Efficient Optimization Strategies with Constraint Programming

Prakash R. Kotecha, Mani Bhushan, and Ravindra D. Gudi
Dept. of Chemical Engineering, Indian Institute of Technology Bombay, Powai, Mumbai 400 076, India

DOI 10.1002/aic.11972
Published online August 24, 2009 in Wiley InterScience (www.interscience.wiley.com).

In this article, we propose novel strategies for the efficient determination of multiple solutions for a single objective, as well as globally optimal pareto fronts for multiobjective, optimization problems using Constraint Programming (CP). In particular, we propose strategies to determine, (i) all the multiple (globally) optimal solutions of a single objective optimization problem, (ii) K*-best feasible solutions of a single objective optimization problem, and (iii) globally optimal pareto fronts (including nonconvex pareto fronts) along with their multiple realizations for multiobjective optimization problems. It is shown here that the proposed strategy for determining* K*-best feasible solutions can be tuned as per the requirement of the user to determine either* K*-best distinct or nondistinct solutions. Similarly, the strategy for determining globally optimal pareto fronts can also be modified as per the requirement of the user to determine either only the distinct set of pareto points or determine the pareto points along with all their multiple realizations. All the proposed techniques involve appropriately modifying the search techniques and are shown to be computationally efficient in terms of not requiring successive re-solving of the problem to obtain the required solutions. This work therefore convincingly addresses the issue of efficiently determining globally optimal pareto fronts; in addition, it also guarantees the determination of all the possible realizations associated with each pareto point. The uncovering of such solutions can greatly aid the designer in making informed decisions. The proposed approaches are demonstrated via two case studies, which are nonlinear, combinatorial optimization problems, taken from the area of sensor network design.* © 2009 American Institute of Chemical Engineers *AIChE J,* 56: 387–404, 2010

Keywords: optimization, design, multiobjective optimization, constraint programming, sensor network design, multiple solutions, pareto fronts

Introduction

Optimization plays an important role in various disciplines of science and engineering. In recent years, significant research efforts have been directed towards global optimization, especially for nonconvex problems involving discrete and continuous variables. Optimization in the presence of multiple (and often times conflicting) objectives is also increasingly finding relevance and has been a focus of recent research thrusts.

In a recent survey article, Grossman and Biegler[1,2] presented a spectrum of methods for global optimization including both deterministic and stochastic approaches. For the class of discrete optimization problems, logic based methods have provenly offered a high potential in reducing computational complexity.[3] Constraint Programming (CP)[2,4,5] is one of the logic-based optimization techniques that has found applications in diverse areas such as computer graphics, software engineering, databases, hybrid systems, finance, and circuit design. CP has been found to be particularly successful in areas involving combinatorial optimization such as planning, resource allocation, and scheduling.

Some of the other important requirements in an optimization problem, in addition to determining the globally optimal

A shorter version of this manuscript was presented in the AIChE Annual Meeting (2007).
Correspondence concerning this article should be addressed to R. D. Gudi at ravigudi@iitb.ac.in

solution, are determining the multiple realizations of the optimal solution (whenever present), and also evolving computationally efficient methods to determine the next best set of solutions.[6] Although CP has been advocated primarily due to its ability to generate global optimum solutions, its use for determining multiple optimum solutions and *K*-best feasible solutions has remained largely unexplored. For combinatorial optimization problems, such multiple optimal realizations have traditionally been determined using integer cuts.[6] The idea is to first solve the original optimization problem and obtain a single optimal solution. The optimization problem is then re-solved with an additional cut which ensures that this optimal solution is not revisited. This procedure of re-solving the optimization problem is continued till the objective function of the successive problem does not deteriorate. Deterioration in the objective function indicates that all the multiple solutions have been obtained. Despite its simplicity, this method suffers from the following drawbacks:

1. The construction of integer cuts is straight forward for binary variables and these are linear in nature but in the case of integer variables, these cuts are nonlinear. If linear cuts are desired for integer variables, then the integer variables need to be suitably transformed to binary variables.

2. This procedure involves successively re-solving the entire optimization problem and hence requires additional computational effort.

3. The success of this strategy is linked with the capability of the solver used for solving the optimization problem. In general, global optimization solvers are not available for nonconvex, combinatorial problems, and hence, this technique might not be able to discover all the multiple solutions in the case of frequently encountered nonlinear integer programming problems.

Recently,[7] an alternative CP-based two-step strategy has been proposed to determine multiple solutions. Although this overcomes the drawbacks related to the addition of integer cuts, it still requires solving two distinct problems. The first step is to solve the optimization problem and determine the optimal objective function value. This is followed by the solution of a feasibility problem such that the objective function values of all the feasible solutions are the same as the optimal objective function value.

The technique for determination of *K*-best feasible (*K* being a user-defined parameter) solutions is similar to the determination of multiple solutions except for the termination criteria. The procedure is not terminated on realizing a deterioration in the objective function but is instead terminated when the required number (*K*) of best feasible solutions are obtained. Similar to the determination of multiple solutions, this method involves successively re-solving the problem. A recent development in this direction has been modification of the branch and bound strategy as in BARON[6] to determine multiple solutions and *K*-best feasible solutions.

A significant number of real life optimization problems are multiobjective in nature. Very often, these objectives are mutually conflicting and hence there may not be a single optimal solution. Optimality in such problems is usually characterized by a set of trade-off points which are commonly referred to as pareto optimal points. A collection of pareto optimal points constitutes a pareto front. These multiobjective optimization problems are relatively more complex than the single objective optimization problems as they involve two search spaces namely the decision variable space and the objective or criterion space.[8] Another interesting possibility that can arise is the existence of multiple realizations (in the decision variable space) of a pareto optimal point in the objective function space.[9] It would be of significant practical importance to uncover these multiple realizations toward analysis of the pareto front. In this article, to provide a clear distinction between the two, we will refer to pareto optimal solutions differing in objective function values as pareto points whereas solutions identical in objective function value but differing in the decision variable space will be referred to as pareto solutions (or realizations). Hence, a single pareto point may correspond to several pareto solutions. Deb[8] has reviewed many of the traditional multiobjective optimization techniques. These traditional techniques, apart from their other drawbacks, do not guarantee the determination of nonconvex pareto fronts. The ε-constraint method has been reported to be capable of determining such nonconvex pareto fronts. This method involves the conversion of multiobjective problem into a single objective problem by considering only one objective and imposing constraints on threshold values for other objectives. However, this method suffers from drawbacks such as:

1. The optimality of pareto front critically depends on the method that handles the underlying single-objective optimization problem which often are combinatorial and nonlinear in nature.

2. To generate the pareto front, several such problems with different thresholds need to be solved. Additionally, systematic methods for choosing different thresholds that will enable generation of all points on the pareto front are not available in general.

3. Generation of all pareto solutions corresponding to a pareto point is not straightforward in the ε-constraint method.

4. This method can lead to suboptimal solutions if the underlying single-objective optimization problems have multiple solutions.

In contrast to the deterministic techniques, evolutionary techniques such as Genetic Algorithms (GA)[8] are popular and suitable for pareto front generation as they work with a population (as against a single solution) and do not share the drawbacks of the traditional techniques. Specifically, they do not require threshold or preference specifications and do not require that the problem be specified in an explicit form (can deal with constructive constraints). However, they involve other issues such as (i) specification of tuning parameters such as crossover and mutation probabilities which can significantly affect the results and (ii) lack of guarantee of global optimality even for well structured problems (such as linear).

Although the broad philosophy in widely used solution approaches to multiobjective problems has been to solve successive problems after translating some of the objectives into constraints, the approach in CP is to treat all the objectives as constraints, and solve successive feasibility problem. In a recent work, the ability of CP to efficiently solve feasibility problems towards the determination of globally optimal pareto fronts (even if they are nonconvex)[7] was demonstrated by generating all the feasible solutions of the optimization problem. This technique can be computationally expensive for combinatorial problems since it requires the generation of all feasible solutions. However, the ability of CP to efficiently solve feasibility problems makes it amenable for incorporation of other criteria during the search (as will be shown in the paper) that facilitate the determination of globally optimal pareto fronts without requiring the determination of all feasible solutions, and thus make it a potent methodology for solving multiobjective optimization problems.

From the earlier literature review, it can be seen that development of techniques for efficient determination of multiple solutions for single objective and pareto optimal solutions for multiobjective optimization problems is an important problem and remains largely unexplored. The thrust of this paper is on the development of efficient techniques which can bridge this gap. As discussed earlier, in the literature, the determination of multiple solutions for single objective optimization problems requires either successively re-solving the optimization problem with an integer cut or the solution of a feasibility problem in succession with an optimization problem. In this article, we propose a CP-based strategy to determine all the multiple optimal solutions of an optimization problem in a single step. In addition, we also present a single-step strategy to determine *K*-best feasible solutions for a single objective optimization problem. We show that this strategy can be tuned as per the requirement of the user to either determine distinct or nondistinct *K*-best feasible solutions. Further, as discussed earlier for the case of multiple objectives, the determination of globally optimal pareto fronts has been hitherto unaddressed. Moreover the determination of all possible realizations associated with a pareto point has received no attention. In this article, we present techniques which address these two lacunae for combinatorial optimization problems in a computationally efficient way. We extend the strategies developed for single objective optimization problem to develop efficient procedures for determining the globally optimal pareto solutions by proposing dynamic addition of appropriate integer cuts during the CP search procedure. It is demonstrated in this article that these strategies lead to globally optimal solutions along with all possible realizations even for problems involving nonconvex pareto optimal fronts in a single run (or two runs if the two-step proposed strategy is selected). All of the above strategies proposed for both the single and multiobjective case have been validated on representative case studies taken from the sensor network design literature.

The rest of the article is organized as follows: In the second section, we present a brief overview of the CP technique and discuss some required preliminary details. This is followed by a discussion of the techniques proposed in this article in the third section. In this section, we first develop the techniques for single objective optimization followed by the strategies for multiobjective optimization. In the fourth section, we present two case studies from the sensor network design literature and demonstrate the applicability of the proposed techniques. Following a discussion on the salient aspects of computational performance of the proposed algorithms, we conclude the article in the fifth section by summarizing the developments in this article and present possible future work.

Constraint Programming

CP is an intelligent, tree-based, implicit enumerative search technique. It has been developed by the Artificial Intelligence and Computer Science Community for solving Constraint Satisfaction Problems (CSPs) arising in these fields. A CSP is a feasibility problem and is similar to an optimization problem except for the absence of an objective function. Hence, any solution which satisfies the set of constraints is an acceptable solution. However, CP can also be used to solve optimization problems by transforming them into corresponding CSPs. This is implemented by ignoring the objective function and instead adding a new variable corresponding to the objective function of the original problem. A condition is enforced during the search procedure which ensures that the value of the objective function (as represented by the value of the new variable) progressively improves throughout the exploration of the feasible space. Figure 1 shows the working principle of CP to solve a CSP and a detailed algorithm can be obtained from literature.[2,10,11]

Unlike mathematical programming techniques, CP does not rely on integer relaxations or gradients but instead uses constraint propagation to reduce the domain of the variables. Thus, it avoids convergence to suboptimal (or local) solutions irrespective of the nature (convex/nonconvex) of the problem. The computational efficiency (speed of convergence) of CP is governed by the algorithms used for the propagation of constraints. Some of the important benefits of CP over traditional mathematical programming techniques can be summarized as

(i) *Modeling:* The expressive modeling ability of CP can be used to develop compact models with minimal efforts. For example, traditional mathematical programming techniques require the constraints of an optimization problem to be of the form $g(x) \leq 0$. However, CP permits a richer description of the problem and does not impose any such restriction. It can directly include, without any transformation, constraints involving the $\neq$ operator, implication

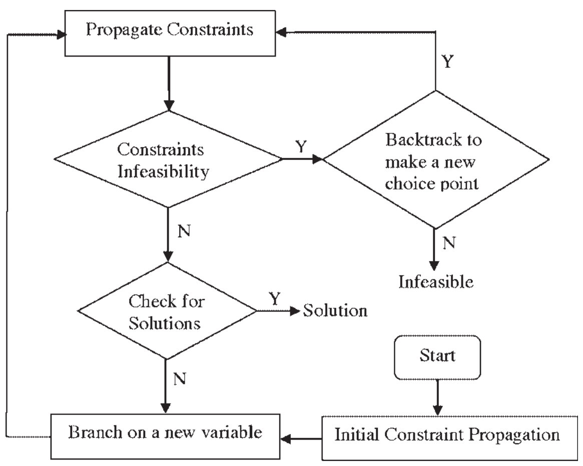


Figure 1. CP algorithm for solving a feasibility problem.

constraints or disjunctive constraints. Additionally, CP modeling environments have problem specific constructs which result in compact models thereby enabling efficient constraint propagation leading to significant computational savings.

