Efficient Combinatorial Optimization under Uncertainty. 1. Algorithmic Development

Ki-Joo Kim and Urmila M. Diwekar*

CUSTOM (Center for Uncertain Systems: Tools for Optimization and Management) and Civil and Environmental Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, Pennsylvania 15213

This paper presents hierarchical improvements to the combinatorial stochastic annealing algorithm using a new and efficient sampling technique. The Hammersley sequence sampling technique is used for updating discrete combinations, reducing the Markov chain length, determining the number of samples automatically, and embedding better confidence intervals of the samples. The improved algorithm, Hammersley stochastic annealing, can significantly improve computational efficiency over traditional stochastic programming methods. Three distinctive example functions considered proved the efficiency improvement of Hammersley stochastic annealing to be up to 99.3% better than the traditional counterparts. Thus, this new method can be a useful tool for large-scale combinatorial stochastic programming problems. Application of this new algorithm to a real world problem of solvent selection under uncertainty is presented in part 2 of this series.

1. Introduction

Optimization under uncertainty refers to that branch of optimization problems where there are uncertainties involved in the data or model, popularly known as stochastic programming problems. Stochastic programming gives the ability to optimize in the face of uncertainties and requires that the objective function and constraints be expressed in terms of some probabilistic representation (e.g., expected value, variance, fractiles, or most likely values). The general way to treat the probabilistic functional of the objective function and constraints is to use stochastic models instead of deterministic models in the problem formulation. Thus, the stochastic optimization problem, where there are decision variables and uncertain parameters, can be viewed as

$$(\mathbb{P}1)\qquad \min z = P_1[f(\mathbf{x},\xi)] \qquad (1)$$

$$\text{s.t.}\quad P_2[h(\mathbf{x},\xi)] = 0$$

$$P_3[g(\mathbf{x},\xi)] \leq 0$$

$$\mathbf{x} \in X,\ \xi \in \Xi$$

where $\mathbf{x}$ is a vector of decision variables and ξ is a vector of uncertain parameters of the domain Ξ. The performance metric to be optimized is represented by a probabilistic function P_1, and the model equality and inequality constraints are defined by a set of probability functions P_2 and P_3, respectively. The probability function P_i represents the cumulative distribution functional such as the expected value, mode, variance, or fractiles. For example, if P_i is the expected value, the above optimization problem becomes

$$(\mathbb{P}2)\qquad \min z = E_\xi f(\mathbf{x},\xi) \qquad (2)$$

where E_ξ is the mathematical expectation with respect to ξ. The optimal solution and optimal value of problem $\mathbb{P}2$ are x^* and z^*, respectively. The main difficulty of stochastic programming stems from evaluating the uncertain functions and their expectations. A generalized method to propagate the uncertainties is to use a sampling method. A common method is to propagate N_{samp} samples generated from the random values of ξ and optimize the following approximated problem:

$$(\mathbb{P}3)\qquad \min z = \frac{1}{N_{\text{samp}}}\sum_{j=1}^{N_{\text{samp}}} f(\mathbf{x},\xi^j) \qquad (3)$$

Similarly, the optimal solution and optimal value of this approximation problem are $\hat{x}$ and $\hat{z}$, respectively. To solve this approximated stochastic problem, the optimization and sampling techniques for ξ are performed simultaneously. Thus, the generalized stochastic framework for solving optimization under uncertainty problems involve two recursive loops: (1) the inner sampling loop and (2) the outer optimization loop. For example, sampling is embedded into the L-shaped method, which is an approximation of the nonlinear term in the objective function of stochastic programming problems.[1,2]

In the sampling loop, Monte Carlo sampling (MCS), a *pseudo*-random number generator, has been commonly used for representing ξ. Even though MCS generates independent and random samples of a given probability distribution, it is known that MCS requires a large number of samples, N_{samp}, to approximate the "true" mean or variance. Decreasing the distance between the true and approximated optimal solutions $|x^* - \hat{x}|$, therefore, increases N_{samp} and results in a high computational intensity. To reduce the computational burden of stochastic programming problems, one can use either decomposition methods with sampling[1−3] or efficient sampling methods. However, most of the decomposition

* To whom correspondence should be addressed. E-mail: urmila@cmu.edu. Phone: +1 412 268 3003. Fax: +1 412 268 7813.

methods require convexity conditions and dual-angular structures and are only applicable to problems involving continuous decisions. The area of discrete optimization problems, the focus of this paper, contains computationally intensive stochastic programming problems.

