$

which by Eq. A8 is in turn equivalent to the nonlinear program

$$L(S) = \max_{\delta,z} \delta$$

$$\text{s.t. } f_i(d,z,\theta^{(i)}) \le 0, i \in I$$

$$\theta^{(i)} = \theta^N + \delta\tilde{\theta}^{*,i}, i \in I \tag{A15}$$

APPENDIX II

Procedure to Locate Nonvertex Critical Points

Given a vertex solution obtained by any of the algorithms and which exhibits a direction of descent as given by Eq. 7, the objective is to locate a local solution in the neighborhood of the vertex. This involves solving the following problem

$$\min_{\tilde{\theta}} \delta^*(\tilde{\theta})$$

$$\text{s.t. } -\Delta\theta^- \le \tilde{\theta} \le \Delta\theta^+ \tag{A16}$$

where $\delta^*(\tilde{\theta})$ is defined by the NLP in Eq. 2. The proposed method is simply to apply the successive quadratic programming approach of Wilson (1963), Han (1977), and Powell (1977) directly to Eq. A16, solving Eq. 2 at each iteration to yield both the objective function and the gradients

$$v^T = -\lambda^T \frac{\partial f}{\partial \theta^T} \tag{A17}$$

where v^T is obtained as shown in Part I, Eqs. 19a–g. The quadratic program (QP) at each iteration takes the form

$$\min_{p\tilde{\theta}} v^T p_{\tilde{\theta}} + \tfrac{1}{2} p_{\tilde{\theta}}^T B_{\tilde{\theta}} p_{\tilde{\theta}}$$

$$\text{s.t. } (-\Delta\theta^- - \tilde{\theta}) \le p_{\tilde{\theta}} \le (\Delta\theta^+ - \tilde{\theta}) \tag{A18}$$

yielding the search direction $p_{\tilde{\theta}}$. The QP constraints are simple bounds; the line-search function reduces to simply $\delta^*(\tilde{\theta} + \alpha p_{\tilde{\theta}})$ for step length α; the approximate Hessian $B_{\tilde{\theta}}$ is estimated in the conventional manner (Powell, 1977) using successive values for v^T. As the solution is approached, computing requirements may be reduced noticeably by using the variable values from the previous iteration as starting values when solving Eq. 2.

NOTATION

A_i = area of heat exchanger i, m^2
d = vector of design variables
f = vector of reduced in equalities
F = flexibility index
F_i = heat capacity flowrate stream i, kW/K
F_O,F_R = molar flowrates, kgmol/s
H = compressor head, kJ/kg
I = index set for components of vector f
K_v = index set for parameter vertices
K_f = pressure drop resistance
L = lower bound for flexibility index
mC_p = heat capacity flowrate, kW/K
M_O,M_R = molecular weight, kg/kgmol
p = number of uncertain parameters
ΔP_v = pressure drop in valve, kPa
Q = compressor throughput, m^3/s
r_{fi} = fouling factor of exchanger i, m^2·K/kW
R_i = inverse of overall transfer coefficient i, m^2·K/kW
S = subset of vertices
T_i = temperature stream i, K
$\tilde{T}$ = hyperrectangle for parameter deviations
U_i = overall heat transfer coefficient exchanger i, kW/m^2·K
v = vector of multipliers
V = set of parameter vertices
$\dot{V}$ = compressor volumetric capacity, m^3/s
W = driver power for compressor, kW
y_j = binary variable for parameter j
Y = set of binary variables
z = vector of control variables
z_i = effective conduction in exchanger i, kW/K

Greek Letters

γ_j = predicted increase of lower bound with parameter j
δ = scaled parameter deviation
η = compressor efficiency
θ = vector of uncertain parameters
θ^N = nominal value of vector θ
$\tilde{\theta}$ = displacement direction from θ^N
$\Delta\theta$ = vector of expected parameter deviations
λ = vector of multipliers
ρ_i = predictor for linearized constraint i
σ_j = sensitivity of F with design variable d_j

LITERATURE CITED

Avriel, M., *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1976).

Brayton, R. K., et al., "A New Algorithm for Statistical Circuit Design Based on Quasi-Newton Methods and Function Splitting," *IEEE Trans. on Circuits and Systems*, **CAS-26**, 784 (1979).

Edahl, R. H., M. H. Locke, and A. W. Westerberg, "Improved Successive Quadratic Programming Optimization Algorithm for Engineering Design Problems," *AIChE J.*, **29**, 871 (1983).

Gill, P. E., et al., "Documentation for SOL/QPSOL: A Fortran Package for Quadratic Programming, Version 1," Office of Technology Licensing, Stanford Univ., Stanford, CA (1982).

Grossmann, I. E., and R. W. H. Sargent, "Optimum Design of Chemical Plants with Uncertain Parameters," *AIChE J.*, **24**, 1,021 (1978).

Han, S. P., "A Globally Convergent Method for Nonlinear Programming," *J. Optimization Theory & Appl.*, **22**, 297 (1977).

Halemane, K. P., and I. E. Grossmann, "Optimal Process Design Under Uncertainty," *AIChE J.*, **29**, 425 (1983).

Harwell Subroutine Library, "MA28AD," Mathematics Branch, A.E.R.E., Harwell, Berkshire, UK (Apr., 1977).

Murty, K. G., *Linear and Combinatorial Programming*, Wiley and Sons, Inc., New York (1976).

Polak, E., "An Implementable Algorithm for the Optimal Design Centering, Tolerancing, and Tuning Problem," *J. Optimization Theory & Appl.*, **37**, 45 (1983).

Polak, E., and A. Sangiovanni-Vincentelli, "Theoretical and Computational Aspects of the Optimal Design Centering, Tolerancing, and Tuning Problem," *IEEE Trans. on Circuits and Systems*, **CAS-26**, 795 (1979).

Powell, M. J. D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations," *Numerical Analysis. Proc. Biennial Conf., Dundee, June 1977*, G. A. Watson, Ed., Lecture Notes in Mathematics, **630**, Springer, Berlin.

Swaney, R. E., "Analysis of Operational Flexibility in Chemical Process Design," Ph.D. Thesis, Dept. Chem. Eng., Carnegie-Mellon Univ., Pittsburgh, PA (1983).

Swaney, R. E., and I. E. Grossmann, "An Index for Operational Flexibility in Chemical Process Design. I: Formulation and Theory," *AIChE J.*, **31** (1985).

Wilson, R. B., "A Simplicial Algorithm for Concave Programming," Ph.D. Thesis, Grad. Sch. Bus. Admin., Harvard Univ., Cambridge, MA (1963).

Manuscript received Jan. 5, 1984; revision received Mar. 5, 1984, and accepted Mar. 21.