(ii) *Optimality:* CP can determine globally optimal solutions for single objective problems and globally optimal pareto points for multiobjective optimization problem irrespective of the combinatorial and convex nature of the problem.

(iii) *Realizations:* An interesting characteristic of combinatorial problems is the existence of multiple solutions. This is especially true for multiobjective optimization problems where each point on the pareto front can have several realizations (pareto solutions). Such realizations can also be determined using CP.

Additionally, the search strategy of CP can be easily modified so as to satisfy some soft constraints.[11] As will be shown in this article, CP can also be used to determine K-best feasible solutions of a single objective optimization problem. In literature, CP and Integer Programming (IP) have been compared for the modified generalized assignment problem,[12] the template design problem,[13] the progressive party problem,[14] and the change problem.[15] It is widely known in literature that CP is very effective for highly constrained problems but is not as effective for loosely constrained problems that have limited potential for domain reduction. Most traditional optimization techniques use some mathematical property such as convexity or bounds obtained by appropriate relaxations to confirm local/global optimality of the obtained solution. However, since CP is purely a search-based technique, it may require considerable amount of computational resource even after the determination of an optimal solution to confirm this optimality. Some of the available CP solvers are ILOG Solver, CHIP, and ECLiPSe. A recent development in the field of combinatorial problems is the development of hybrid methods,[10,16,17] which use the complimentary properties of IP and CP to efficiently solve combinatorial problems.

Before discussing the proposed strategies, we present some preliminary details which may aid in the clarity of the rest of the article. All the strategies proposed in this article require the conversion of an optimization problem to a corresponding feasibility problem. Before discussing the proposed algorithms, we briefly mention this conversion. Consider a single objective optimization problem as shown in formulation SO.

$$\textbf{SO} \qquad \begin{array}{ll} \text{Optimize} & f(\mathbf{x}) \\ \text{s.t.} & G(\mathbf{x}) \end{array} \tag{1}$$

where $f(\mathbf{x})$ is a multivariable single objective function that needs to be optimized subject to the set of constraints represented by $G(\mathbf{x})$. The optimization problem in SO can be converted to a feasibility problem as shown in SF

$$\textbf{SF} \qquad \begin{array}{l} \text{Solve} \quad G(\mathbf{x}) \\ F = f(\mathbf{x}) \end{array} \tag{2}$$

The problem statement in SF is a feasibility problem and any point which satisfies the set of constraints represented by $G(\mathbf{x})$ is considered to be a solution. It can be seen that the concept of optimization is ignored as the objective function is only evaluated (and stored in the variable F) and not directly optimized (as in SO). The problem statement in SF can also be viewed as a relaxation (with no change in the feasible region) of SO and hence it is obvious that the set of optimal solutions of SO form a subset of the solutions in SF.

On similar lines, the above strategy of converting a single objective optimization problem to a feasibility problem can be extended to multiobjective optimization problems also. MO and MF are the corresponding multiobjective counterparts of SO and SF. The problem statement represented in MO is a multiobjective optimization problem with m multivariable objectives.

$$\textbf{MO} \qquad \begin{array}{ll} \text{Optimize} & f_j(\mathbf{x}); \quad j = 1, 2, \ldots, m \\ \text{s.t.} & G(\mathbf{x}) \end{array} \tag{3}$$

MF represents the feasibility problem of MO with each of the objectives evaluated. F is a m dimensional vector whose jth element corresponds to the value of the jth objective function.

$$\textbf{MF} \qquad \begin{array}{l} \text{Solve} \quad G(\mathbf{x}) \\ F_j = f_j(\mathbf{x}); \quad j = 1, 2, \ldots, m \end{array} \tag{4}$$

The use of CP removes some of the common restrictions posed in traditional mathematical programming techniques. For example, the individual constraints in the set of constraints represented by $G(\mathbf{x})$ need not be of the form $g(\mathbf{x}) \leq 0$. Bound constraints, if any, on the individual variables can also be included and are considered to be represented by $G(\mathbf{x})$. Additionally, there is no restriction on the variables to be non-negative which is in contrast to some existing techniques such as the popular simplex method that uses additional variables to incorporate this non-negativity constraint. In the rest of this article, we will be using the dominant operator $\lhd$[8] to avoid the distinction between a maximization problem and a minimization problem. For maximization problems, $\lhd(\unlhd)$ will represent the $>(\geq)$ operator and for a minimization problem, it will represent the $<(\leq)$ operator. We now present the algorithms proposed in this article.

Proposed Algorithms

Single objective optimization

In this section, we present efficient strategies for single objective optimization problems to determine multiple solutions which can be used to gain further insights into the optimization problem. We first present the strategy to determine multiple optimal solutions followed by strategies to determine the K-best feasible solutions for a single objective optimization problem.

Determination of Multiple Solutions Using a Single-Step Strategy. The determination of multiple solutions is important for a wide variety of reasons. Some of them are:

(1) Measure of operational flexibility/robustness: A large number of multiple optimal solutions indicate greater operational flexibility as the design can tolerate some failures without affecting the optimality as other potential alternatives could be available.

(2) Identifying promising solutions: Secondary objectives can be considered for the selection of a particular solution from the set of multiple solutions. This can be viewed as satisfaction of soft constraints/soft objectives and is particularly useful when these objectives cannot be easily formulated using an explicit expression.

(3) Identification of critical/key decisions: The occurrence of a particular decision in a large number of multiple realizations of the optimal solution might indicate that the decision is very critical to maintain the optimality. An example could be the usage of an equipment in all the multiple solutions for a process planning problem indicating that the equipment is critical for maintaining the optimality of the process.

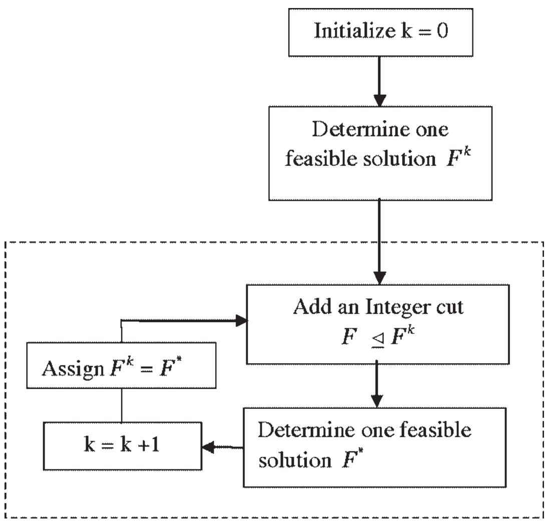
In the proposed strategy, we modify the conventional search strategy used in CP to solve single objective optimization problems. In this strategy, we search for solutions that are as good as or better than the current best solution. This is different from the conventional search strategy used in CP to solve optimization problems that explores solutions which are strictly better than the current best solution. This modification helps in determining all possible multiple realizations. The algorithm involves converting the optimization problem to a feasibility problem as shown earlier and determining a single feasible solution. The search is continued after adding the following cut on determining every subsequent (kth) solution.

$$F \trianglelefteq F^k \tag{5}$$

The above cut ensures that any subsequent ($k + 1$)th solution is as good as or better than the kth solution. Since this strategy is implemented after determining every feasible solution, it can be seen that the ($k + 1$)th solution will be as good as or better than the best of previously determined k solutions. This strategy ensures that all the multiple optimal solutions are obtained at the completion of search. This strategy is summarized in Figure 2. An important advantage of the proposed strategy is that it does not require the user to a priori specify the total number of optimal solutions. However, if required, this strategy can be appropriately modified to determine a specified number of optimal solutions.

Remark. If NOpt indicates the total number of multiple optimal solutions for an optimization problem and *Iter* indicates the total number of feasible solutions that have been explored, then the (Iter − NOpt)th feasible solution need not necessarily correspond to the second best optimal solution. This behavior is attributed to the fact that CP is an intelligent search technique and an optimum solution can be found even without exploring the second best optimal solution.

Determination of K*-best feasible solutions.* Many a times, the designer might be interested in determining K-best feasible solutions as it gives an idea of the deterioration in the objective function that has to be tolerated in case the optimal solution cannot be implemented/maintained due to unforeseen reasons. Additionally, the successive best solutions for an optimization problem may lead to only a small deterioration in the objective function but result in a large improvement in a secondary objective function. Such K-best feasible solutions can either be distinct or nondistinct. In the case of distinct solutions, all the K-best feasible solutions



Initialize k = 0
Determine one feasible solution F^k
Add an Integer cut $F \trianglelefteq F^k$
Assign $F^k = F^*$
k = k +1
Determine one feasible solution F^*

Figure 2. A single-step strategy for the determination of all multiple optimal solutions.

have distinct objective function value. Solutions with identical objectives but corresponding to different values of decision variables are considered as one single solution in the set of K-best feasible solutions. For the case of nondistinct K-best feasible solutions, the K-best solutions need not necessarily have distinct objective values and thus solutions with identical objective values but differing in the solution space are considered as distinct solutions in the set of K-best feasible solutions. In the following discussions, it is assumed without any loss of generality, that there are atleast K feasible solutions (distinct/nondistinct as the case may be).

Determination of nondistinct K*-best feasible solutions.* A naive way to determine nondistinct K-best feasible solutions is to determine the globally optimal solution and re-solve the optimization problem such that the previous solution is not revisited. This is achieved by the addition of suitable cuts. Apart from the various issues involved in the construction of such cuts discussed earlier, this technique also requires the solution of K optimization problems and hence can prove computationally expensive. In this section, we present a CP-based strategy which helps in the determination of such K-best feasible solutions even for nonlinear IP formulations. The idea in the proposed strategy is to maintain a list of K-best feasible solutions and keep updating this list as better solutions are determined during the exploration of the search space.

The determination of nondistinct K-best feasible solutions involves the conversion of the given optimization problem (SO) into a corresponding feasibility problem (SF). The first step is then to determine K feasible solutions of such a feasibility problem. The search strategy then proceeds with the criterion that successive solutions need to be better than the worst of these current set of K-best feasible solutions. This can be implemented using the following cut in the search procedure:

$$F \triangleleft \underset{\forall i \in K\text{sol}}{\text{Infimum}}(F^i) \tag{6}$$

where Ksol is the set of current K-best feasible solutions and *Infimum* represents the min (max) operator for a maximization (minimization) problem. If at the kth iteration, a solution better than the most inferior solution from the set of current K-best feasible solution is obtained, the set of K-best feasible solutions is updated by adding the better solution and excluding this most inferior solution (any one of them in case there are multiple such solutions) as shown in Eq. 7:

$$K\text{sol} = \{K\text{sol}\} \cup \{F^k\} \backslash (\text{Infimum}(K\text{sol})) \tag{7}$$

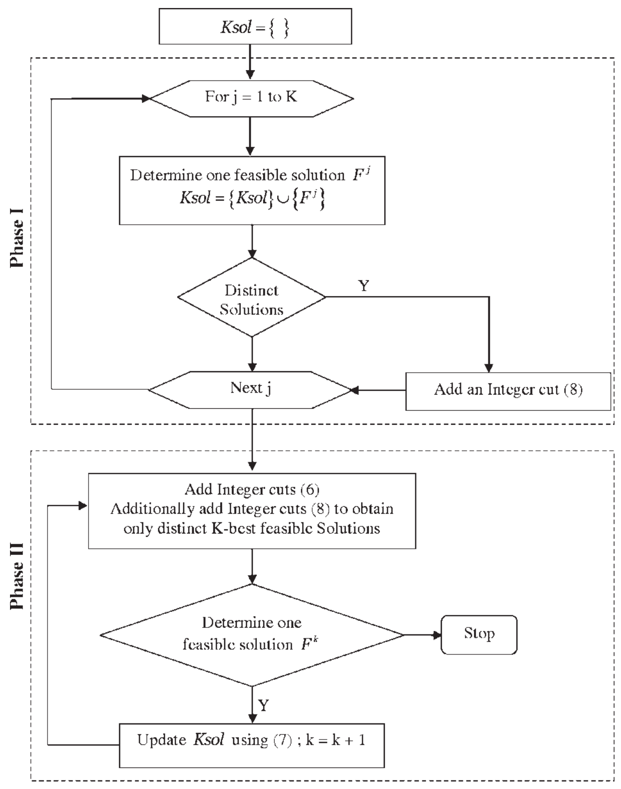
The K-best feasible solutions obtained at the end of the search strategy then correspond to the K-best feasible solutions of the problem SO.