Recently, a new *quasi*-random sampling technique referred to as Hammersley sequence sampling (HSS)[4,5] was proposed, and it has been shown to exhibit better *homogeneity* over the multivariate parameter space. Further, for this new sampling technique, it has been found that the number of samples required to converge to the different performance measures (such as mean, variance, or fractiles) of an output random variable, subject to input uncertainties, is lower than the crude MCS or the variance reduction techniques such as Latin hypercube sampling (LHS).[6] This rapid convergence property of HSS has important implications for stochastic programming, suggesting that precise estimates of any probabilistic function evaluations are achievable by taking a smaller sample size. These uniformity and faster convergence properties of HSS can be used for the outer optimization loop as well as the inner loop to achieve better computational efficiency, with the stochastic programming problems involving discrete decision and continuous decision variables.

Discrete optimization problems involving deterministic models can be classified as IP (integer programming), MILP (mixed-integer linear programming), and MINLP (mixed-integer nonlinear programming) problems. They involve discrete decision variables that have many real world applications: molecular design, DNA alignment, vehicle routing, staff scheduling, production planning, inventory management, facility location, very large-scale integration design, image processing, data network design, etc. For representing discrete variables, the binary variable representation has been used in traditional mathematical programming algorithms. The approaches for solving discrete optimization methods such as branch and bound, cutting plane, generalized Bender's decomposition, and outer approximation[7−9] involve bounding heuristics or integer relaxation at the discrete optimization level and use the traditional nonlinear programming or linear programming algorithms for solving the continuous optimization problem. Probabilistic methods such as simulated annealing (SA),[10] on the other hand, use physical analogy in a discrete space to progress toward optimum. These methods can also be used to solve MINLPs but tend to be less efficient than their mathematical programming counterparts. However, coupling SA with mathematical programming algorithms for continuous optimization, such as the traditional nonlinear programming algorithms, can provide a viable alternative to the mathematical programming methods. Further, like the mathematical programming approach, this coupled approach does not encounter difficulties when functions do not satisfy convexity conditions, when systems have large combinatorial explosion, or when the solution space has discontinuity. In recent years, a new variant of SA called stochastic annealing[11−13] was designed to efficiently optimize a probabilistic objective function by automatically selecting the number of samples needed to approximate the uncertain surface. This paper focuses on improving the efficiency of the SA-based algorithms. The probabilistic nature of these SA-based algorithms is the basis for the hierarchical improvement to be presented in this paper.

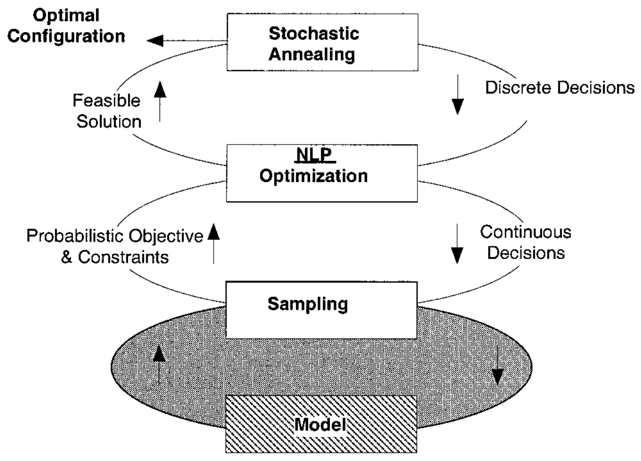

Figure 1. Coupled framework of STA−NLP for MINLP problems.

