Stochastic Mixed Integer Nonlinear Programming Using Rank Filter and Ordinal Optimization

Chengtao Wen and B. Erik Ydstie
Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

Xiaoyan Ma
Information Systems Management, Heinz College, Carnegie Mellon University, Pittsburgh, PA 15213

DOI 10.1002/aic.11901
Published online August 13, 2009 in Wiley InterScience (www.interscience.wiley.com).

A rank filter algorithm is developed to cope with the computational-difficulty in solving stochastic mixed integer nonlinear programming (SMINLP) problems. The proposed approximation method estimates the expected performance values, whose relative rank forms a subset of good solutions with high probability. Suboptimal solutions are obtained by searching the subset using the accurate performances. High-computational efficiency is achieved, because the accurate performance is limited to a small subset of the search space. Three benchmark problems show that the rank filter algorithm can reduce computational expense by several orders of magnitude without significant loss of precision. The rank filter algorithm presents an efficient approach for solving the large-scale SMINLP problems that are nonconvex, highly combinatorial, and strongly nonlinear. © 2009 American Institute of Chemical Engineers *AIChE J,* 55: 2873–2882, 2009

Keywords: stochastic programming, MINLP optimization, rank filter algorithm, ordinal optimization, approximation algorithm, process system synthesis

Introduction

Stochastic mixed integer nonlinear programming (MINLP) is an important and fundamental problem in the area of process system engineering. It presents a natural approach of formulating problems where it is necessary to simultaneously optimize the structure (discrete variables) and parameters (continuous variables) of a process system under uncertain supply/demand conditions. Stochastic MINLPs have found many applications in process industry, e.g., process system synthesis,[1–3] batch processing,[4–6] process control,[7] flexibility analysis,[8] and optimal design of transmission networks.[9,10] The needs in such diverse areas have motivated the development of efficient general purpose algorithms for handling the large-scale, nonconvex, highly combinatorial, and strongly nonlinear problems.

Stochastic MINLP problems combine the combinatorial nature of mixed integer programming (MIP) with the intrinsic difficulty of nonlinear program (NLP) under uncertainty. The methods to solve such problems fall into two categories: numerical integration and sample averaging.

The numerical integration methods utilize numerical integration to calculate multidimensional integrals, e.g., Gaussian Quadratures or Cubatures method.[3,10] Clay and Grossmann[11] develop a discretization scheme, in which the continuous probability distribution functions are discretized and the lower order moments of the original distributions are preserved. The parameters in the resulting discrete distribution are obtained

Correspondence concerning this article should be addressed to C. Wen at chengtao@andrew.cmu.edu or E. Ydstie@cmu.edu by using Gaussian integration. Novak and Kravanja[12] suggest calculating an objective function by a weighted average over the vertices of the feasible regions.[12]

Monte Carlo simulation is one of the most widely used sample average methods. Independent pseudo-random samples are generated to approximate a uniform distribution and then specific values of a probability distribution are created by inverse transformation of the cumulative distribution function. The Hammersley sequence sampling technique is proposed, which may converge faster to the same optimal solution and variance of a distribution compared with the classical Monte Carlo method.[13]

The sample average method can be further categorized into the internal and external sampling methods.[14] In the internal sampling methods, samples may be modified during the optimization procedure. The modifications include the addition of new samples to previously generated ones, removal of subset of previous samples, or generation of completely new samples. The "stochastic decomposition" algorithm for solving two-stage stochastic linear programs[15] and "stochastic branch and bound" (SBB) algorithm[16] are examples of internal sampling algorithms.

The external sampling approaches reduce the stochastic problem to a deterministic one, which can be solved by existing deterministic algorithms. The sample averaging approximation (SSA) is a typical external sampling approach. It can be easily embedded into the framework of many deterministic algorithms. Liu and Sahinidis[17] suggest incorporating the bender-based decomposition method with the SSA method for process planning. Verweij et al.[14] apply the SSA method within a branch-and-cut framework for the solution of stochastic routing problems. For solving the convex problems, Wei and Realff[18] propose a method that combines the SSA method with the outer approximation.[19] Goyal and Ierapetritou[20] suggest the combination of SSA method with the simplicial-based approach.[21] The efficiency of SSA method can be increased by using variant sample sizes. Kleywegt et al.[22] use a smaller sample size to make decisions and a larger sample size to recompute the objective value with the decisions fixed at the values obtained previously from solving the smaller problems. Acevedo and Pistikopoulos[1] suggested the use of numerical integration methods for smaller dimension problems and sampling-based methods for larger problems.

The established algorithms to stochastic MINLP problems are successful in many applications. However, they are built on cardinal optimization, in which the optimums are obtained from the accurate calculation of cardinal values. The cardinal optimization is generally associated with inherent computational difficulties from the combinatorial natural of discrete variables and stochastic natural of uncertain parameters. These approaches may become intractable where the number of discrete variables and/or uncertain parameters becomes very large. Solving the problem to a high degree of accuracy is also counter productive since the problem formulation involves stochastic variables.

The ordinal optimization (OO) is a novel optimization methodology.[23] It is designed to cope with hard problems such as problems with uncertainties, or problems with huge sample space that grows exponentially with respect to the problem size. The OO theory provides a way to obtain solutions of high quality with much less computation effort than the conventional cardinal optimization methods.

The OO theory is based on the following two tenets: (1) it is easier to determine "order" than "value." To determine whether *A* is better or worse than *B* is a simpler task than to determine how much better is *A* than *B* (i.e., the value of *A*-*B*) especially when uncertainties exist. (2) Instead of asking the "best for sure," we seek the "good enough" solution with high probability. This goal softening technique makes the optimization problem much easier. The OO method has found successful applications in a variety of fields, e.g., optimal control, network topology design, planning, and scheduling of manufacturing systems.

In this article, a rank filter algorithm is developed to solve the stochastic MINLP problems. A novel approximation method is proposed, in which the expected performances are estimated by the intermediate NLP optimization solutions and Monte-Carlo simulations with a small sample size. The intermediate results are calculated by a NLP solver with a fixed number of iterations. The theoretical basis for this approximation method is also developed. It is proven that the optimal decision (discrete variable) can be obtained by using the intermediate optimization solutions and rank comparison technique.

The OO theory is then applied to form a subset of good enough decisions with high probability by ranking the crude performance estimations. The suboptimal solution is obtained by searching the subset of selected good decisions and using the accurate performance values. Three benchmark problems are illustrated to show the extra-high-computation efficiency and the quality of the obtained solutions.