Remark. If the number of multiple optimal solutions is greater than the value of K, the solutions for nondistinct K-best feasible solutions will be the same as the multiple optimal solutions. It can also be noted that the number of distinct solutions in the set of K-best feasible solutions need not necessarily increase monotonically because an inferior solution can be replaced by a realization of an existing superior solution thereby potentially decreasing the number of distinct solutions. However, since there cannot be any solution superior than the optimal solution, the number of optimal solutions in the set of K-best feasible solutions can only monotonically increase.

Determination of distinct K*-best feasible solutions.* The set of distinct K-best feasible solutions can also be naively determined by successively re-solving the problem after adding appropriate cuts. These cuts should ensure that not only the previous solution is not revisited but also the previous optimality (in terms of the objective function value) is not revisited. Both these requirements can be incorporated by adding a single cut which ensures that the previous optimality is not revisited. In this section, we present a strategy that determines these distinct K-best feasible solutions without necessarily re-solving the problem. The idea is similar to the determination of nondistinct K-best solutions but the following cut is additionally [along with (6)] added to ensure that the previous optimality is not revisited.

$$F \neq F^k \tag{8}$$

The above equation ensures that any of the previously determined k solutions are not revisited. It is straight forward to note that unlike in the nondistinct case, the number of distinct solutions in the set of K-best feasible solutions will remain constant during the entire search procedure. However as the search proceeds, the set of K-best feasible solutions keeps improving monotonically i.e., the most inferior solution of the set keeps improving. Both the proposed strategies to determine K-best feasible solutions are summarized in Figure 3. It may be noted from this figure that the determination of initial K feasible solutions (Phase I) does not involve the use of cuts in (6). Though it is desirable to use these cuts as successive solutions will be better, these are not added as it may hinder the discovery of K-best feasible solutions. For example, consider a scenario wherein the first feasible solution is the globally optimum solution. In this case, we will not discover any further feasible solutions if we include the constraint (6) as there will not be a better solution than this first feasible solution.



Ksol = { }
Phase I
For j = 1 to K
Determine one feasible solution F^j
$Ksol = \{Ksol\} \cup \{F^j\}$
Distinct Solutions
Y
Next j
Add an Integer cut (8)
Phase II
Add Integer cuts (6)
Additionally add Integer cuts (8) to obtain only distinct K-best feasible Solutions
Determine one feasible solution F^k
Stop
Y
Update Ksol using (7) ; k = k + 1

Figure 3. Algorithm to determine *K*-best feasible solutions.

Thus, these proposed strategies ensure the determination of K-best feasible solutions by dynamically adding cuts during the search procedure and avoid successively re-solving the optimization problem.

Remark. The cuts presented in Eqs. 6 and 8 are not of the conventional form $g(x) \leq 0$ but can be directly implemented in CP due to its expressive modeling power. However, these cuts can be replaced by the following two cuts for any optimization problem whose objective function takes only integer values.

$$F \trianglelefteq \text{Infimum } (K\text{sol}) + \alpha \tag{9}$$

In the above equation, $\alpha = 1$ for maximization problems and $\alpha = -1$ for minimization problems.

$$q(F - (F^k + 1)) \geq (1 - q)(F - (F^k - 1)) \tag{10}$$

where q is a binary variable.

Multiobjective optimization

Most real life optimization problems are typically multiobjective in nature. However, very often these objectives are competing, i.e. improvement in one objective is obtained at the cost of deterioration in atleast one of the other objectives. For such cases, the trade-offs between these competing objectives are evaluated using a pareto optimal front. The pareto optimal front is a collection of the set of nondominated solutions. A solution is said to be nondominated if it is feasible and there is no other feasible solution which has better values for all the objectives.[8] As the name indicates, a nondominated solution is neither superior nor inferior to any other point on the pareto front but is superior to any feasible solution that is not a part of the pareto front. However, a pareto point is superior to all other pareto points in atleast one of the objectives.

Determination of Pareto Points. Similar to single objective optimization problems, an interesting aspect of multiobjective optimization problems is the existence of multiple solutions i.e., each point on the pareto front can correspond to several realizations. As mentioned earlier, the pareto points aid the designer in making informed choices. However, if the realizations associated with each pareto point are also known, it may provide greater flexibility to the designer. Infact, the number of realizations can also aid the designer in selecting one of the pareto points. A pareto point with greater number of realizations might be a better choice than other pareto points as it will provide higher flexibility in ensuring that corresponding optimal objective values may be attained even in the presence of unforeseen circumstances. The existing CP-based method[7,18] for generation of the pareto optimal solutions perceives the problem to be composed of two separate entities, viz (i) the set of constraints and (ii) the set of objectives with no interaction between them. The set of constraints is used to generate all feasible solutions (from MF) and the pareto optimality (nondomination) criteria is then used to identify the pareto optimal solutions from the set of feasible solutions. The determination of all the feasible solutions can be computationally intensive for combinatorial problems. In the strategies proposed in the current article, we propose to use the pareto optimality criteria while generating a feasible solution thereby leading to potential reduction in the computational efforts. In this section, we discuss these strategies to determine pareto fronts with and without realizations. We first present a strategy to determine single realization of the pareto points.

Single Realization. This strategy is similar to the strategy traditionally used in CP to determine optimal solutions for solving single objective optimization problem. The optimization problem is first converted into a feasibility problem (as in MF) and a single feasible solution is initially determined. We then enforce the criterion that every subsequent solution should be better than the current solution in atleast one of the m objectives. This constraint can be implemented by the addition of the following Type I cut corresponding to the tth solution

$$\textbf{Type I} \qquad \bigvee_{j=1,2,\ldots,m} \left(F_j \lhd F_j^t\right) \tag{11}$$

where $\vee$ represents the commonly used OR operation in boolean logic and $\lhd$ indicates the dominant operator. The sequential addition of Type I cuts (one after each solution is generated) ensures that any subsequent solution will not be dominated by the earlier t solutions, since it will have to satisfy the following set of constraints

$$\left(F_1 \lhd F_1^i\right) \vee \left(F_2 \lhd F_2^i\right) \vee .. \vee \left(F_m \lhd F_m^i\right) \quad \forall i = 1,2,\ldots,t \tag{12}$$

The sequential addition of Type I cut continues till either the problem becomes infeasible or the search space gets completely explored. Let the number of feasible solutions (or cuts) generated by this procedure be T. It may be noted that some of these Type I cuts that were added to realize these T feasible solutions may actually be redundant. This can happen for example, when a subsequent solution is better in all objectives than one of the previous solutions. These redundant cuts correspond to solutions that are feasible in the original problem (MF) but are not pareto optimal points (i.e., they are dominated). These dominated feasible solutions can be removed by a sorting technique and the pareto front can be determined.

This procedure of adding Type I cuts (as in Eq. 11) sequentially will ensure that all the pareto-points are part of the set of T solutions, but it will not be helpful in determining additional realizations of the pareto points. To aid in the determination of such additional realizations, we present two strategies as discussed next.

Multiple Realization of all Pareto Points. In this section, we present two strategies to determine the pareto optimal points along with all their possible realizations. One of these strategies is based on a two-step approach and requires solving the feasibility problem twice whereas the alternative single-step strategy determines all the pareto points along with their realizations by solving a single feasibility problem.

Two-step strategy. This strategy is an extension of the two-step strategy used by Kotecha et al.[7] to determine multiple optimal solutions for a single objective optimization problem to a multiobjective optimization problem. The two steps are (i) determining the pareto optimal points (with single realization) and (ii) determining all possible realizations. The first step is implemented with the use of Type I cuts as discussed in the previous section (on determining pareto points without realizations). The second step involves the incorporation of the following Type II cut corresponding to each Type I cut to the original feasibility problem (as in MF).

$$\textbf{Type II} \qquad \left\{ \bigwedge_{j=1,2,\ldots,m} \left(F_j = F_j^i\right) \right\} \vee \left\{ \bigvee_{j=1,2,\ldots,m} \left(F_j \lhd F_j^i\right) \right\} \tag{13}$$

where i corresponds to the ith Type I cut, $\wedge$ denotes the boolean AND operator. It can be then seen that MF will be solved after incorporating the following T Type II cuts:

$$\left\{ \left(F_1 = F_1^i\right) \wedge \left(F_2 = F_2^i\right) \wedge .. \wedge \left(F_m = F_m^i\right) \right\} \vee \left\{ \left(F_1 \lhd F_1^i\right) \vee \left(F_2 \lhd F_2^i\right) \vee .. \vee \left(F_m \lhd F_m^i\right) \right\}; \quad \forall i = 1,2,\ldots,T \tag{14}$$

The Type II cut can be viewed as composed of two distinct parts. The first part is $\bigvee_{j=1,2,\ldots,m} \left(F_j \lhd F_j^i\right)$ with the rest of the cut being the second part (similar to Type I cut). Similarly, the T cuts can be considered to consist of two sets, the first one corresponding to the set of dominated solutions (redundant cuts) and the second corresponding to the set of nondominated solutions. All realizations corresponding to pth pareto point will satisfy the first part of the constraint corresponding to the pth pareto point. For all other constraints, they will satisfy the second part since they are not dominated by any other solution. On the other hand, the nonpareto solutions are dominated and hence there will be atleast one constraint (corresponding to a dominating pareto point) which is not satisfied by them. As a result, all feasible solutions of MF which satisfy the set of constraints (13) are pareto optimal solutions. These feasible solutions can be easily generated by CP using its intelligent search approach. It may be noted that though this strategy requires the solution of the feasibility problem (in MF) twice, the second step will be much faster than the first step due to the presence of the Type II cuts which may aid in considerable reduction of the feasible space. At the end of this two-step algorithm, the set of pareto points along with their realizations are directly determined without the need for any post processing (sorting).

Remark. It may be noted that the Type I cuts are determined and added sequentially after the determination of each feasible solution whereas the Type II cuts do not involve such sequential addition as they are directly constructed from the set of Type I cuts.

Single-step strategy. This strategy is similar to the single-step strategy proposed in this article for determining multiple solutions of a single objective optimization problem. The basic idea in this single-step approach is to search for pareto optimal points that are either better or as good as the current set of the pareto points instead of searching for only strictly better points. As in the previous strategies, the multiobjective optimization problem is converted into a feasibility problem and a single feasible solution is initially determined. The following Type III cut is then successively added after the determination of every single feasible solution.

$$\textbf{Type III} \qquad \left\{\underset{j=1,2,\ldots,m}{\wedge}\left(F_j = F_j^i\right)\right\} \vee \left\{\underset{j=1,2,\ldots,m}{\vee}\left(F_j \lhd F_j^i\right)\right\} \tag{15}$$

The above cut ensures that any successive solution will have to be either better (in any one of the objective) or equal (in all the m objectives) to the previously obtained i solutions. Thus at any stage, this strategy can also determine realizations for the solutions which are pareto optimal amongst the solutions explored so far. The algorithm will converge when the problem either becomes infeasible or the search space gets fully explored. The set of pareto solutions are then obtained by a post optimality analysis which involves sorting the set of obtained solutions for the pareto points and the corresponding realizations.