Table 1. Three-Level Efficiency Improvement in Stochastic Optimization

level	algorithm	modification target
i	ESA	optimization loop
ii	ESTA	sampling loop
iii	HSTA	confidence interval

Hierarchical improvements of stochastic annealing for reducing computational intensity are achieved by using a new efficient sampling technique, HSS, both in the inner sampling and in the outer optimization loops. New improved SA-based stochastic programming algorithms are summarized in Table 1. In SA and its variants, each configuration that is a set of discrete decision variables is randomly generated from the previous configuration, and this update is generally based on a random probability given by a uniform distribution of the neighboring configurations. Because it is known that this probability can significantly affect the overall efficiency of the annealing process,[14] efficient SA (ESA) aims to improve this generation mechanism by using the *uniformity* property of HSS. The efficient stochastic annealing (ESTA) algorithm not only improves the generation mechanism but also reduces the number of samples required in the inner sampling loop by using the *faster convergence* property of HSS. Finally Hammersley stochastic annealing (HSTA) applies an accurate confidence interval of the samples to the ESTA algorithm. Similar to SA−NLP, a coupled framework of STA−NLP designed to solve MINLP under uncertainty is shown in Figure 1, where the upper stochastic annealing block is modified to HSTA. The proposed combinatorial optimization algorithm under uncertainty can also be used for dependent uncertain variables. In this case, the sampling loop in Figure 1 can generate correlated samples for the uncertain variables, and the STA−NLP algorithm can do the same thing.

This paper is composed as follows: Section 2 describes the generalized stochastic models involved in stochastic programming problems, section 3 explains and demonstrates hierarchical improvements in the SA-based algorithms, and the last section concludes the paper. In part 2 of this series, HSTA is applied to a solvent selection problem under uncertainty, whose aim is to find the best promising solvents with the properties desired. All possible solvent molecules are generated by constructing their building blocks or groups, and their chemical and physical properties are estimated. HSTA is used to select groups to form molecules, impose uncertainties on property estimation methods, and