The rest of this article is organized as follows. "Stochastic MINLP Problems and the Source of Complexity" presents the formulation of the stochastic MINLP problems. The basic principles of OO are presented in "Ordinal Optimization." "Rank Filter Algorithm to Stochastic MINLP" is the main part of this article. The rank filter algorithm is proposed and its theoretical foundations are developed. Numerical simulation results are shown in "Numerical Examples," and the concluding remarks are given in "Conclusions."

Stochastic MINLP Problems and the Source of Complexity

We consider the following stochastic MINLP problem:

$$\min_{x,y} E_\theta[f(x,y,\theta)] \tag{1}$$

$$s.t. \begin{cases} h(x,y,\theta) = 0 \\ g(x,y,\theta) \le 0 \end{cases} \tag{2}$$

where $x \in \Re^m$ is the vector of continuous variables, $y \in Z^n$ represents the vector of discrete variables, and $\theta \in \Re^k$ is the vector of uncertain parameters. The vector functions h and g denote equality and inequality constraints, respectively. The scalar function f represents the objective function, typically, the expected value of economic performance. The functions $f(x, y_i, \theta_j)$, $g(x, y_i, \theta_j)$ and $h(x, y_i, \theta_j)$ are assumed to be twice differentiable, where y_i and θ_j are special realizations of y and θ, respectively. We call y the decision variable and x the operation variable. The objective of the stochastic MINLP problem is to find the optimal decision and operation variables in presence of uncertain conditions.

Take the problem of process system synthesis as an example. The decision variable y represents the flow-sheet structure, e.g., the existence of a unit operation or not. The operation variables x denote the continuous design, control, and state variables, e.g., the volume of a unit process, the flow rate, temperature, or pressure of a stream. The parameter θ can represent the product price, which is a vector of random variables with known distributions. The set of equations h denotes the process equations, e.g., the mass and energy balances. The inequalities g corresponds to the design speculations and logical constraints.

The complexity comes from two sources: the stochastic part and combinatorial part. The stochastic part deals with the exact evaluation of the expected performance. The closed form solution is usually not available because the analytical solutions to multidimensional integrals are only available in limited cases. Using the sample average approach, the expected performance of decision y_i is therefore estimated by the following:

$$E_\theta[f(x, y_i, \theta)] = \frac{1}{N}\sum_{j=1}^{N} f(x, y_i, \theta_j) \tag{3}$$

where θ_j is the j-th sample of the random variables. According to the central limit theorem, the accuracy of this approximation cannot be improved faster than $1/\sqrt{N}$. This implies that one order of magnitude increase in accuracy requires two order of magnitudes increase in the sample size. Note that each sample corresponds to a NLP problem. It is, therefore, intractable to ask for an expected performance of high accuracy unless the problem is small.[23]

The second source of difficulty comes from the combinatorial nature of the decision variables. It usually leads to a NP hard problem, i.e., the time needed to find a solution increases exponentially with the number of decision variables. Furthermore, the decision space lacks structure. A more or less blind search is the only alternative to the optimization algorithms based on the mathematical analysis, e.g., gradient computation and general hill climbing. According to the no free lunch theory,[24] no random search algorithms can perform better on the average than the blind search. Therefore, it is very hard to improve the computation efficiency of random search like genetic algorithms, simulated annealing algorithms, and other meta-heuristic algorithms.

To make things worse, the effect of these two sources of difficulties is multiplicative instead of additive. This property induces an insurmountable computational burden, if we require solving a stochastic MINLP problem to a high degree of accuracy.

Ordinal Optimization

The OO theory is based on two central ideas: order comparison and goal softening.[23] The order comparison evaluates the quality of the expected performances with respect to their relative order rather than the exact cardinal values. According to Dai,[25] the relative order of the expected performances converges exponentially fast. By comparison, the cardinal values converge at rate of $1/\sqrt{N}$. The relative order, therefore, converges much faster than the cardinal values with respect to the number of samples.

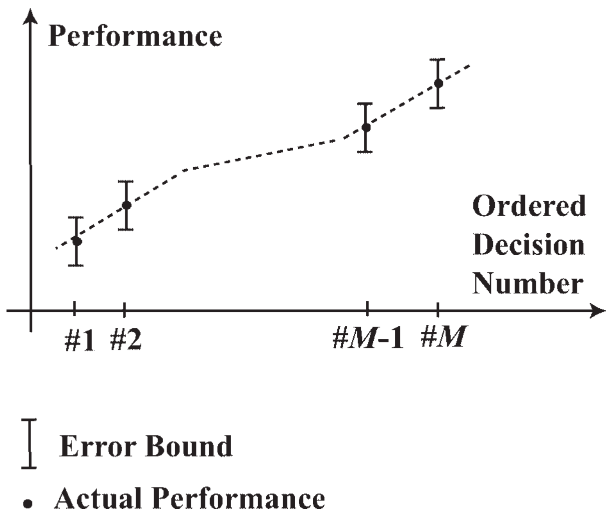

Figure 1. Concept of ordered performance curve (OPC).

The second basis is goal softening. The OO settles for the "good" solutions with high probability. This deviates from the conventional optimization algorithms, which ask for the best solution for sure. By softening our goal to the "good" solutions, e.g., top-g solutions in the search space, the number of good solutions in the set of estimated top-g decisions can be quite substantial even in presence of very large estimation errors in the performance values.

Suppose, we simultaneously evaluate a large set of alternatives very approximately and order them according to the approximate evaluations. High probability can be anticipated that we can find the actual good alternatives by limiting ourselves to the top of the observed good choices.

Consider a search in a decision space Y. Define a "good enough" subset $G \subset Y$ with size $|G| = g$ as the top-g decisions or top-$n\%$ of the decisions in decision space. Define a selected subset $S \subset Y$ with size $|S| = s$ as the set of selected decisions from Y. The alignment probability can then be defined as the probability that at least $k \leq g$ of the member of G is in S, i.e., $P(|G \cap S| \geq k)$. It is proved by Dai[26] that the alignment probability is exponentially increasing to 1 with respect to the size of G and S. It implies that G and S with small sizes are often enough to obtain a high-alignment probability, say $P(|G \cap S| \geq k) > 0.95$.

The goal of OO theory is to construct a selected subset S such that the alignment probability is larger than some given alignment level. Note that $|S| \ll |\Theta|$. A major speedup in search for a good decision can be achieved.