It is difficult to give specific comparison between the two approaches (for determining the multiple realization of all pareto points) as the performances can be problem dependent. However, some generic observations can be made. The single-step strategy may perform better than the two-step strategy since it does not involve solving the problem twice. However, for problems having a large number of realizations (especially for nonpareto points), the two-step strategy may be better than the single-step strategy. This is because the Type III cuts are weaker than the Type I cuts and can potentially keep exploring the realizations of a nonpareto point.

Remark. It can be seen that the Type III cuts are identical to those in Type II. Despite this similarity, we have separately labeled them since the Type III cuts are added sequentially while all Type II cuts are added simultaneously.

Multiple realizations of any pareto point. In certain cases, the designer may be interested in the multiple realizations only for a single pareto point that she or he has selected based on some (other) criteria and not in the multiple realizations of all the pareto points. These multiple realizations of a single pareto point can be determined in a straight forward fashion without using any of the above two techniques suggested for obtaining multiple realizations. This strategy involves determining all the feasible solutions of the feasibility problem (MF) after adding the following Type IV cut.

$$\textbf{Type IV} \qquad \underset{j=1,2,\ldots,m}{\wedge}\left(F_j = F_j^{\text{Selected}}\right) \tag{16}$$

where F_j^{Selected} indicates the value of the jth objective for the selected pareto point (obtained by sorting the Type I cuts). The Type IV cut can be easily modified to generate multiple realizations of a specified set (and not necessarily a single pareto point) of pareto points as:

$$\underset{\forall i=1\ldots sp}{\vee}\left(\underset{\forall j=1\ldots m}{\wedge}\left(F_j = F_j^i\right)\right) \tag{17}$$

where *sp* indicates the number of selected points.

The following observations can be made regarding the number of cuts added in the various algorithms to determine pareto points.

(1) The total number of Type I and Type II cuts are equal. The total number of Type III cuts will in general be more than the number of Type II (or Type I) cuts though the actual numbers may vary depending on the underlying search strategy.

(2) The upper bound on the total number of Type I or Type II cuts is the total number of distinct feasible points whereas the upper bound on the total number of Type III cuts is the total number (including multiple realizations) of feasible solutions. Similarly, the lower bound on the total number of Type I or Type II cuts is the number of distinct pareto solutions, while the lower bound on the total number of Type III cuts is the total number (including realizations) of pareto optimal solutions.

(3) Some of the T Type II cuts that are added may be redundant. The number of such Type II redundant cuts is equal to the difference between the total number of distinct feasible solutions obtained (T) and the number of distinct pareto-points. These need not be removed since most standard solvers are capable of removing these redundancies during the preprocessing stages. However, if required, the redundant cuts can be easily identified and removed.

(4) The set of Type I cuts in general will be a subset of Type III cuts. Moreover, the last of the Type I and Type III cut will necessarily correspond to a pareto point.

(5) The number of cuts added in these algorithms is strongly dependent on the nature of the underlying search strategy and hence depending on the solver used to solve the problems, the computational time can show significant variations.

(6) It can be noted that all the four types of cuts are disjunctive in nature. Though these can be easily accommodated in the CP modeling framework, if required, they can

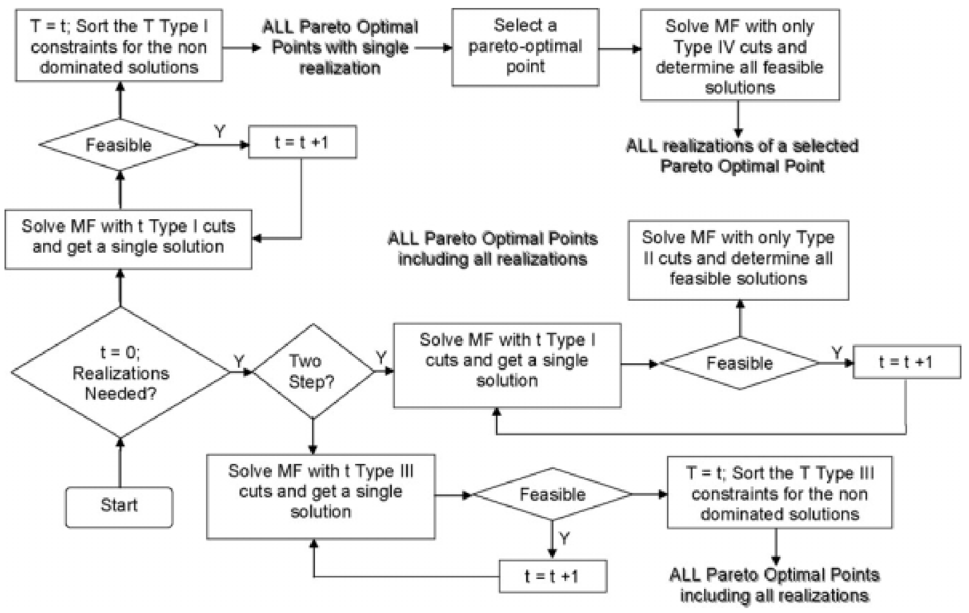

T = t; Sort the T Type I constraints for the non dominated solutions
ALL Pareto Optimal Points with single realization
Select a pareto-optimal point
Solve MF with only Type IV cuts and determine all feasible solutions
ALL realizations of a selected Pareto Optimal Point
Feasible
Y
t = t +1
Solve MF with t Type I cuts and get a single solution
ALL Pareto Optimal Points including all realizations
Solve MF with only Type II cuts and determine all feasible solutions
t = 0; Realizations Needed?
Y
Two Step?
Y
Solve MF with t Type I cuts and get a single solution
Feasible
Y
t = t +1
Start
Solve MF with t Type III cuts and get a single solution
Feasible
T = t; Sort the T Type III constraints for the non dominated solutions
Y
t = t +1
ALL Pareto Optimal Points including all realizations

Figure 4. Various algorithms to determine globally optimum pareto fronts.

also be reformulated as traditional inequalities of the form $g(x) \leq 0$.

All the above four strategies, guaranteed to determine globally optimal pareto front (including nonconvex pareto fronts) for combinatorial optimization problems, are summarized in Figure 4. All these four strategies along with other strategies proposed for single objective optimization problem in this article are guaranteed to converge in a finite number of steps because for any bounded integer programming problem, the set of feasible solutions is finite. However, the time required for convergence may be arbitrarily large depending on the computational complexity of the problem. We believe that the proposed approaches are also generically applicable to nonlinear optimization problems involving continuous and discrete variables subject to the availability of CP solvers that can accommodate continuous variables. We next demonstrate all the strategies proposed in this article on two case studies.

Case Studies

In this section, we present two case studies taken from the sensor network design literature to demonstrate the applicability of the proposed techniques. We provide only a brief description of these case studies and a detailed overview can be obtained from the cited literature. The results reported are generated using the default options of ILOG OPL Studio modeling language and ILOG CP Solver. We will use the parallel coordinate system for visualizing the pareto fronts. Additional details on the parallel coordinate system can be obtained from Kotecha et al.[7] and Bagajewicz and Carbera.[19]

Design of sensor network from a fault diagnosis perspective

The design problem is to determine the variables (along with the number of sensors for each of the selected variable) to be measured in the presence of uncertainties (in available data, such as fault occurrence and sensor failure probabilities) so as to aid in efficient fault diagnosis. Bhushan et al.[20] proposed an explicit MILP formulation with four different objectives namely (i) minimization of nominal system unreliability of detection (U), (ii) maximization of robustness to the uncertainties in the probability data (ϕ), (iii) maximization of the network distribution (N), and (iv) minimization of the sensor network cost (or the maximization of the cost saving (x_s)). However, their work considered these multiple objectives in a lexicographic fashion and hence did not fully evaluate the trade-offs between the various objectives. Subsequently, Kotecha et al.[7] reformulated this problem into an explicit CP model using the expressive power of CP which led to significant reduction in the dimensionality of the optimization problem. This work also determined the pareto optimal solutions for various objectives using a simple strategy which required the determination of all feasible solutions. In the current work, we demonstrate efficient generation of the same pareto optimal solutions with the proposed techniques. In addition, we also demonstrate the utility of the other proposed techniques in generating optimal solutions. An efficient reformulation of the CP-based formulation of Kotecha et al.[7] has been presented in F1. We use the objective function in (18) for demonstrating the benefits of the proposed techniques on a single objective optimization problem while the objective function and constraint in (19) is used for showing the benefits of the proposed techniques in determining globally optimal solutions for a multiobjective optimization problem.

$$\text{Max} \ -[\alpha_1 U - \alpha_2 \phi - \alpha_3 N - x_s] \tag{18}$$

Or

$$\begin{aligned} \text{Max} \quad & [\phi, N, x_s] \\ & U = U^{\text{Optimal}} \end{aligned} \tag{19}$$

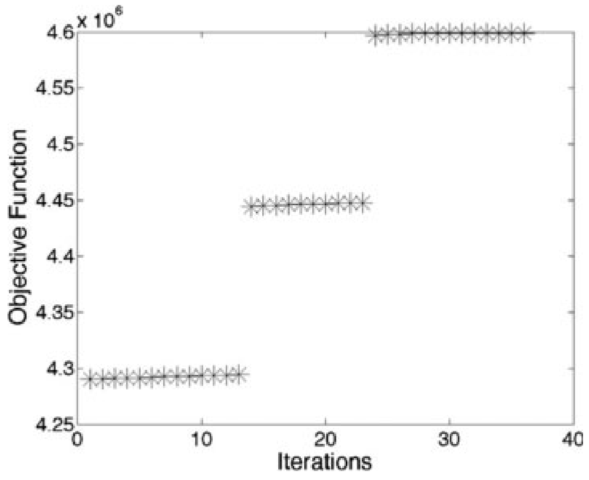


Figure 5. Combined objective function value for determining the multiple solutions for the TE problem.

$$s.t. \sum_{j=1}^{n} c_j x_j + x_s = C^* \tag{20}$$

$$U_i = \log(f_i) + \sum_{j=1}^{n} B_{ij} x_j \log(s_j) + \phi_i, \quad \forall i \in (I_f \cup I_s) \tag{21}$$

$$U_i = \log(f_i) + \sum_{j=1}^{n} B_{ij} x_j \log(s_j), \quad \forall i \in I \backslash (I_f \cup I_s) \tag{22}$$

$$U = \max_{\forall i \in I}(U_i) \tag{23}$$

$$\phi_i^* = \phi_{fi}^*, \quad \forall i \in (I_f \backslash I_s) \tag{24}$$

F1

$$\phi_i^* = \phi_{si}^*, \quad \forall i \in (I_s \backslash I_f) \tag{25}$$

$$\phi_i^* = \phi_{fi}^* + \phi_{si}^*, \quad \forall i \in (I_s \cap I_f) \tag{26}$$

$$\left.\begin{array}{l} \phi_i < \phi_i^* \Rightarrow \tilde{\phi}_i = \phi_i \\ \phi_i \geq \phi_i^* \Rightarrow \tilde{\phi}_i = \phi^* \end{array}\right\}, \quad \forall i \in (I_s \cup I_f) \tag{27}$$

$$\phi = \min(\tilde{\phi}_i) \quad \forall i \in (I_s \cup I_f) \tag{28}$$

$$\phi_{si}^* = -\sum_{j \in J_s} B_{ij} (\log s_j) x_j, \quad \forall i \in I_s \tag{29}$$

$$\phi \leq \phi^* \tag{30}$$

$$N = \sum_{j=1}^{n} \min(x_j, 1) \tag{31}$$

$$(U_i, U) \in \mathbf{Z}^-; \quad (x_j, x_s, \phi, \phi_i^*, \phi_i, \phi_{si}^*, N) \in \mathbf{Z}^+ \tag{32}$$

$$\alpha_1 = (1+\phi^*)(1+C^*)(1+N^*); \quad \alpha_2 = (1+C^*)(1+N^*); \quad \alpha_3 = (1+C^*); \tag{33}$$

In both the references (Bhushan et al.[20] and Kotecha et al.[7]), the optimization formulations were applied on the Tennessee Eastman (TE) case study,[21] which has been considered by other researchers[22] as well for illustrating sensor network design techniques and is considered a benchmark challenge problem in process systems engineering literature. In this article also, we apply the various techniques on TE problem which is briefly discussed next.