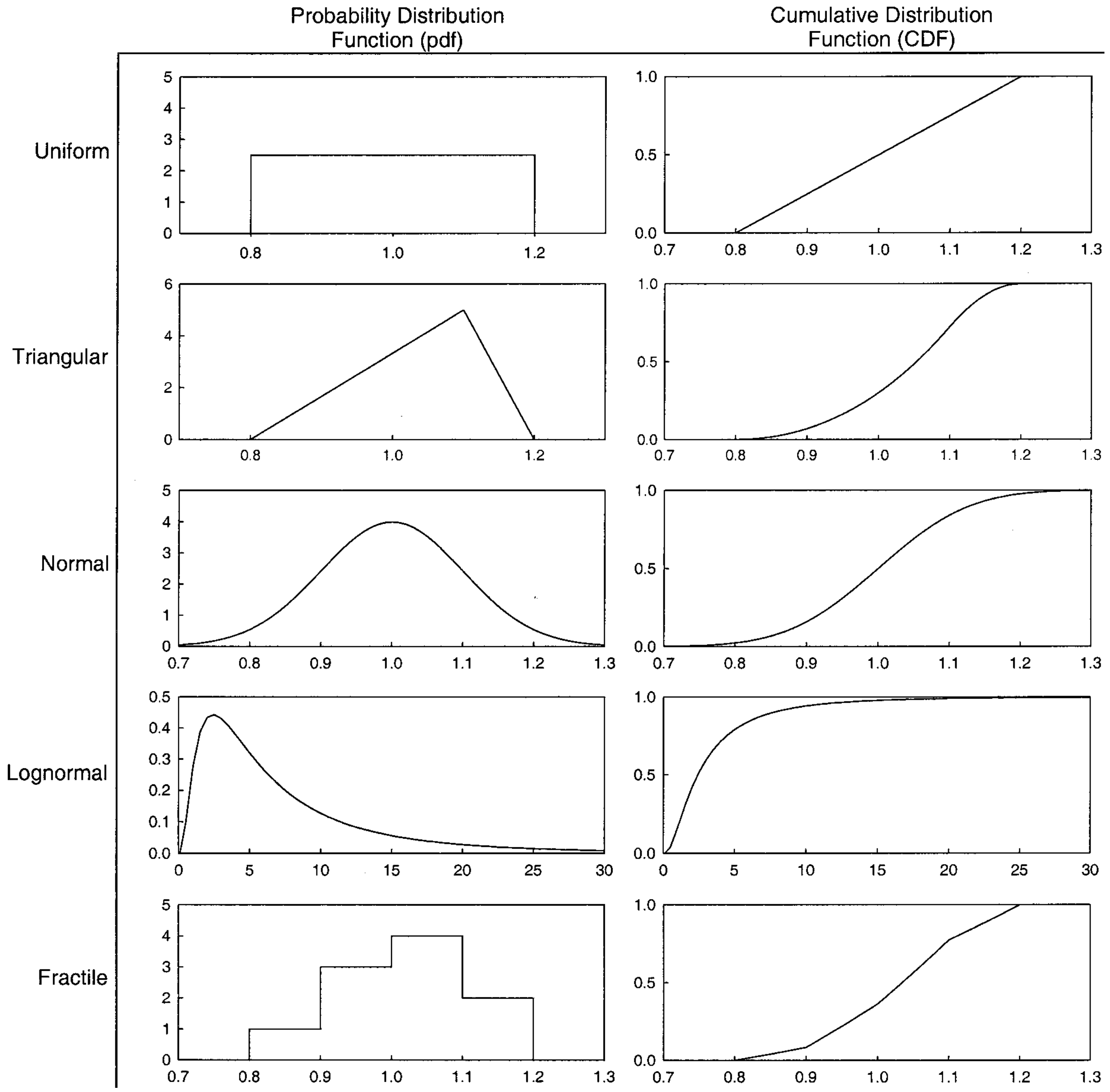

Figure 2. Examples of probabilistic distribution functions for stochastic modeling.

determine whether the probabilistic objective functional is optimum.

2. Uncertainty Propagation and Sampling

The probabilistic or stochastic modeling procedure involves (1) specifying the uncertainties in key input parameters in terms of probability distributions, (2) sampling the distribution of the specified parameter in an iterative fashion, and (3) propagating the effects of uncertainties through the process flowsheets and applying statistical techniques to analyze the results.[15]

2.1. Specifying Uncertainty Using Probability Distributions. To accommodate the diverse nature of uncertainty, different distributions can be used. Some of the representative distributions are shown in Figure 2. The type of distribution chosen for an uncertain variable reflects the amount of information that is available. For example, uniform and log-uniform distributions represent an equal likelihood of a value lying anywhere within a specified range, on either a linear or a logarithmic scale, respectively. The normal (Gaussian) distribution reflects a symmetric but varying probability of a parameter value being above or below the mean value. In contrast, the log-normal and some triangular distributions are skewed such that there is a higher probability of values lying on one side of the median than on the other. The β distribution provides a wide range of shapes and is a very flexible means of representing variability over a fixed range. Finally, in some special cases, user-specified distributions can be used to represent any arbitrary characterization of uncertainty, including the fractile distribution (i.e., fixed probabilities of discrete values).

2.2. Sampling Techniques in Stochastic Modeling. Once probability distributions are assigned to the uncertain parameters, the next step is to perform a sampling operation from the multivariable uncertain parameter domain.[4] Alternatively, one can use analytical methods to obtain the effect of uncertainties on the output. These methods, however, tend to be applicable to special kinds of uncertainty distributions and optimization surfaces only. The sampling approach provides wider applicability and is discussed below.

Monte Carlo Technique. One of the most widely used sampling techniques is the MCS technique, which is based on a *pseudo*-random number generator to approximate a uniform distribution (i.e., having equal probability in the range of 0 to 1). The specific values for each input variable are selected by inverse transformation over the cumulative probability distribution.

The main advantage of Monte Carlo methods lies in the fact that the results from any Monte Carlo simulation can be treated using classic statistical methods because of the randomness and independence of generated samples. Results can thus be presented in the form of histograms, and methods of statistical estimation and inference may be applied. Nevertheless, in most applications, the actual relationship between successive points in a sample has no physical significance. Hence, the randomness/independence for approximating a uniform distribution is not critical.[16] Moreover, the error of approximating a distribution by a finite number of samples depends on the distribution properties of the sample used for $U(0,1)$ rather than on its randomness. Once it is apparent that the uniformity property is central to the design of sampling techniques, a constrained or stratified sampling technique such as LHS becomes appealing.[17] LHS is one form of the variance reduction technique and is widely used in risk and decision analysis literature.

LHS. LHS[6] is one form of the stratified sampling technique, which can yield more precise estimates of the distribution function. In LHS, the range of each uncertain parameter Ξ_i is subdivided into nonoverlapping intervals of equal probability. One value from each interval is selected at random with respect to the probability distribution in the interval. The N_{samp} values thus obtained for Ξ_1 are paired in a random manner (i.e., equally likely combinations) with N_{samp} values of Ξ_2. These N_{samp} values are then combined with N_{samp} values of Ξ_3 to form N_{samp} triplets, and so on, until N_{samp} k-tuplets are formed. The main drawback of this stratification scheme is that, though it is uniform in one dimension, it does not provide the uniformity property in a k-dimensional hypercube.