The major task in applying OO theory is to construct the selected subset and determine its size s. The quantitative measure of the "good enough" is the alignment probability, which is related to the ordered performance curves (OPC) of a particular problem. Given M decisions, whose performances are estimated by a crude model, the OPC is obtained by plotting the estimated values from the smallest to the largest against the order indices. An OPC is monotonic by definition and the typical OPCs are shown in Figure 1. The shape of an OPC can show the density and distribution of the top solutions for an optimization problem. Although the exact OPC is generally very difficult to know in a practical problem, its shape may help us estimate the alignment probability.

There are two ways to pick a set of s decisions from a set of crude approximations: Blind Pick (BP) and Horse Race (HR). The BP means that sets of decisions are randomly selected. In the HR method, sets of bid curves are preliminarily evaluated and the best sets of strategies are selected. For the BP method, the size can be determined in closed form.[23] However, the HR method is often preferable since it can make use of results based on a simplified problem or a crude model, and generally ends up with a smaller s when compared with the BP method. The HR rule is used to form the selected set in this article.

In OO, the size of selected set M does not have to be very large to obtain a representative coverage of all possible alternative designs. For example, suppose we take 1000 samples uniformly from any distribution. Denote A as the event that at least five of these samples are in the top 5% of the search space. The probability of the event A can be calculated by the following:

$$P(A) = 1 - \sum_{i=1}^{4} \binom{1000}{i} 0.95^{1000-i} 0.05^{i} \approx 1 - 10^{-17} \approx 1 \tag{4}$$

It follows that $P(A)$ is essentially uniform over all possible realization of the M samples when $M \geq 1000$. This implies that enough "good" enough alternatives are presented in the sampled set, although the sample size is a couple of thousands. Then, one can work with the sampled set of M and treat them as if they were the total population. More precisely, the probability of this rank filter approach is a conditional probability $P(\cdot/A)$. The unconditional probability with respect to the whole decision space is given by the following:

$$P_A = \sum P(\cdot/A)P(A) \tag{5}$$

It is easy to see that the difference between P_A and $P(\cdot/A)$ is minimal for any $M \geq 1000$. According to (4), the calculation of $P(A)$ is independent from $|Y|$. This implies that the computational complexity of the rank filter algorithm is determined by the sample size M rather than the decision space size $|Y|$.

Rank Filter Algorithm to Stochastic MINLP

The rank filter algorithm presents a systematic way of generalizing the classical OO theory to the optimization problems with complicated nonlinear constraints.

Approximation of expected performances

The expected performances are estimated by the average of NLP solutions using the sample average method. The approximation accuracy is determined by the sample size and the quality of NLP solutions. The computational resource is always limited. Hence, it is impossible to choose a very large sample size for each alternative. Given a limited number of samples, we cannot obtain accurate performance expectation even using the optimal NLP solutions. It will lose the computational efficiency to spend computational resources in calculating the optimal solutions. This partly explains why the conventional cardinal optimization methods are computationally expensive.

To relieve the computational burden, two methods are used to approximate the expected performances. The first method is a direct heritage from the classical OO theory. Short Monte Carlo simulation is used with sample size of the order 10^2. This is insufficient from the cardinal optimization point of view. It is often enough for rank comparison.

The second method is to use the NLP solvers with a fixed number of iterations. Instead of running a NLP solver until it converges to the optimum, we ask for an intermediate optimization results that are "good" enough with high probability. The intermediate results are defined as "good" if they contain enough information for rank comparison. Mathematically, this idea is formulated as follows.

Given a stochastic MINLP problem with m decision variables

$$\left\{ \min_{x,y_i} E_\theta[f(x,y_i,\theta)] | h(x,y_i,\theta) = 0, g(x,y_i,\theta) \leq 0, i = 1,\ldots,m \right\} \tag{6}$$

This problem can be reduced to m stochastic NLP problems, i.e.,

$$\left\{ \min_{x} E_\theta[f(x,\theta|y_i)] | h(x,\theta|y_i) = 0, g(x,\theta|y_i) \leq 0 \right\} \tag{7}$$

where y_i, $i \in \{1,\ldots,m\}$ is a fixed decision variable.

In conventional NLP optimizations, the solvers are usually in the form of iterative algorithms, i.e.,

$$x^{k+1} = p(x^k|y_i,\theta_j) \tag{8}$$

The iterative procedure stops when $|x^{k+1} - x^k| \leq \varepsilon$, where ε^* is a predefined accuracy. A variable x^* is called as an optimal solution if the accuracy is high enough, e.g., $\varepsilon = 10^{-6}$. Here, $p(x|y_i, \theta_j)$ denotes a continuous nonlinear function for any decision y_i and any realization of the stochastic parameter θ_j. For convenience, we denote $p_i(x) = p(x|y_i, \theta_j)$.

Assumption 1. The objective function $f(x|y_i, \theta)$ is twice continuously differentiable for any y_i, $i \in \{1,\cdots,m\}$ and uncertain parameter θ.

Assumption 2. The map $p_i(x)$, $i \in \{1,\cdots,m\}$ is a continuously differentiable and the following requirements

1. $p_i(x) \in D$, $\forall x \in D$;
2. $\exists L_i \in (0,1)$ such that $|p_i'(x)| \leq L_i$, $\forall x \in D$

hold with $D \in \Re^m$ is a continuous and compact set.

***Theorem 1**. Given a stochastic MINLP problem stated in (6) subject to Assumption 1 and 2. Denote M_i^* as the number of iterations for calculating the optimal performance $f(x^* \mid y_i, \theta)$. Then there exist m positive integers $M_i \leq M_i^*$, such that*

$$\min_{1\leq i\leq m} \left\{ f\left(x^{M_i}|y_i,\theta\right) \right\} = \min_{1\leq i\leq m} \{ f(x^*|y_i,\theta) \} \tag{9}$$

where x^{M_i} is the intermediate optimal results of $f(x \mid y_i, \theta)$ obtained from M_i iterations.

Proof. Denote $N^* = \sum_{i=1}^{m} M_i^*$ is the total number of iterations for evaluating m decisions. It follows from the Banach fixed point theorem and Assumption 2 that

$$x^* - x^k = \frac{L_i}{1 - L_i}|x^k - x^{k-1}| \tag{10}$$

where the subscript k denotes the number of iterations.