TE Problem. The TE flowsheet has 50 potential measurements and 33 faults (16 bidirectional and one unidirectional). The cost of the sensors, the sensor failure probability, and fault occurrence probability data are taken from Bhushan et al.[20] and are the same as used by Kotecha et al.[7] For the sake of brevity, further details of the TE problem are not presented here and the interested reader is referred to Downs and Vogel.[21] Similar to literature (Bhushan et al.[20] and Kotecha et al.[7]), we have considered that the sensor failure probabilities of sensors 3 and 4 and occurrence probabilities of faults 1 and 9, are not known accurately. We present the results for the case when the maximum available monetary resource (C*) is 1000 units as in Kotecha et al.[7] Additionally, we have also compared the results for a monetary resource of 5500 units but have not presented them for the sake of brevity. For this case study, there are 69 integer variables and 36 constraints.

Determination of Multiple Solutions. Figure 5 shows the progression of the objective function, for the lexicographic objective stated in (18), towards optimality. It can be seen that the optimal solution of 4,598,594 is reached in the 27th iteration and there are 10 realizations for this globally optimum solution. The progress of the individual objectives is shown in Figure 6. The optimal unreliability is −2 while the maximum robustness is 6 with a network distribution of 4 whereas the monetary resource of 1000 units is fully utilized with no cost savings. The designer can choose from these 10 solutions based on some other objective which has not been considered in the optimization formulation in F1. For example, the designer can prefer a solution which involves measuring the least number of corrosive streams. Thus, the knowledge of multiple solutions offers additional flexibility by enabling the designer to incorporate or consider other additional design criteria.

Determination of K*-Best Feasible Solutions.*

Determination of nondistinct K*-best solutions.* Figures 7 and 8 show the determination of five best nondistinct solutions. It can be seen that as the number of iterations increase, the value of the five best solutions keeps increasing. These five best solutions are determined in 126 iterations. In this case, the

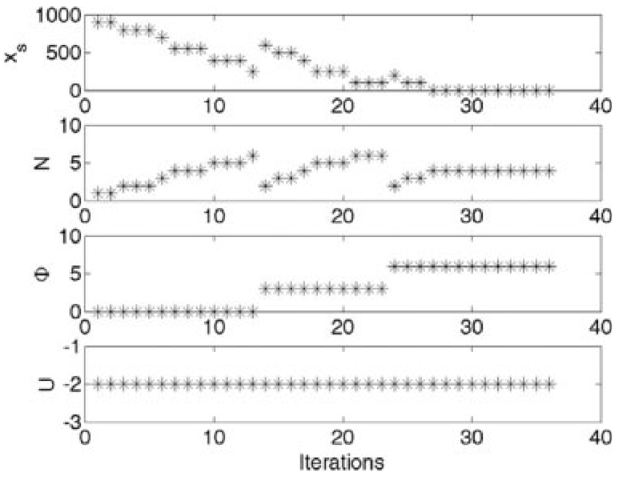


Figure 6. Individual objectives for determining multiple solutions for the TE problem.

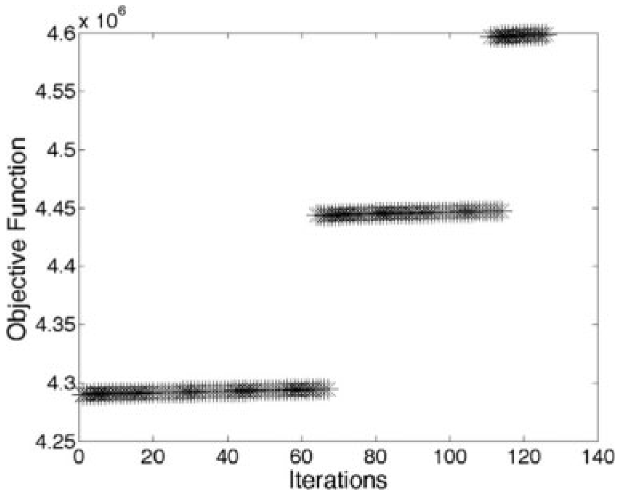


Figure 7. Combined objective function for determining nondistinct *K*-best feasible solutions for TE problem (*K* = 5).

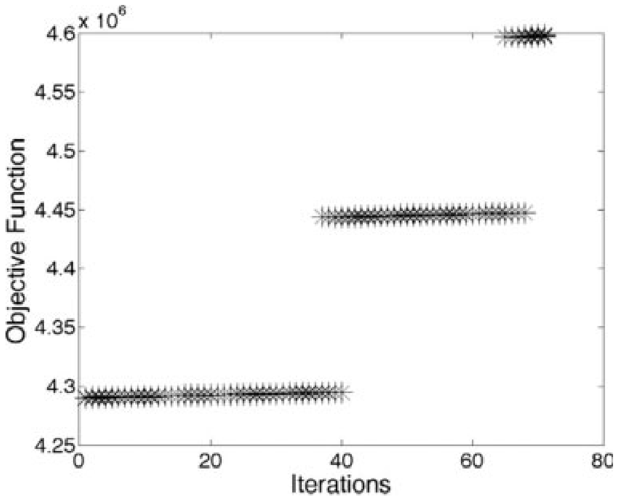


Figure 9. Combined objective function for determining distinct *K*-best feasible solutions for TE problem (*K* = 5).

final five best solutions are also equal to the globally optimal solution of 4,598,594 and the individual objectives correspond to those already mentioned. This is expected since there are 10 multiple optimal solutions as seen in Figure 5.

Determination of distinct K-*best solutions.* Figure 9 shows the determination of five best distinct solutions. It takes 71 iterations to determine these five best distinct feasible solutions. These solutions are [B1: 4,596,792, B2: 4,597,593, B3: 4,597,643, B4: 4,597,693, B5: 4,598,594]. Figure 10 shows the individual objective function values and it can be seen that an unreliability value of −2 is maintained in all the 71 iterations. The five best distinct feasible solution in terms of individual objectives are [B1: (−2,6,2,200), B2: (−2,6,3,0), B3: (−2,6,3,50), B4: (−2,6,3,100), and B5: (−2,6,4,0)]. As expected, the best of these five solutions also corresponds to the global optimal solution. It can be seen that among these solutions, there is no deterioration in the unreliability and robustness to the uncertainties in the probability data. The deterioration in the two best solutions (B5 and B4) occurs in the network distribution. The deterioration across solutions B4, B3, and B2 occurs in the cost saving. This behavior is expected due to the assumed precedence ordering which gives highest importance to the network unreliability followed by the network robustness, network distribution and finally the cost saving.

Hence, it can be seen that the study of *K*-best feasible points gives the designer an insight about the potential deterioration in the objective function that would have to be tolerated in the case of some unforeseen circumstances.

Determination of Pareto Optimal Solutions. We now present results for the determination of pareto optimal front by using the proposed strategies.

Single realizations. Figure 11 shows the objective function value for the Type I cuts obtained in the determination of pareto optimal solution using the proposed algorithm. It can be seen that 31 Type I cuts are generated through this algorithm. A total of 17 pareto points are obtained by sorting

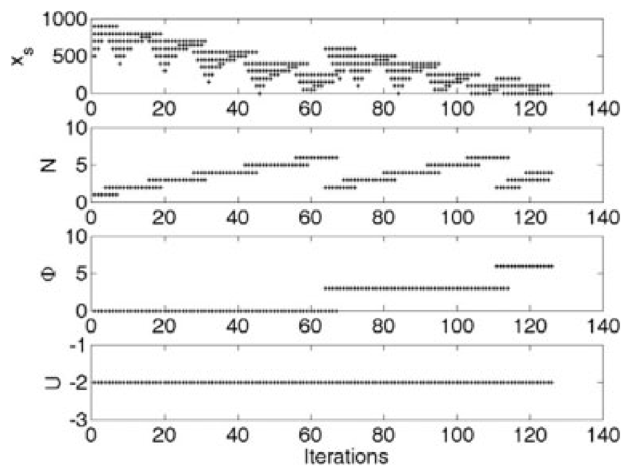


Figure 8. Individual objectives for determining nondistinct *K*-best feasible solutions for TE problem (*K* = 5).

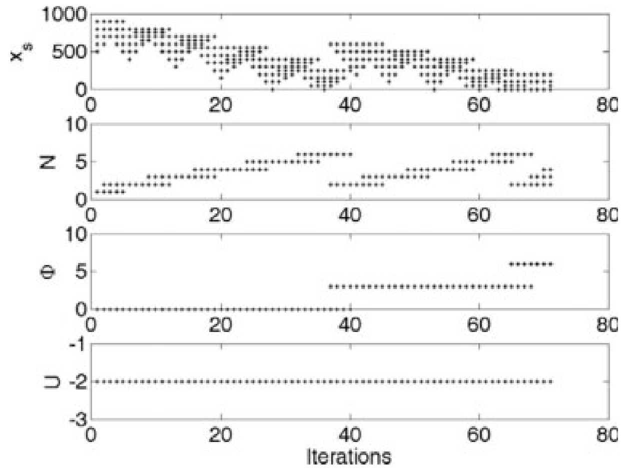


Figure 10. Individual objectives for determining distinct *K*-best feasible solutions for TE problem (*K* = 5).

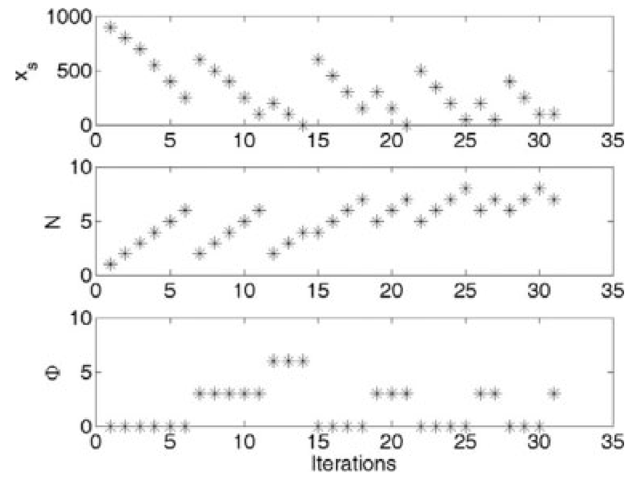


Figure 11. Objective function values for Type I cuts used for determining pareto front and realizations using two-step strategy for TE problem.

Table 1. Details of Pareto Optimal Points for the TE Case Study

Tag	Pareto Points ϕ	Pareto Points N	Pareto Points x_s	No. of Multiple Pareto Solutions
A	0	1	900	2
B	0	2	800	9
C	3	2	600	1
D	6	2	200	1
E	0	3	700	16
F	3	3	500	5
G	6	3	100	5
H	0	4	600	14
I	3	4	400	10
J	6	4	0	10
K	0	5	500	6
L	3	5	300	10
M	0	6	400	1
N	3	6	200	5
O	0	7	250	3
P	3	7	100	1
Q	0	8	100	3

the Type I cuts (the corresponding objective function values) and these pareto points are shown in Figure 12 (also in Table 1). However, this technique determines only one realization for each of the pareto optimal point.