Importance Sampling. Stratified sampling techniques ensure that more samples are generated from high-probability regions. On the other hand, importance sampling techniques[1] guarantee full coverage of high-consequence regions in the sample space, even if these regions are associated with low probabilities. This makes importance sampling techniques problem-dependent.

HSS. Recently, an efficient sampling technique, called HSS and based on Hammersley points, has been developed by Kalagnanam and Diwekar,[4,5] which uses an optimal design scheme for placing N_{samp} points on a k-dimensional hypercube. This scheme ensures that the sample set is more representative of the population, showing better uniformity in the multidimensional uncertain surface, unlike Monte Carlo, Latin hypercube, and its variant, the median LHS techniques.

Figure 3 shows samples generated by different sampling techniques on a unit square and provides a qualitative picture of how uniform the samples are. It is clear from Figure 3 that HSS has better uniformity than the other sampling techniques. The main reason for this is that the Hammersley points, which are one of the minimum discrepancy designs, provide an optimal design for placing N_{samp} points on a k-dimensional hypercube. In contrast, stratified techniques such as the LHS technique are designed for uniformity along a single dimension and then randomly paired for placement on a k-dimensional cube. Therefore, the likelihood of such schemes providing a good uniformity property on high-dimensional cubes is small.

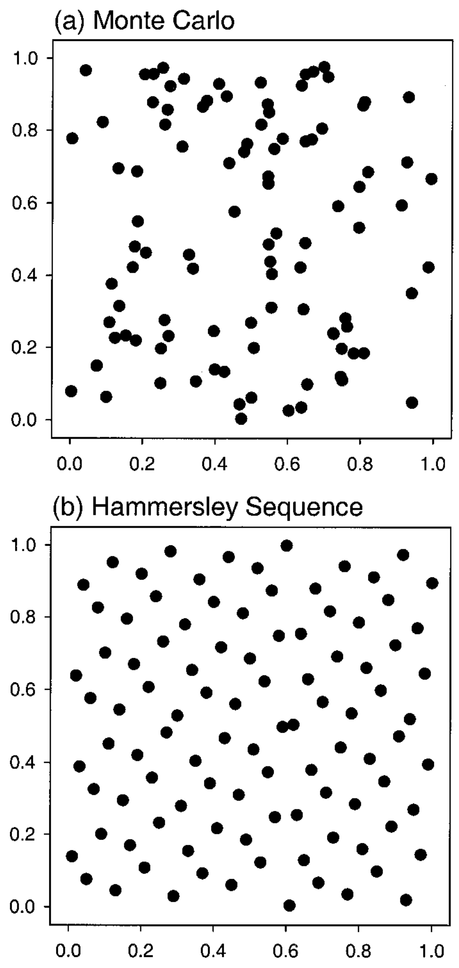

Figure 3. Sample points (100) on a unit square using (a) the MCS and (b) the HSS techniques.

The better uniformity property of HSS results in faster convergence to the “true” mean, variance, or fractiles of a function with multidimensional uncertainties. Because there are no analytical approaches (for stratified designs) to calculate the number of samples required for convergence to the “true” mean or variance, Kalagnanam and Diwekar[4] also conducted a large matrix of numerical tests and showed that the HSS technique is at least 3−100 times faster than the MCS and LHS techniques and hence is a preferred sampling technique for uncertainty analysis as well as for stochastic programming problems. This same uniformity property of the Hammersley sequence can be used to systematically improve the efficiency of the SA−NLP-based framework for optimization under uncertainty and is presented in the next section.

3. Hierarchical Efficiency Improvement

The hierarchical improvements to the SA-based algorithm for discrete optimization under uncertainty are described below. These improvements are at three levels: (1) the inner sampling loop (MCS to HSS), (2) the outer discrete deterministic optimization loop (SA to ESA), and (3) the interaction between the optimization and the sampling loops (stochastic annealing to HSTA). While the HSS technique is described in the earlier section, the other steps are described below.