It can be inferred from Assumption 1 that the objective function $f(x \mid y_i, \theta)$ is Lipschitz continuous. Similarly, we denote $f(x \mid y_i, \theta)$ as $f_i(x)$. Let ℓ_i be the Lipschitz constant of $f_i(x)$. The following inequality

$$|f_i(x_1) - f_i(x_2)| \le \ell_i|x_1 - x_2| \tag{11}$$

holds. Substituting (10) to (11), we have the estimate:

$$|f_i(x^*) - f_i(x^k)| \le \frac{\ell_i L_i}{1 - L_i}|x^k - x^{k-1}| \tag{12}$$

Denote $f_I(x^*)$ is the optimum of m feasible decisions, i.e.,

$$\min_{1 \le i \le m}\{f_i(x^*)\} = f_I(x^*) \tag{13}$$

Denote further that

$$f_i(x^*) - f_I(x^*) \ge \varepsilon^* \tag{14}$$

with $i = 1,\ldots,m, i \ne I$. Without losing generality, we can choose any $J \in \{1,\ldots,m\}$, $J \ne I$. It is easy to see that

$$f_J(x^*) - f_I(x^*) = \varepsilon_J^* \ge \varepsilon^* \tag{15}$$

Given any set of two positive numbers $\varepsilon_{I,J}$, ε_J that satisfy

$$\varepsilon_{I,J} + \varepsilon_J \le \varepsilon_J^* \tag{16}$$

There must exist two positive integers $M_{I,J} \le M_I^*$, $M_J \le M_J^*$, such that

$$|x^{M_{I,J}} - x^*| \le \frac{1 - L_I}{\ell_I L_I}\varepsilon_{I,J} \tag{17}$$

$$|x^{M_J} - x^*| \le \frac{1 - L_J}{\ell_J L_J}\varepsilon_J \tag{18}$$

Substituting (12) into (17) and (18), we have the following:

$$\varepsilon_{I,J} \ge |f_I(x^{M_{I,J}}) - f_I(x^*)| \tag{19}$$

$$\varepsilon_J \ge |f_J(x^{M_J}) - f_J(x^*)| \tag{20}$$

It follows that

$$f_I(x^{M_{I,J}}) \le f_I(x^*) + \varepsilon_{I,J} \tag{21}$$

$$f_J(x^{M_J}) \ge f_J(x^*) - \varepsilon_J \tag{22}$$

Subtracting (21) from (22), we can get the estimate

$$f_J(x^{M_J}) - f_I(x^{M_{I,J}}) \ge \varepsilon_j^* - (\varepsilon_J + \varepsilon_{I,J}) \tag{23}$$

Using (16), we finally have

$$f_J(x^{M_J}) - f_I(x^{M_{I,J}}) \ge 0 \tag{24}$$

It follows from (24) that the same decision variable can be obtained by ranking the intermediate results $\{f_I(x^{M_{I,J}}), f_J(x^{M_J})\}$ or the final optimal results $\{f_I(x^*), f_J(x^*)\}$ with $J = 1,\ldots,m$, $J \ne I$. This shows that the qualities of different decisions can be obtained using the intermediate optimums from the truncated iterations of NLP solvers.

Note that (24) is valid for any $J \in \{1,\ldots,m\}$. The result above can be immediately generalized to the cases with many alternative decisions. Denote $M_I = \max\limits_{\substack{I \le J \le m \\ J \ne I}}\{M_{I,J}\}$ and $k = \max\limits_{1 \le p \le m}\{M_p\}$. We can get the following inequalities

$$\varepsilon_I + \varepsilon_J \le \hat{\varepsilon} \tag{25}$$

where

$$\varepsilon_p \ge |f_p(x^k) - f(x^*)| \tag{26}$$

with $p = I,J$ and $1 \le I, J \le m$. It implies that the optimal decision of m alternatives can be obtained by ranking the intermediate optimization results, i.e.,

$$\min_{1 \le p \le m}\{f_p(x^k)\} = f_I(x^k) \tag{27}$$

The total number of iterations for the intermediate optimization is calculated by the following:

$$N^k = \sum_{i=1}^{m}\min\{k, M_i^*\} \tag{28}$$

Recalling that the total number of iterations for the real optimization is $N^* = \sum_{i=1}^{m} M_i^*$. It is evident that $N^k \le N^*$. This completes the proof of Theorem 1.

Theorem 1 shows that the optimal decision can be obtained with less or equal computation cost if the intermediate optimization results are fully utilized in term of the rank-based comparison. Equality holds in the unlikely case that each NLP in the rank filter algorithm executes exactly the same number of time as in the cardinal optimization algorithms. Here, $\hat{\varepsilon}$ is the predefined error bound, which define the quality of NLP optimization results. The bound can be chosen by the users, who can then effectively trade off between computation time and accuracy. Using the order comparison and goal softening, the error bound can be significantly increased without affecting the final optimum significantly, because the order is very robust to approximation errors.[23] It should be noted that each sampling realization corresponds to an NLP programming problem. The use of intermediate optimizations can lead to a significant reduction in computation burden when the time saving of a single NLP optimization is amplified by the sample size.

Theorem 1 provides an existence proof. It does not hint any closed-form solution on the required number of iterations to guarantee (27). In real applications, these numbers are obtained from the deterministic MINLP problems. By substituting the stochastic parameters with their expectations, we can calculate the required number of iterations to satisfy (27). This number is used in the stochastic optimization problems. Another way to determine the number of iterations is to choose a small value, typically between 5 and 20. It is shown via extensive numerical experimentation that the suboptimal solutions are often of high quality for rank comparison.

Because of the uncertainty, we cannot get a clear differentiation of the "good" from the "bad" for sure even from the order comparison point of view. The separation algorithm needs to work under noises, i.e., the approximation errors. In this proposed algorithm, the design of a filter is the formation of a selected subset S. On one hand, the set of S filters out the "good" decisions from the others. On the other hand, the size of S is adjusted to the magnitude of noises. If the expectations contain no noise, the top-1 decision in rank comparison is in fact the real optimal solution. To circumvent the effect of noise, we choose a set of decisions instead of a single one. It gives the proposed algorithm good noise rejection characteristics. This is similar with the working mechanism of classical filters. To differentiate this algorithm from the classical OO theory, we name it as the rank filter algorithm.

Selection of good enough decisions

Given a representative set Π with M decisions, whose performances are estimated by using the approximation algorithm stated in previous section. Denote the true and crude expected performances as J_i and $\hat{J}_i$, respectively. We can get the following:

$$J_i = \hat{J}_i + e_i \tag{29}$$

where e_i is the approximation error with $i = 1,\ldots,M$. There may be significant errors due to the rough approximations. The advantage of an OO-based method is its capability to separate the good from the bad even using very crude approximation. The performance order is relatively immune to large approximation errors. Therefore, even if the rough estimation is used to rank M selections' performances, some good enough decisions will be kept within the select set with high probability.