Multiple realizations. (a) Two-step strategy: The two-step strategy also involves the construction of the Type I cuts as discussed in the earlier section. This is followed by the construction of Type II cuts with the information in Type I cuts. Thus, a total of 31 Type II cuts are constructed. The Type II cuts are appended to the feasibility problem in MF and all the feasible solutions are determined. This set of all feasible solutions corresponds to the set of all the pareto optimal points with all their possible realizations. In this case, there are 102 pareto optimal solutions. These pareto optimal solutions along with the corresponding number of realizations have been reported in Table 1. It can be seen that the pareto optimal point represented by the Tag E has the maximum number of realizations (16) whereas the points C, D, M, and P have only one realization each. It can also be inferred that 14 (difference between the number of Type I cuts and number of distinct pareto points) of the Type II cuts are redundant.

(b) Single-step strategy: Figure 13 shows the details of the 171 Type III cuts that get added while determining the pareto optimal solutions in a single step. As discussed earlier, the sorting of these 171 Type III cuts will lead to the pareto optimal points along with their realizations. The pareto

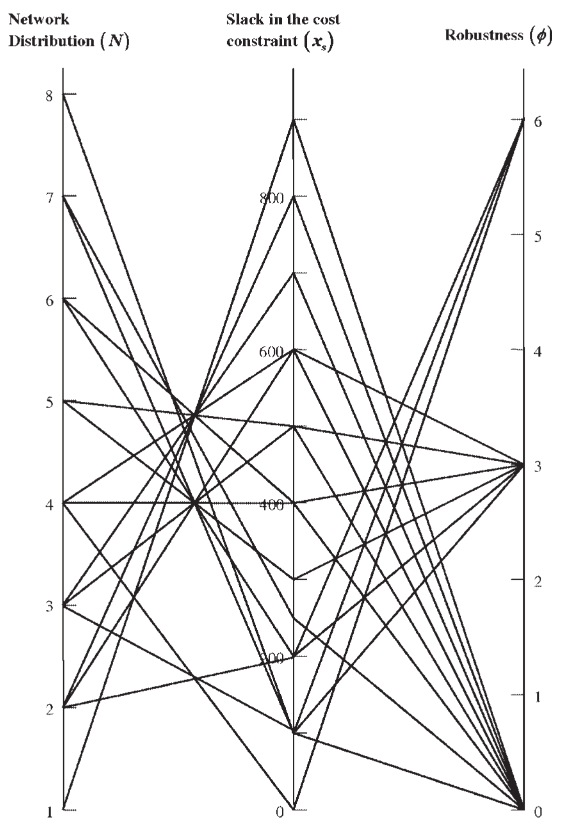


Figure 12. Pareto front for TE problem.

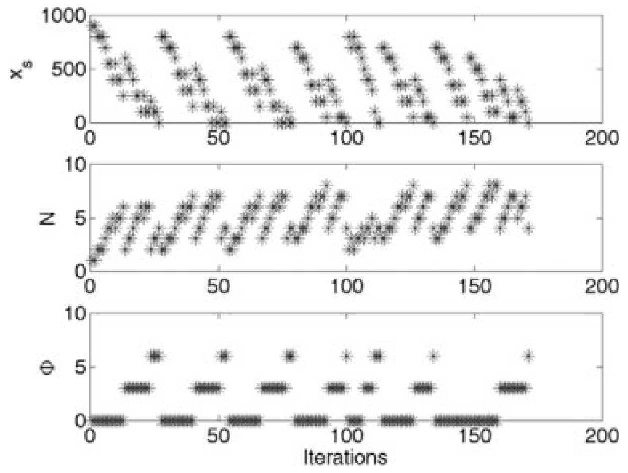


Figure 13. Objective function values for Type III cuts used for determining pareto front and realizations using single-step strategy for TE problem.

optimal points obtained by this algorithm are the same as those obtained in the two-step algorithm. It can be seen that a total of 69 (=171 − 102) Type III cuts are redundant.

The determination of realizations of a single pareto point is straight forward and can be easily implemented with the Type IV cuts. For the sake of brevity, results for this case are not presented in this article.

Design of sensor network from a reliability perspective

This design problem involves the placement of sensors for a linear overall mass flow process, to ensure nonredundant sensor network design. Kotecha et al.[11] proposed a cutset-based framework that can been used to unify objectives like reliability and precision. In particular, we consider three different objectives namely (i) minimization of the network failure rate (λ), (ii) minimization of network variance (σ^2), and (iii) minimization of the network cost (C) subject to the criterion of an observable network. Towards this end, we propose an explicit nonlinear IP formulation as shown in F2. For the sake of simplicity, we have ignored the uncertainty in the sensor failure rate data and the sensor variances. However, the uncertainty to these data can be included on the lines of Kotecha et al.[11] We will use the objective function in (34) for demonstrating the benefits of the proposed techniques on a single objective optimization problem while the objective function in (35) will be used for showing the benefits of the proposed technique to determine globally optimal solutions for a multiobjective problem.

$$\min \quad \beta_1\lambda + \beta_2\sigma^2 + C \tag{34}$$

Or

$$\min \quad \left[\lambda, \sigma^2, C\right] \tag{35}$$

$$\sigma^2 = \max\left(\hat{\sigma}_1^2, \hat{\sigma}_2^2 \ldots \hat{\sigma}_n^2\right) \tag{36}$$

$$\lambda = \max\left(\hat{\lambda}_1, \hat{\lambda}_2 \ldots \hat{\lambda}_n\right) \tag{37}$$

$$\sum_{i \in E'} x_i \leq |S| - 1, \qquad \forall S \subset V, \quad |S| > 2 \tag{38}$$

$$\sum_{i \in N} (1 - x_i) = n - m \tag{39}$$

$$\hat{\sigma}_i^2 = (1 - x_i)\sigma_i^2 + x_i \sum_{k=1}^{|C^i|} \left[\left(\sum_{\substack{j \in C_k^i, \\ j \neq i}} \sigma_j^2 \right) \left(\prod_{\substack{j \in C_k^i, \\ j \neq i}} (1 - x_j) \right) \right] \quad \forall i \in N \tag{40}$$

$$\hat{\lambda}_i = (1 - x_i)\lambda_i + x_i \sum_{k=1}^{|C^i|} \left[\left(\sum_{\substack{j \in C_k^i, \\ j \neq i}} \lambda_j \right) \left(\prod_{\substack{j \in C_k^i, \\ j \neq i}} (1 - x_j) \right) \right] \quad \forall i \in N \tag{41}$$

Table 2. Data for the "Retrofit Design" Problem

Variable i	Sensor Failure Rate λ_i	Sensor Variances σ_i^2	Sensor Cost c_i
1	13	10	19
2	19	9	17
3	21	12	13
4	25	13	12
5	10	14	25
6	20	18	10
7	15	22	7
8	19	14	6
9	12	16	15
10	10	18	13
11	11	12	17
12	19	19	13
13	16	10	12
14	17	12	12
15	19	16	17
16	28	20	19
17	14	20	17
18	19	14	18
19	14	18	17
20	11	19	15
21	20	13	15
22	22	15	13
23	11	10	13
24	16	11	13

$$C = \sum_{i \in N} c_i(1 - x_i) \tag{42}$$

$$x_j \in \{0, 1\} \quad \forall j \in N \tag{43}$$

$$\sum_{i \in C^t} x_i \geq 1 \quad \forall t = 1, 2, .., T \tag{44}$$

$$\beta_1 = (1 + \sigma^{2\mathrm{u}})(1 + C^{\mathrm{u}}); \quad \beta_2 = (1 + C^{\mathrm{u}}) \tag{45}$$

$$\lambda, \sigma^2, C, \hat{\sigma}_i^2, \hat{\lambda}_i \in \mathbf{Z}^+$$

In the above formulation, $x_j = 1$ indicates that the jth variable is not measured whereas $x_j = 0$ indicates that the jth variable is measured.

Retrofit Design Problem. This flowsheet has been widely discussed in literature[19,23–25] and has 24 variables and 11 equations (overall mass balances corresponding to 11 units). This flowsheet was initially proposed as a retrofit design problem with variables 11–24 already being measured and the decision variables were the placement of sensors on streams 1–10. However, we consider grass root design, and hence will assume that none of the streams are already measured. Hence, there are $^{24}C_{11}$ combinations for choosing 13 variables to be measured for a nonredundant sensor network design. The cost of the sensors, failure rates and variances are given in Table 2. There are 24 binary variables, 51 integer variables and 2160 constraints for this case study.

Determination of Multiple Solutions. Figure 14 shows the optimal solution and the number of multiple optimal solutions for minimizing the combined objective shown in (34). It can be seen that the optimal solution of 4,779,172 is achieved in the third iteration and there are only two realizations for this globally optimum solution. The progress of the individual objectives

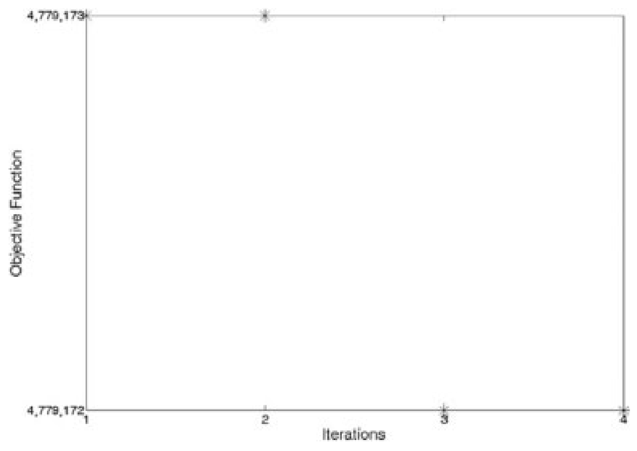


Figure 14. Combined objective function for determining multiple solutions for the "Retrofit design" problem.

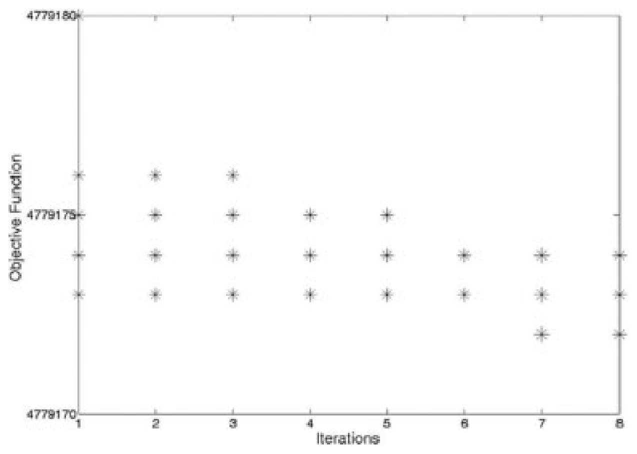


Figure 16. Combined objective function for determining nondistinct *K*-best feasible solutions for "Retrofit design" problem ($K = 5$).

is shown in Figure 15. The optimal values for the network failure rate, network variance and network cost are 92, 80, and 172, respectively. The sensor networks (measured variables) corresponding to these two realizations are [2,3,7,8,10,12,14,16, 18,21,22,23,24] and [2,3,7,8,10,12,13,16,18,21,22,23,24]. It can be seen that the only difference between these two sensor networks is in the measurement of variables 13 and 14. Hence, it can be inferred that the sensors measuring the streams [2,3,7,8,10,12,16,18,21,22,23,24] are critical as a failure in any of them will lead to a situation wherein the optimal solution of 4,779,172 cannot be maintained. The designer can choose any of these two solutions based on some other objective. Thus, we see that knowledge of multiple solutions offers additional flexibility and insight into the design problem.

Determination of K-*best Feasible Solutions.* As mentioned earlier, in addition to knowing the global optima, it is also useful to know the next best alternative solutions and the deterioration of the objective function that needs to be tolerated for each of these solutions. In the following, we consider the case of distinct and nondistinct solutions for this problem.