3.1. ESA: Backgrounds. SA, proposed by Kirkpatrick et al.,[10] is a probabilistic method for combinatorial optimization problems based on ideas from statistical mechanics. The analogy in SA is to the behavior of physical systems in the presence of a heat bath: in physical annealing, all atomic particles arrange themselves in a lattice formation that minimizes the amount of energy in the system, provided the initial temperature is sufficiently high and the cooling is carried out slowly. At each temperature T, the system is allowed to reach thermal equilibrium, which is characterized by the probability of being in a state with energy E given by the Boltzmann distribution function:

$$Pr(E) = \frac{1}{Z}\exp\left(-\frac{1}{k_B}\frac{E}{T}\right) \qquad (4)$$

where $1/Z$ is a normalization factor and k_B is Boltzmann's constant (1.3806×10^{-23} J/K). Furthermore, at each temperature level, the system should follow Markovian moves, where the next move is only dependent on the current position and not the previous ones.

In SA, the objective function (usually cost) becomes the energy of the system, and the goal is to minimize the energy. Simulating the behavior of the system then becomes a question of generating a random perturbation that displaces a current "particle" (moving the system to another configuration). If the configuration S, representing a set of discrete decision variables that results from the moves, has a lower energy state, the move is accepted. However, if the move is to a higher energy state, the move is accepted according to the Metropolis criterion that is given by van Laarhoven and Aarts:[14]

$$\text{accepted if } Pr \le A_{ij} = \begin{Bmatrix} \exp(-\Delta E/T) & \text{if } \Delta E = E_j - E_i \ge 0 \\ 1 & \text{otherwise} \end{Bmatrix} \quad \forall i, j \in S \qquad (5)$$

where A_{ij} is the acceptance probability for generating configuration j from i. [G_{ij} is not a part of the Boltzmann energy distribution (eq 4). Instead, a sufficient condition in order to guarantee that the configurations follow eq 4 is the following balance condition (please remember that configurations i and j are vectors): $G_{ij}A_{ij}\exp(-E(i)/T) = G_{ji}A_{ji}\exp(-E(j)/T)$. Because G_{ij} is a symmetric function ($=G_{ji}$), the balance equation becomes the Metropolis criterion as shown in eq 5. Therefore, the Metropolis criterion is independent of G_{ij} as long as G_{ij} is symmetric.] The uphill moves can be accepted if a random probability (Pr) is less than or equal to A_{ij}. Hence, the Metropolis criterion implies that, at high temperatures, a large percentage of uphill moves are accepted. However, as the temperature gets colder, only a small percentage of uphill moves are accepted. Note that these uphill moves are not allowed in all local optimization algorithms. After the system has evolved to thermal equilibrium at a given temperature, then the temperature is lowered, and the annealing process continues until the system reaches a certain "freezing" temperature determined a priori. Thus, SA combines both iterative improvements in local areas and random jumpings to help ensure that the system does not become stuck in a local optimum.

Because SA is a probabilistic method, several random probability functions are involved in this procedure. A_{ij} represents the acceptance probability, and one or more generation probabilities G_{ij} are used to generate subsequent configurational moves. It is known that the efficiency of the annealing algorithm is affected little by the use of different acceptance probability distributions.[14] However, G_{ij} for generating configuration j from i at each temperature can significantly affect the overall efficiency of the annealing process. The cooling schedule is strongly dependent on G_{ij}: if the cooling is fast, then G_{ij} should cover a wider range of the configuration space at each temperature level. Generally, G_{ij} is a random probability given by the uniform distribution within the neighborhood. Thus, recent research efforts for SA improvement have been focused on modifying or changing G_{ij}. These new SA algorithms differ mainly in the choice of G_{ij} and the cooling schedule.[18]

Among the proposed SA variants, fast SA (FSA)[19] and hybrid SA (HSA)[18] are worth mentioning. G_{ij} in FSA has a Gaussian-like peak and Lorentzian long tails which can make an occasional long jump from the current configuration to increase the speed of annealing. However, this G_{ij} cannot guarantee uniform coverage of the moves over the configuration surface. HSA applies the hybrid Monte Carlo method to obtain G_{ij}, in which the design variable $\mathbf{x}$ and a Gaussian-like auxiliary momenta $\mathbf{p}$ are mapped using Hamilton's equations of motion. The acceptance probability is similar to the Metropolis criterion, but the energy difference ΔE is replaced by the Hamiltonian function difference $\Delta\mathscr{H}(\mathbf{x},\mathbf{p})$. Although this algorithm is found to be very fast, HSA requires an evaluation of the derivative of the objective function, $-\partial f(\mathbf{x})/\partial x_i$, for mapping and hence destroys one of the advantages of the standard SA algorithm: that SA does not require derivative information.