The major task in applying OO theory is to construct a selected subset S containing "good enough" decisions with high probability. The quantitative measure of the "good enough" is the alignment probability defined as follows:

$$P(G, S, a) = P_A(|G \cap S| \geq a) \tag{30}$$

where G is the "good enough" set and a is called the alignment level. Intuitively, the alignment probability is the probability of the event that there are at least a elements in the intersection set of G and S.

To select s good ones from M representatives, the crude approximated performances are calculated and ranked. The top s decisions are then selected as S and its size s is determined by a regressed nonlinear equation to satisfy certain confidence requirement.[27] The value of s can be estimated by the following:

$$s \approx Z(a, g) = e^{Z_0} a^{\rho} g^{\gamma} + \eta \tag{31}$$

where $g = |G|$ is the size of G, and Z_0, ρ, γ, η are coefficients or parameters obtained by nonlinear regression. These coefficients depend on the alignment probability P_A, the value of M and the shape of the ordered performance curve (OPC). In an OPC, the evaluate profits of M decisions are ordered and plotted. The alignment probability for typical coefficients, e.g., $P_A = 0.95$ and $M = 1000$ are published through intensive simulation for different OPCs.[27]

The evaluation of M decisions using crude approximations is computationally efficient and the OO theory guarantees that good decisions will be among the selected set in high probability. More accurate but time-consuming evaluation is applied to evaluate the selected decisions. For each decision, the accurate performance is calculated using a long Monte-Carlo simulation and the real optimums of NLP problems. The best decision is then selected by evaluating those decisions in the selected subset based on the accurate performances. Since $s = |S|$ is much smaller than M, the OO method is extremely efficient in comparison with brute and force method by calculating M accurate performances.

Rank-filter algorithm

The main steps of the rank filter algorithm are summarized as follows:

1) Choose M decisions from the decision space randomly;
2) Run short simulations with a small sample size of N_1 for each decision;
3) Run the NLP solver with k iterations for each sample realization;
4) Calculate the approximated expectations of M decisions using the average of N_1 intermediate optimization solutions;
5) Rank the approximated expectations and choose the top-s ones, where s is calculated from (26);
6) Calculate the accurate performances of the selected s decisions using a big sample size of N_2 and optimal NLP solutions;
7) Get the suboptimal solution by ranking the accurate performances of s decisions.

The parameters in this algorithm can be obtained from the corresponding deterministic MINLP problem, where the random parameters are substituted by their mathematical expectations. A typical set of parameters are shown in Table 1. In this table, the first four parameters are taken from the classical OO configurations, which fit for most of the real applications. For the last three parameters, we only give the order of magnitude of the values, which can be adjusted to different problems.

The set of "good enough" solutions of the ordinal optimization can be different from that of the cardinal optimization. For example, assume that the design space consists of

Table 1. Typical Parameters of the Rank Filter Algorithm

Parameter	Typical value
M	1000
P_A	0.95
g	50
a	5
N_1	$O(10^2)$
N_2	$O(10^4)$
k	$O(10)$

1 million possible designs, i.e., $|Y| = 10^6$. Define the "good enough" set as the top 0.1% designs in Y. This implies that the design no. 1 is the best design, no. 2 the second best, and the design no. 5001- #10^6 is the lower 99.9% designs in Y. To search within the top 0.1% of Y does not necessarily mean that we are within 0.1% of the best performance value. One can always construct a problem in which the performances of the top designs change sharply, e.g., [10^6,10^5, 10^4,1,...,1]. This kind of problem is inherently hard because they lack the structure needed to find the global optimums.[23]

In the rank filter algorithm, the size of selected set M is determined by the softened goal. It is independent from the size of design space $|Y|$. Furthermore, the softened goal is ultimately specified by the available computational resource. More resources imply less softening. In the rank filter algorithm, M is scaled to the user's resources instead of $|Y|$. However, if the design space is very huge, e.g., $|Y| = 2^{100}$, it does not mean very much to narrow down the designs to 1% or even 0.01%. In this case, it is not sufficient to restrict the sampling to a few thousands designs. The concept of iterative OO (IOO) needs to be introduced, which is in the spirit of hill climbing in traditional cardinal optimization. The IOO iterates on a set of successive "good enough" solutions instead of successive points in conventional optimization.[23] More research needs to be done to verify the performance and efficiency of the IOO-based algorithm in stochastic MINLP problems.

Numerical Examples

The first benchmark clarifies the procedure of the rank filter algorithm. The second demonstrates the robustness of the proposed algorithm to the number of random parameters. The last example is challenging because it is formulated as a nonconvex MINLP problem with a huge-decision space. This example shows the potential of using the rank filter algorithm in real industrial problems.

Example 1. The first benchmark is the process synthesis/planning problem proposed by Ierapetritou, Acevedo and Pistikopoulos (1996) and also studied by Goyal and Ierapetritou (2007). Figure 2 shows the process superstructure for the production of C from the raw material A and intermediate product B_f. In this superstructure, process 1 and 2 are fixed operation units, whereas process 3 and 4 are optional ones. There is also an option of purchasing the intermediate product B_f to complement the production requirements.

The objective function is given by the maximization of an expected profit. This expectation is calculated from the revenue obtained from the product C, the cost of raw material A, B_f, the fixed and operating costs. The constraints are given by the material balances, the yield relations, the design, and logical restrictions. The mathematical model is formulated as follows:

$$\begin{aligned}
\max_{x,y}\quad & E_\theta\{555x_{10} - (250x_1 + 300x_9) - [100 + 15(x_5 + x_9) \\
& +(80 + 5x_2) + (130y_3 + 15x_3) + (150y_4 + 5x_4)]\} \\
s.t.\quad & x_1 = x_2 + x_3 + x_4 \\
& x_5 = x_6 + x_7 + x_8 \\
& x_{10} = 0.9(x_5 + x_9) \\
& x_6 = 18\ln(1 + \tfrac{x_2}{20}) \\
& x_7 = 20\ln(1 + \tfrac{x_3}{21}) \\
& x_8 = 15\ln(1 + \tfrac{x_4}{26}) \\
& x_1 \le \theta_2, \quad x_2 \ge 5 \\
& x_3 \le 20y_3, \quad x_4 \le 20y_4 \\
& x_9 \le 15, \quad x_{10} \le \theta_1 \\
& x \in \Re^{10}, \quad y \in \{0,1\}^2 \\
& \theta_1 \in N(20,2) \\
& \theta_2 \in N(15,1)
\end{aligned}$$

where $x = [A_1,A_2,A_3,A_4,B_1,B_2,B_3,B_4,B_f,C]^T$ and $y = [y_3,y_4]^T$ represents the existence of process 3 and 4, respectively. The θ_1,θ_2 are random parameters with normal distributions.