Determination of nondistinct K-*best solutions.* The combined objective for determining the five best nondistinct solutions is shown in Figure 16 while the individual objectives are depicted in Figure 17. These five best solutions are determined in eight iterations. In all these iterations, the network failure rate value is 92 and the network variance value is 80. As mentioned earlier, it can be seen that the number of distinct solutions need not have a monotonic behavior with the number of iterations. For example, the number of distinct solutions in 5th iteration is 3 but it decreases to 2 in the next iteration. However in the 7th iteration, the number of distinct solutions again increases to 3. The five best nondistinct feasible solutions are [4,779,174, 4,779,173, 4,779,173, 4,779,172, and 4,779,172]. As expected, both the multiple solutions obtained in the determination of multiple optimal solutions are also obtained as the two best solutions.

Determination of distinct K-*best solutions.* The five best distinct feasible solutions are shown in Figure 18 and are determined in four iterations. These solutions are [4,779,176,

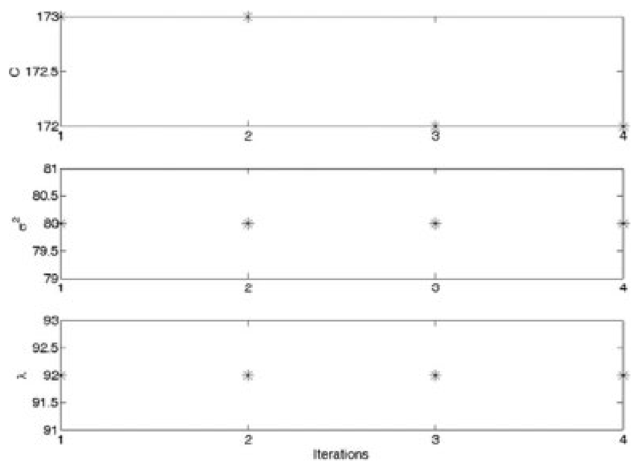


Figure 15. Individual objectives for determining multiple solutions for the "Retrofit design" problem.

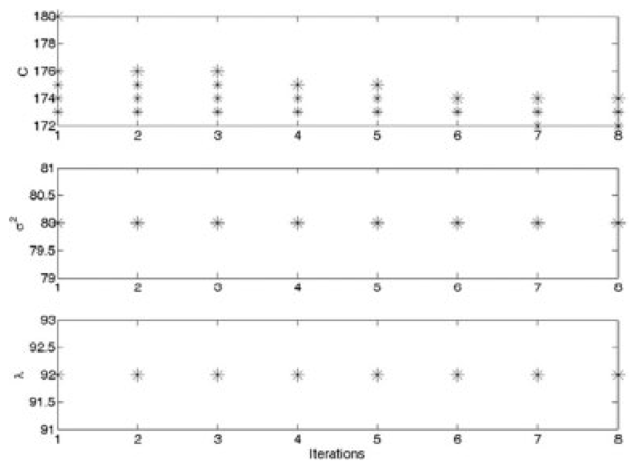


Figure 17. Individual objectives for determining nondistinct *K*-best feasible solutions for "Retrofit design" problem ($K = 5$).

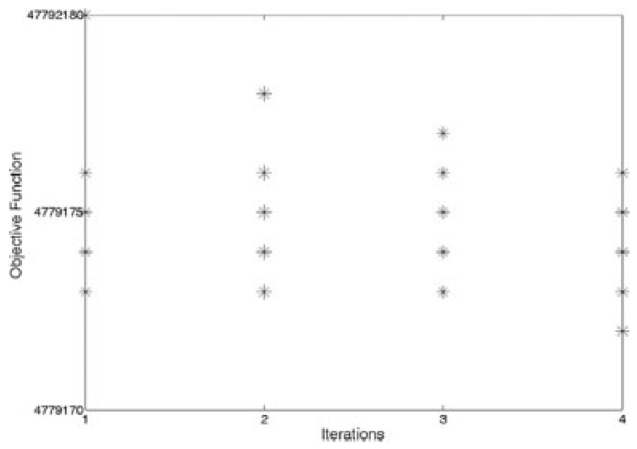


Figure 18. Combined objective function for determining distinct *K*-best feasible solutions for "Retrofit design" problem ($K = 5$).

4,779,175, 4,779,174, 4,779,173, and 4,779,172]. Figure 19 shows the individual objective function values and it can be seen that the network failure rate is 92 and the network variance is 80 in all the four iterations. The five best distinct feasible solutions in terms of individual objectives are [(92,80,176), (92,80,175), (92,80,174), (92,80,173), and (92,80,172)]. As mentioned earlier, the best of these five solutions also corresponds to the global optimal solution. It can also be seen that the assumed precedence in the lexicographic objective function ensures that the deterioration of the primary objective is the least.

Hence, it can be seen that the study of *K*-best feasible solutions gives the designer an insight about the potential deterioration in the objective function that would have to be tolerated in the case of some unforeseen circumstances.

Determination of Pareto Optimal Solutions. We now present results for the determination of pareto optimal front by using the proposed strategies.

Single realizations. Figure 20 shows the objective function value for the Type I cuts obtained during the determination of pareto optimal solution using the proposed algorithm. It can be seen that 31 Type I cuts are generated during the exploration of the search space. A total of 12 pareto points are obtained by sorting the Type I cuts and these pareto points are shown in Figure 21 (also in Table 3). Hence, it can also be inferred that 19 of the Type I cuts are redundant.

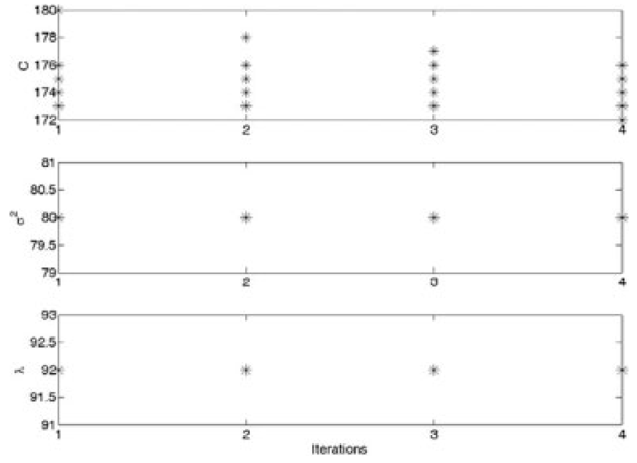


Figure 19. Individual objectives for determining distinct *K*-best feasible solutions for "Retrofit design" problem ($K = 5$).

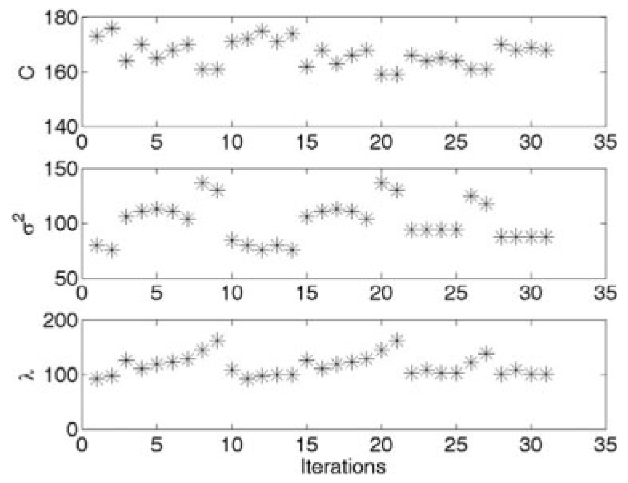


Figure 20. Objective function values for Type I cuts used for determining pareto front and realizations using two-step strategy for "Retrofit design" problem.

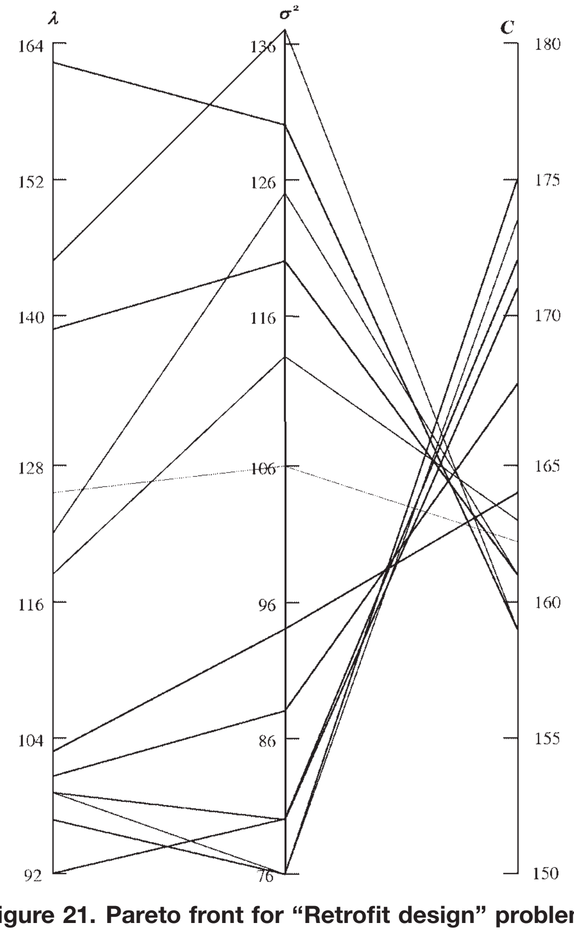


Figure 21. Pareto front for "Retrofit design" problem.

Table 3. Details of Pareto Optimal Points for the Retrofit Case Study

Tag	λ	σ^2	C	No. of Multiple Pareto Solutions
A1	92	80	172	2
B1	97	76	175	2
C1	100	76	174	2
D1	100	80	171	2
E1	101	88	168	2
F1	103	94	164	2
G1	119	113	163	2
H1	122	125	161	2
I1	126	106	162	4
J1	139	118	161	2
K1	145	137	159	2
L1	162	130	159	2

However, this technique determines only one realization for each of the pareto optimal point.

Multiple realizations. (a) Two-step strategy: The two-step strategy also involves the construction of the Type I cuts as discussed in the above section. This is followed by the construction of Type II cuts with the information obtained in Type I cuts. Thus, a total of 31 Type II cuts are constructed. The Type II cuts are appended to the feasibility problem in MF and all the feasible solutions are determined. This extended feasibility problem has 26 solutions which are the pareto optimal solutions. These pareto optimal solutions along with the corresponding number of realizations have been reported in Table 3. It can be seen that all the pareto optimal points except I1 are characterized by two realizations each. Pareto point I1 is however characterized by four realizations and hence offers higher flexibility to the designer.

(b) Single-step strategy: Figure 22 shows the details of the 66 Type III cuts that get added while determining the pareto optimal solutions in a single step. As discussed earlier, the sorting of these 66 Type III cuts will lead to the pareto optimal points along with their realizations. The pareto optimal points obtained by this algorithm are the same as those obtained in the two-step algorithm. It can also be noted that the last Type I and Type III cut ($\lambda = 101$; $\sigma^2 = 88$; $C = 168$) correspond to the pareto optimal point E1.

Thus, we have shown the applicability of the proposed strategies on two case studies from the literature. Application of CP-based approaches for determining multiple solutions and evaluating them for multiple objectives for a batch scheduling problem can also be found in literature.[26] We now briefly discuss the computational efficiency of the proposed algorithms.

Computational Performance

In this section, we briefly describe the implementation issues and computational performance of the proposed algorithms. The proposed techniques are implemented on a 3.20 GHz Pentium IV CPU with 1 GB RAM and using Windows XP Operating System. The CP solver used is SOLVER from ILOG with ILOG OPL Studio as the modeling language. Before presenting the results, we discuss some of the implementation issues related with the OPL Studio.