3.2. ESA. The G_{ij} generation of the current annealing algorithms rely on pseudo-random number generators such as the crude MCS, which can result in clustered moves over the configuration surface, as shown in Figure 3a. Therefore, more moves or generations are required to cover the whole configuration surface evenly, and this results in a longer Markov chain length (number of moves) at each temperature level. As described in the previous section, the HSS technique (Figure 3b), a *quasi*-random number generator, can generate uniform samples over the k-dimensional hypercube. In this work, we use HSS to generate the G_{ij} for SA and derive a new SA algorithm called ESA. It should be noted that, in using the HSS for ESA, one has to keep the k-dimensional uniformity property of HSS by generating the k probabilities for G_{ij} from configuration i in a complete space of a Markov chain length. To illustrate this, consider the following stochastic IP (SIP) problem from Birge and Louveaux:[20]

$$\min\ -2y_1 - 3y_2 \qquad (6)$$
$$\text{s.t.}\ \ y_1 + 2y_2 \le \xi_1 - x_1$$
$$y_1 \le \xi_2 - x_2$$
$$y \ge 0,\ \text{integer}$$

where $\xi = (2, 2)^T$ or $(4, 3)^T$, each with an equal probability, and the current iterate point is at $\mathbf{x} = (0, 1)^T$. The optimal solution is at $\mathbf{y} = (2, 1)^T$. The implementation of SA to this example requires four random probabilities: one for assigning ξ, two for moving $\mathbf{y}$, and one for the Metropolis criterion. In traditional SA algorithms, these probabilities are established by generating a *pseudo*-random number based on MCS at a time. However, to exploit the k-dimensional uniformity

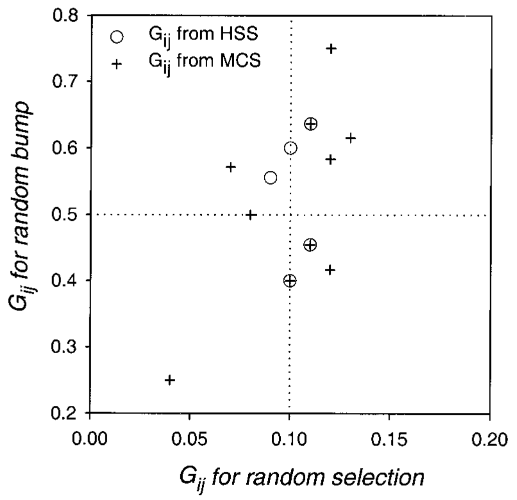

Figure 4. Generation probabilities, G_{ij}, for HSS and MCS, example I, with ND = 10 (several points of HSS are overlapping).

of HSS, these four probabilities are generated together by generating all of the N (number of moves at each temperature) quasi-random numbers based on HSS at once and then using them one at a time.

To compare the performance of ESA with SA, both MCS and HSS ways are used to generate the four probabilities. The average value of the total moves is obtained from six different initial conditions. The total moves for the SIP with the HSS technique is found to be 691, while the MCS technique required a total of 1503 moves, a significant savings.

To further demonstrate the efficiency improvement of ESA over SA, the following examples are tested:

Example I:

$$f(y) = \sum_{i=1}^{\mathrm{ND}} y_i^2 \tag{7}$$

Example II:

$$f(y) = \sum_{i=1}^{y_1} \{(y_1 - 3)^2 + [y_2(i) - 3]^2 + [y_3(i) - 3]^2\} \tag{8}$$

Example III:

$$f(x,y) = \sum_{i=1}^{\mathrm{ND}} \left(x_i - \frac{i}{\mathrm{ND}}\right)^2 + \sum_{i}^{\mathrm{ND}} y_i^2 - \prod_{i}^{\mathrm{ND}} \cos(4\pi y_i) \tag{9}$$

where **x** denotes a vector of continuous variables and **y** denotes a vector of discrete variables. Example I is a multidimensional parabolic function taken from Salazar and Toral[18] where ND is the dimension of the function. This example has one global optimum at zero for all decision variables. The second example, a pure combinatorial problem, appears in work by Painton and Diwekar,[21] in which the objective space is *discontinuous* with respect to y_1. This example also has one global optimum when y_1 is 3 and all $y_2(i)$ and $y_3(i)$ are 3 (i = 1, ..., y_1). Because the third example involves discrete and continuous decision variables, this example function is a MINLP problem. As an alternative to MINLP, a coupling of SA and NLP, SA−NLP,[22] is used to solve this problem. This function has one global optimum [$f(x,y) = -1$] and many local optima.