The rank filter algorithm is used to solve this problem. All samples are generated by a truncated range of $\mu \pm 3\delta$, where μ is the expectation and δ the standard deviation. This implies that $14 \le \theta_1 \le 26$ and $12 \le \theta_2 \le 18$. The decision space consists of four alternative decisions. The coarse expected performances are obtained by generating 100 samples for each decision. The corresponding NLP problems are solved by a Matlab NLP solver with five iterations. Accordingly, the coarse expectation is calculated from the average of 100 samples' performances, and each performance is essentially an intermediate optimization result. By ranking the four coarse approximations, we choose the top-2 decisions to form the selected set. To get the accurate expectations, the sample size of 2000 and rigorous NLP optimization are used. The optimums are calculated by running the NLP solver until it converges to a precise error criterion, e.g., $\varepsilon \le 10^{-6}$. The coarse and accurate estimate are shown in Table 2.

By comparing the accurate performances of two selected decisions, the optimal decision $y = [0,1]^T$ is obtained with an expected cost of 3441.3. Table 3 lists the optimization

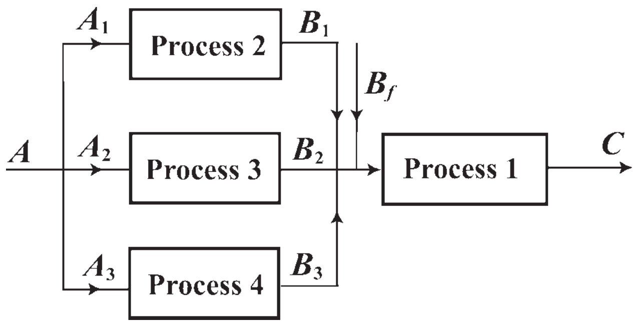

Figure 2. Process superstructure for Example 1 and 2.

Table 2. Expected Performances of Decisions

No.	y	Coarse expectation	Accurate expectation
1	[0 0]	2674.1	2712.9
2	[0 1]	3019.5	2968.8
3	[1 0]	3418.4	3441.1
4	[1 1]	3262.3	3293.3

results using the rank filter algorithm, simple approximation algorithm, and GAMS/SBB.[20] It is evident that the optimal result of this rank filter algorithm shares the same optimal decision with the other two methods. The small discrepancy in the optimal expectations is caused by random samplings. In addition, the computation time of the rank filter algorithm is 70.5 CPUs. This implies a high computational efficiency, although the Matlab NLP solver is generally several times slower than the commercial solvers, e.g., GAMS.

Table 2 also lists the accurate expectations of all decisions. By comparing these performance expectations, we can see that the relative order of the coarse approximations is exactly the same with that of the accurate ones. It proves that the rank comparison is not sensitive to the approximation error. This feature contributes to the extra high computational efficiency of the rank filter approach.

Example 2. The second example is the process flow-sheet synthesis problem studied by Duran and Grossmann (1986) and Goyal and Ierapetritou (2004). This benchmark shares the same superstructure shown in Figure 2. In this example, only process 1 is fixed. Hence, three binary variables are needed to represent the existence of the process 2, 3 and 4.

This example involves seven random parameters. The normal distributions are assumed for four continuous random variables, i.e., the availability of raw material A, the demand of product C, the prices of material A, and pure $B(B_f)$. Table 4 lists the distribution data of three discrete random parameters, i.e., the kinetic constants of process 2, 3, and 4.

$$\begin{aligned}
\max_{x,y}\quad & E_\theta\{555x_{10} - (\theta_3 x_1 + \theta_4 x_9) - [100 + 15(x_5 + x_9) \\
& +(80 + 5x_2) + (130y_3 + 15x_3) + (150y_4 + 5x_4)]\} \\
s.t.\quad & x_1 = x_2 + x_3 + x_4 \\
& x_5 = x_6 + x_7 + x_8 \\
& x_{10} = 0.9(x_5 + x_9) \\
& x_6 = 18\ln(1 + \tfrac{x_2}{\theta_5}) \\
& x_7 = 20\ln(1 + \tfrac{x_3}{\theta_6}) \\
& x_8 = 15\ln(1 + \tfrac{x_4}{\theta_7}) \\
& x_1 \le \theta_2, \quad 5 \le x_2 \le 25y_2 \\
& x_3 \le 20y_3, \quad x_4 \le 20y_4 \\
& x_9 \le 15, \quad x_{10} \le \theta_1 \\
& x \in \Re^{10}, \quad y \in \{0,1\}^3 \\
& \theta_1 \in N(17.5, 2.5) \\
& \theta_2 \in N(27.5, 2.5) \\
& \theta_3 \in N(250, 10) \\
& \theta_4 \in N(300, 15)
\end{aligned}$$

with $x = [A_1,A_2,A_3,A_4,B_1,B_2,B_3,B_4,B_f,C]^T$ and $y = [y_2,y_3,y_4]^T$.

As in Example 1, the realizations of continuous random parameters are generated using a truncated range of $\mu \pm 3\delta$. To calculate the coarse expected performances, a sample size of 200 is used. The intermediate optimization results are calculated by running a Matlab NLP solver with 15 iterations. The top-2 designs are chosen to form the selected set S. The precise expected performances of the selected designs are calculated from 5000 optimums of NLP problems.

Table 5 lists the calculation results of the top-2 decisions. By comparing the accurate optimums, we can get the optimal decision vector $y^* = [1,1,1]^T$ with an expected performance of 4068.6. The same optimal decision is obtained by the decomposition-based algorithm in Goyal and Ierapetritou.[10]

The overall calculation time is 282.7 CPUs. It is of high-computation efficiency by considering the big sample size of 5000 for calculating the accurate expectations. In this example, the total number of NLPs is 40,000, which is five times of that in Example 1. The calculation time of Example 2 is four times of that of Example 1. Hence, the proposed algorithm solves Example 2 faster than Example 1 on the average, although the former consists of more random parameters. This shows that the rank filter algorithm is robust to the number of random parameters.