(1) All the algorithms proposed in this article require dynamic addition of cuts. However, to the best of our knowledge, these cuts cannot be added in a truly dynamic fashion (i.e., adding a constraint after determining a single solution and continuing the exploration of the rest of the solution space) in the current version of ILOG OPL Studio, without resetting (re-solving) the model. For demonstrating proof of the concept, without loss of generality, we chose to re-solve the models after the addition of the appropriate cuts in the various proposed algorithms. As the problems are re-solved, we incorporate the following additional cut (after determining a feasible solution) to avoid revisiting the previously obtained solution.

$$\bigwedge_{i=1,2,\ldots,n-1} \left(x_i = x_i^k\right) \Rightarrow \left(x_n \neq x_n^k\right) \tag{46}$$

where n indicates the number of decision variables and x_i^k indicates the value of the ith decision variable at the kth iteration. It can be seen that the cut in (46) is effective irrespective of the decision variables being integer or binary and can be directly incorporated because of the rich expressive modeling power of CP. All the computational times reported in this article involve such an implementation (re-solving) and hence these reported times may not always accurately indicate the efficiency of the proposed algorithms.

(2) As mentioned earlier, most of the proposed strategies solve an infeasible problem at the end of the algorithm and to the best of our knowledge, ILOG OPL Studio does not report the time used for determining the infeasibility of a model. In view of this, we have reported the actual clock time in sec. (which is usually greater than the CPU time) used by the solver (including the time spent on identifying the infeasibility) and hence these times should not be considered as the time taken for computations.

We have preferred to compare the proposed algorithms even in the presence of the above implementation issues so as to aid the reader in appreciating the efficiency of the proposed algorithms.

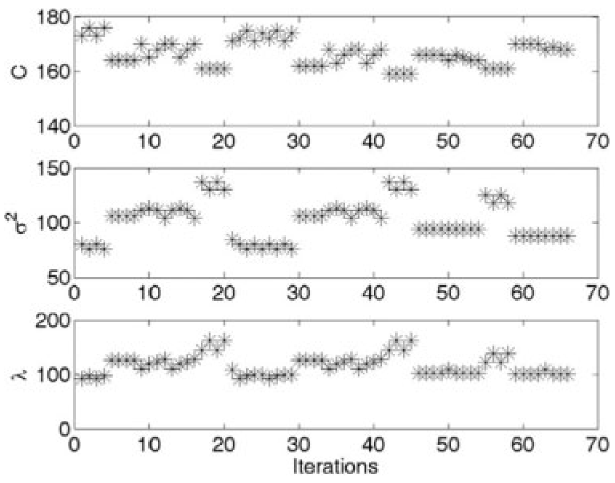


Figure 22. Objective function values for Type III cuts used for determining pareto front and realizations using single-step strategy for "Retrofit design" problem.

Table 4. Computational Performance of the Various Algorithms

	Determination of Multiple Solutions		Determination of K-Best Feasible Solutions		Determination of Pareto Points with Realizations		
Case Study	Two Step [7]	Single Step	Distinct	Non Distinct	All Feasible Solutions [7]*	Two Step Strategy	Single Step Strategy*
TE case study	2	2	4	2	907	3	285
Retrofit case study	239	141	156	149	794	729	900

*Time for sorting not included.

Determination of multiple optimal solutions

Table 4 shows the comparison of the time required for determining the multiple solutions by the two-step strategy proposed in literature and the single-step strategy proposed in this article. It can be seen from Table 4 that the time required for the TE case study in the proposed single-step strategy is the same as the two-step strategy proposed in literature. This is despite the fact that the current implementation (as discussed earlier) involves re-solving of the problem (36 times in this case study to determine all the 10 realizations) and this may be contributing substantially to the computational effort. On the other hand, for the retrofit case study the proposed single-step strategy performs significantly better than the two-step strategy in spite of re-solving the problem. The cut added in this case study is as presented in (46) that prevents the same solution from being revisited. Thus, we can see that the proposed single-step strategy can be computationally efficient than the two-step strategy available in literature. Moreover, the single-step strategy does not require any manual intervention.

Determination of K*-best solutions*

The time required for determining the *K*-best feasible solutions (distinct and nondistinct) for both the case studies using the proposed strategies are given in Table 4. There has not been any attempt in literature to determine *K*-best feasible solutions for these problems and hence no comparison is being made. The number of times the problems are re-solved can be obtained from the number of cuts added (as discussed previously).

Determination of pareto optimal solutions

Table 4 shows the time required by three different algorithms to determine the pareto optimal solutions. For the TE problem, it can be seen that the two-step strategy proposed in this article is better than the existing strategy of determining all feasible solutions in spite of the implementation drawback. In the two-step strategy for determining the pareto solutions, the pareto front (without realizations) is obtained in $\sim$1 sec (involving re-solving the problem 31 times) and the remaining time is spent in determining the realizations via the second step. For determining the pareto solutions with a single-step strategy, the problem is re-solved 171 times (number of Type III cuts) which could be a major reason for the relatively larger amount of time taken. The results for the retrofit problem show that the two-step strategy is marginally better than the strategy of determining all feasible solutions in spite of re-solving the problem 32 times. The single-step strategy to determine all pareto solutions involves re-solving the problem 66 times and takes 900 sec. In this section, we have attempted a comparison of the algorithms based on their computation times on specific problems; however, a more realistic comparison that considers the earlier mentioned implementation issues as well is necessary to further demonstrate the relative efficacies of the algorithms proposed in this article.

Conclusion

In this work, we have presented efficient CP-based strategies to determine the multiple optimal solutions and also to determine both distinct and nondistinct *K*-best feasible solutions for a single objective optimization problem. In this article, we have also presented strategies that can not only determine globally optimal pareto fronts for multiobjective optimization but can additionally determine all the possible realizations, if necessary. None of these strategies require re-solving the optimization problem for the discovery of each potential point and hence can lead to potential computational benefits. These techniques have been demonstrated on two case studies reported in literature. These strategies supplement the other advantages of CP and make it a potential candidate for solving a wide range of combinatorial optimization problems.

Acknowledgments

The authors would like to acknowledge Dr. K.P. Madhavan, Professor-Emeritus, IIT Bombay for the mentoring and guidance in each of the broad facets of process systems engineering, and in particular for directing us to the exciting area of constraint programming.

Notation

m =	no. of objective functions
F_j^t =	value of the j^{th} objective function at the t^{th} iteration
F_j =	value of the j^{th} objective function
K =	user specified parameter to determine K-best feasible solutions
Ksol =	set of K solutions
T =	number of Type I cuts
sp =	number of selected points

Design of sensor network from a fault diagnosis perspective

B_{ij} =	(i,j)th element of bipartite matrix B
c_j =	cost of j^{th} sensor
C^* =	total cost allowed for sensor location
f_i =	fault occurrence probability of i^{th} fault
I =	set of all faults
I_{f} =	set of faults whose occurrence probability is uncertain
I_{s} =	set of faults that affect variables which can only be measured by sensors with uncertain failure probability.
J_{s} =	set of indices of inaccurate sensors
N =	number of variables measured in the process
n =	number of variables in the process
s_j =	sensor failure probability of j^{th} sensor
U =	network unreliability

U_i =	unreliability of i^{th} fault
x_j =	number of sensors to measure j^{th} variable
x_s =	slack variable in the cost constraint
$\alpha_1,\alpha_2,\alpha_3$ =	positive lexicographic constants
ϕ =	maximum of all the individual slacks
$\phi_i, \tilde{\phi}_i$ =	actual, maximum meaningful value of the slack for the i^{th} inaccurate fault

Design of sensor network from a reliability perspective

G =	graph with V nodes and E edges
N =	set of variables
V,E =	set of all nodes and edges in graph G
c_i,C =	cost of i^{th} sensor, total utilized cost
σ^{2u}, C^u =	upper bounds on network variance and network cost
C^i =	set of all the cutsets involving the i^{th} variable
S,E' =	set of nodes $S \subset V$, edges in S
n,m =	number of edges/variables/streams, number of nodes/units excluding the environment node
x_i =	binary variable to indicate the measurement of variable i
β_1, β_2 =	positive lexicographic constants
$\hat{\sigma}_i^2, \hat{\lambda}_i$ =	variance, failure rate of the i^{th} variable
σ^2, λ =	network variance, network failure rate

Literature Cited

1. Biegler LT, Grossmann IE. Retrospective on optimization. *Comput Chem Eng*. 2004;28:1169–1192.
2. Grossmann IE, Biegler LT. Part II. Future perspective on optimization. *Comput Chem Eng*. 2004;28:1193–1218.
3. Raman R, Grossmann IE. Symbolic integration of logic in mixed-integer linear programming techniques for process synthesis. *Comput Chem Eng*. 1993;17:909–927.
4. Hooker J. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. New York: Wiley, 2000.
5. Lustig IJ, Puget JF. Program does not equal program: constraint programming and its relationship to mathematical programming. *Interfaces*. 2001;31:9–53.
6. Tawarmalani M, Sahinidis NV. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Norwell, MA: Kluwer Academic Publishers, 2002.
7. Kotecha PR, Bhushan M, Gudi RD. Constraint proramming based robust sensor network design. *Ind Eng Chem Res*. 2007;46:5985–5999.
8. Deb K. *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY: Wiley, 2001.
9. Tarafder A, Rangaiah GP, Ray AK. A study of finding many desirable solutions in multiobjective optimization of chemical processes. *Comput Chem Eng*. 2007;31:1257–1271.
10. Roe B, Papageorgiou LG, Shah N. A hybrid MILP/CLP algorithm for multipurpose batch process scheduling. *Comput Chem Eng*. 2005; 29:1277–1291.
11. Kotecha PR, Bhushan M, Gudi RD. Design of robust, reliable sensor networks using constraint programming. *Comput Chem Eng*. 2008; 32:1963–2168.
12. Darby-Dowman K, Little J, Mitra G, Zaffalon M. Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem. *Constraints*. 1997;1:245–264.
13. Proll L, Smith B. Integer linear programming and constraint programming approaches to a template design problem. *INFORMS J Comput*. 1998;10:265–275.
14. Smith BM, Brailsford SC, Hubbard PM, Williams HP. The progressive party problem: integer linear programming and constraint programming compared. *Constraints*. 1996;1:119–138.
15. Heipcke S. Comparing constraint programming and mathematical programming approaches to discrete optimisation—the change problem. *Ann Oper Res*. 1999;50:581–595.
16. Maravelias CT, Grossmann IE. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Comput Chem Eng*. 2004;28:1921–1949.
17. Jain V, Grossmann IE. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS J Comput*. 2001;13: 258–276.
18. Kotecha PR, Bhushan M, Gudi RD. Constraint Programming based multi-objective sensor network design for fault diagnosis. Presented at *17th European Symposium on Computer Aided Process Engineering—ESCAPE17*, Bucharest, Romania, 2007.
19. Bagajewicz M, Cabrera E. Pareto optimal solutions visualization techniques for multiobjective design and upgrade of instrumentation networks. *Ind Eng Chem Res*. 2003;42:5195–5203.
20. Bhushan M, Narasimhan S, Rengaswamy R. Robust sensor network design for fault diagnosis. *Comput Chem Eng*. 2008;32: 1067–1084.
21. Downs JJ, Vogel EF. A plant-wide industrial-process control problem. *Comput Chem Eng*. 1993;17:245–255.
22. Musulin E, Benqlilou C, Bagajewicz MJ, Puigjaner L. Instrumentation design based on optimal Kalman filtering. *J Process Control*. 2005;15:629–638.
23. Bagajewicz M, Cabrera E. New MILP formulation for instrumentation network design and upgrade. *AIChE J*. 2002;48:2271–2282.
24. Madron F, Veverka V. Optimal selection of measuring points in complex plants by linear models. *AIChE J*. 1992;38:227–236.
25. Meyer M, Le Lann JM, Koehret B, Enjalbert M. Optimal selection of sensor location on a complex plant, using a graph oriented approach. *Comput Chem Eng*. 1994;18:535–540.
26. Kotecha PR, Kapadi MD, Bhushan M, Gudi RD. On the determination of multiple solutions for batch scheduling using constraint programming. *AIChE Annual Meeting*, Utah, 2007.

Manuscript received July 9, 2008, and revision received Apr. 27, 2009.