Example 3. The pump network synthesis problem is studied by Goyal and Ierapetritou,[21] Adjiman et al.,[28] and Westerlund et al.[29] This problem belongs to the class of nonconvex MINLPs. The aim is to identify the least costly configuration of centrifugal pumps that achieves a prespecified pressure rise based on a given total flow rate. For a three-level superstructure, the minimization of the annualized cost is given by the following:

$$\begin{aligned}
& \min_{x,y} \sum_{i=1}^{3} (C_i + \overline{C}_i P_i) Np_i Ns_i z_i \\
s.t.\quad & \sum_{i=1}^{3} x_i = 1 \\
& P_i - \alpha_i\left(\frac{\omega_i}{\omega_{\max}}\right)^3 - \beta_i\left(\frac{\omega_i}{\omega_{\max}}\right)^2 \dot{v}_i - \gamma_i\left(\frac{\omega_i}{\omega_{\max}}\right)\dot{v}_i^2 = 0 \\
& \Delta P_i - a_i\left(\frac{\omega_i}{\omega_{\max}}\right)^2 - b_i\left(\frac{\omega_i}{\omega_{\max}}\right)\dot{v}_i - c_i\dot{v}_i^2 = 0 \\
& \dot{v}_i Np_i - x_i V_{\text{tot}} = 0 \\
& \Delta P_{\text{tot}} z_i - \Delta p_i Ns_i = 0 \\
& 0 \le x_i \le 1, \quad 0 \le \dot{v}_i \le V_{\text{tot}} \\
& 0 \le \omega_i \le \omega_{\max}, \quad 0 \le P_i \le P_i^{\max} \\
& 0 \le \Delta p_i \le \Delta P_{\text{tot}}, \quad z_j \in \{0,1\} \\
& Np_i, Ns_i \in \{0,1,2,3\} \\
& V_{\text{tot}} \in N(350, 40) \\
& \overline{C}_i \in N(1800, 200)
\end{aligned}$$

Table 3. Comparison of Different Algorithms

Solver	y^*	Performance	No of samples
Rank-Filter-Based Algorithm	[1 0]	3441.1	100
Simplicial Approximation	[1 0]	3340.8	2000
GAMS/SBB Algorithm	[1 0]	3340.8	2000

Table 4. Uncertain Kinetic Constants

	1	2	3	4	5	6	7	8
θ_5	19	19	20	20	20	21	21	21
θ_6	20	21	21	22	22	22	21	20
θ_7	25	26	25	26	27	27	26	26
Probability	0.1	0.15	0.15	0.15	0.1	0.1	0.15	0.1

where $x = [x_i, w_i, \dot{v}_i, P_i, \Delta P_i]^T$ and $y = [z_i, Np_i, Ns_i]^T$ with $i =$, 1,2,3. The parameter values are shown in Table 6. This example has two random parameters with normal distributions, i.e., the total volume V_{tot} and the operation cost $\overline{C}_i$.

The problem is nonconvex in the continuous and integer variables. Thirty-seven local minima have been reported in Pettersson and coworkers[29] for the deterministic problem. Another source of difficulty comes from the huge space of decision variables. There are more than 32,000 combinations if we consider the range of values of the integer variables.

For each decision, the coarse expected performance is calculated using a sample size of 100 and a NLP solver with 30 iterations. More iterations as to 30 are used here because the corresponding deterministic NLP problems are nonconvex. This number is obtained from the simulation of the deterministic problem.

To obtain the size of the selected set S, the knowledge on the shape of OPC is needed. The rough expectations of different decisions are ordered, normalized, and plotted as shown in Figure 3. It is shown that the OPC is of bell shape. According to Lau and Ho (1997), the top-39 designs are chosen to form the selected set. The rigorous expected performances of the selected decisions are calculated from 5000 NLP optimization results. By ranking the selected decisions with their accurate expected performances, the optimal decision $y = [1,1,0,2,1,1,2,0,0]^T$ is obtained with an expected performance of 128011.1. Table 7 lists the optimization results of the top-5 optimal results. The calculation time is 2.50 h, which is at the same order of magnitude of the computation time of the deterministic algorithms.[29] This shows that the proposed algorithm has high-computation efficiency.

Note that the deviations of random parameters are very small. There should be no major difference in the optimums between the stochastic and deterministic MINLP problems. The deterministic pump network synthesis problem gives the same optimal decision[29] with an optimal performance value of 128893.7. This shows that the optimal result of the rank filter algorithm is of high quality.

To test the impact of the uncertainties, we introduce large standard deviations into the random parameters, i.e., $V_{\text{tot}} \in N(350,120)$, $\overline{C}_i \in N(1800,600)$. Using the same algorithm stated above, the optimal decision $y = [1,0,0,2,1,0,0,0,0]^T$ is obtained with an expected performance of 96592.3. Meanwhile, we enumerate all the feasible solutions, whose performances are calculated from the rigorous NLP optimization and a sample size of 2000 for each design. The optimal decision is $y = [1,0,1,1,1,0,0,1,2]^T$ whose expected performance is 94131.0. The rank filter algorithm converges to a suboptimal solution. The relative error is calculated as follows:

$$\varepsilon = \frac{96592.3 - 94131.0}{94131.0} \times 100\% = 2.6\% \tag{32}$$

It follows that the suboptimal solution is within top 3% of the best performance value. It is shown that the rank filter algorithm can produce a suboptimal solution of high quality even solving the problems with large uncertainties.

In fact, the proposed algorithm often leads to a conservative selected set, i.e., the number of selected decisions is larger than required. In this case, the global optimal can be obtained even by choosing the top 20 designs. The conservation of OO theory provides an extra safety margin at the cost of higher computation expense.

Table 5. Expected Performances of Selected Decisions

No.	y	Approximate optimal	Real optimal
1	[1 1 1]	3848.3	4068.6
2	[1 1 0]	3873.2	4057.6

Table 6. Data for Pump Network Synthesis Problem

	Pump 1	Pump 2	Pump 3
C_i	6329.03	2489.31	3270.27
α_i	19.9	1.21	6.52
β_i	0.161	0.0644	0.102
γ_i	−0.000561	−0.000564	−0.000232
a_i	629.0	215.0	361.0
b_i	0.696	2.95	0.530
c_i	−0.0116	−0.115	−0.00946
$P_i^{\max} = 400$	80	25	35
	$\Delta P_{tot} = 400$	$\omega_{\max} = 2950$	

Conclusions

In this article, we propose a rank filter algorithm to solve stochastic MINLP problems. This algorithm is based on a novel approximation method and the OO-theory framework.

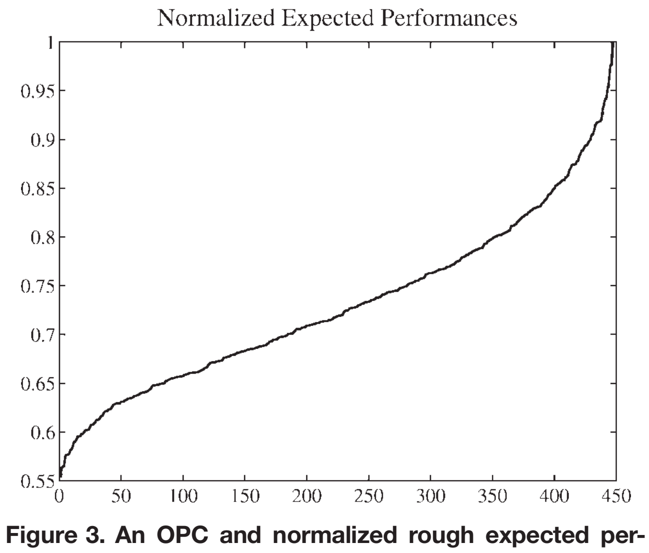

Figure 3. An OPC and normalized rough expected performances of the deterministic MINLP problem.

Table 7. Expected Performances of Selected Decisions

No.	y	Approximate optimal	Real optimal
1	[1 1 0 2 1 1 2 0 0]	127905.6	128011.1
2	[1 1 1 1 1 1 2 1 2]	131296.0	131616.8
3	[1 0 1 2 1 0 0 1 2]	132361.2	131628.3
4	[1 0 0 3 1 0 0 0 0]	134051.5	134342.0
5	[0 0 1 0 0 0 0 3 2]	134668.0	134916.0

It guarantees a suboptimal solution of high quality with high probability. The proposed algorithm can ease several order of magnitude of computation expense without significant loss of solution precision compared with the conventional algorithms based on cardinal optimization. The rank filter algorithm is promising to find successful applications in a variety fields, such as the synthesis of large-scale chemical process, optimization of supply chain, and control of hybrid systems.

Acknowledgments

The authors like to thank the reviewers and Associate Editor for their insightful and valuable comments and suggestions.

Literature Cited

1. Acevedo J, Pistikopoulos EN. Stochastic optimization based algorithms for process synthesis under uncertainty. *Comput Chem Eng*. 1998;22:647–671.
2. Chaudhuri PD, Diwekar UM. Process synthesis under uncertainty: a penalty function approach. *AIChE J*. 1996;42:742–752.
3. Ierapetritou MG, Acevedo J, Pistikopoulos EN. An optimization approach for process engineering problems under uncertainty. *Comput Chem Eng*. 1996;20:703–709.
4. Balasubramanian J, Grossmann IE. A novel branch and bound algorithm for scheduling flowshop plants with uncertain processing times. *Comput Chem Eng*. 2002;26:41–57.
5. Petkov SB, Maranas CD. Design of single-product campaign batch plants under demand uncertainty. *AIChE J*. 1998;44:896–911.
6. Li Z, Ierapetritou MG. Process scheduling under uncertainty using multiparametric programming. *AIChE J*. 2007;53:3183–3203.
7. Mohideen MJ, Perkins JD, Pistikopoulos EN. Optimal design of dynamic systems under uncertainty. *AIChE J*. 1996;42:2251–2272.
8. Ostrovsky GM, Datskov IV, Achenie LEK, Volin YM. Process uncertainty: case of insufficient process data at the operation stage. *AIChE J*. 2003;49:1216–1232.
9. Chaudhuri PD, Diwekar UM. Synthesis approach to the determination of optimal waste blends under uncertainty. *AIChE J*. 1999;45:1671–1687.
10. Goel V, Grossmann IE. A stochastic programming approach to planning of offshore gas field developments under uncertainty in reserves. *Comput Chem Eng*. 2004;28:1409–1429.
11. Clay RL, Grossmann IE. A disaggregation algorithm for the optimization of stochastic planning models. *Comput Chem Eng*. 1997;21:751–774.
12. Novak Z, Kravanja Z. Mixed-integer nonlinear programming problem process synthesis under uncertainty by reduced dimensional stochastic optimization. *Ind Eng Chem Res*. 1999;38:2680–2698.
13. Kalagnanam JR, Diwekar UM. An efficient sampling technique for offline quality control. *Technometrics* 1997;39:308.
14. Verweij B, Ahmed S, Kleywegt AJ, Nemhauser G, Shapiro A. The sample average approximation method applied to stochastic routing problems: a computational study. *Comput Optim Appl*. 2003;24:289.
15. Higle JL, Sen S. Stochastic decomposition: an algorithm for two-stage linear programs with recourse. *Math Oper Res*. 1991;16:650–669.
16. Norkin WI, Pflug GCh, Ruszczyñski A. A branch and bound method for stochastic global optimization. *Math Program.* 1998;83:425–450.
17. Liu ML, Sahinidis NV. Optimization in process planning under uncertainty. *Ind Eng Chem Res*. 1996;35:4154–4165.
18. Wei J, Realff MJ. Sample average approximation methods for stochastic MINLPs. *Comput Chem Eng*. 2004;28:333–346.
19. Duran MA, Grossmann IE. An outer-approximation algorithm for a class of mixed integer nonlinear programs. *Math Program.* 1986;36:307.
20. Goyal V, Ierapetritou MG. Stochastic MINLP optimization using simplicial approximation. *Comput Chem Eng*. 2007;31:1081–1087.
21. Goyal V, Ierapetritou MG. MINLP optimization using simplicial approximation method for classes of nonconvex problems. In: Floudas CA, Pardalos PM, editors. *Frontiers in Global Optimization*. Kluwer Academic Publishers, Boston. 2004.
22. Kleywegt AJ, Shapiro A, Homem-De-Mello T. The sample average approximation method for stochastic discrete optimization. *SIAM J Optim*. 2001;12:479.
23. Ho YC. An explanation of ordinal optimization: soft computing for hard problems. *Inform Sci*. 1999;113:169–192.
24. Wolpert DH, Mcreadym WG. No free lunch theorems for optimization. *IEEE Trans Evolut Comput*. 1997;1:67–82.
25. Dai LY. Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems. *J Optim Theor Appl*. 1996;91:363–388.
26. Xie XL. Dynamics and convergence rate of ordinal comparison of stochastic discrete-event systems. *IEEE Trans Automat Contr*. 1997;42:586–590.
27. Lau TW, Ho YC. Universal alignment probabilities and subset selection for ordinal optimization. *J Optim Theor Appl*. 1997;93:455–489.
28. Adjiman CS, Androulakis IP, Floudas CA. Global optimization of mixed integer nonlinear problems. *AIChE J*. 2000;46:1769–1797.
29. Westerlund T, Pettersson F, Grossmann IE. Optimization of pump configurations as a MINLP problem. *Comput Chem Eng*. 1994;18:845–858.

Manuscript received Jun. 22, 2008, and revision received Mar. 2, 2009.