Figure 4 shows G_{ij} probabilities of HSS and MCS for example I with ND = 10. Because there are 10 elements in **y**, the ideal probability of selecting any element is 0.1, and the value of a selected element can be randomly bumped up or down with a probability of 0.5. Thus, the dotted lines in this figure are used to show ideal two-dimensional G_{ij} probabilities, while circle and cross symbols are for the actual generation probabilities from HSS and MCS, respectively. Because HSS can generate more uniform samples in the multivariate space, G_{ij} probability, generated using HSS, is closer to the ideal probabilities than the results from MCS. In contrast, G_{ij}, generated from MCS, has a high deviation from the ideal probabilities, and hence, MCS requires a large number of moves to approximate the ideal probabilities.

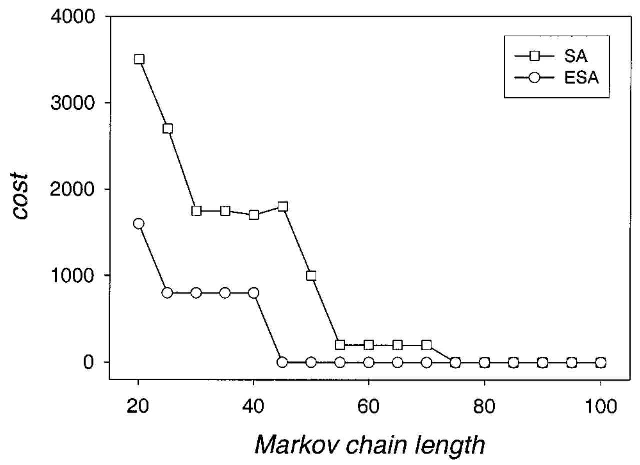

Figure 5. Cost trajectories of ESA and SA with different Markov chain lengths (ND = 10).

Table 2. Comparison of SA and ESA in Terms of the Total Number of Moves

		total number of moves		
example	ND	SA	ESA	% savings
I	10	3109	1536	50.6
	50	18598	12995	30.1
	100	39790	23000	42.0
II		1285	592	54.0
III	10	2742	1700	38.0
	40	13238	6985	47.2

Figure 5 shows trajectories of the objective value for example I with different Markov chain lengths. ESA found the global solution with a Markov chain length of 45 at each temperature, while the traditional SA exploited a Markov chain length of 75 to reach the same solution. As can be seen, ESA provides a significant reduction in moves at each temperature. Table 2 presents the efficiency improvements of the ESA algorithm in terms of the total number of configurational moves. The results obtained here are the average values for 10 different initial conditions used. From this table, it can be said that ESA is approximately 30−54% more efficient than SA.

3.3. ESTA. Diwekar and co-workers[11−13] recently proposed stochastic annealing, a variant of SA, that is designed to solve discrete optimization problems under uncertainty. This algorithm is designed to efficiently optimize a probabilistic objective function by balancing the solution accuracy and computational efficiency. In the stochastic annealing algorithm, the optimizer obtains not only the decision variables but also the number of samples required for the stochastic model. Stochastic annealing uses a substitute objective function, which involves the true value of the probabilistic objective function augmented by a penalty term involving an error in sampling related to the number of samples.

Table 3. Main Steps in the Stochastic Annealing Algorithm

```
1. initialize variables: $T_{initial}$, $T_{freeze}$, $\alpha$, accept and reject limits, and initial configuration ($S$)
2. if $T > T_{freeze}$, then perform the following loop $N$ (number of moves at a given temperature) times
   2.1. generate a move $S'$ from the current configuration $S$ as follows:
      2.1.1. select the decision variables for new configuration $S'$ (0−1, integer, discrete, and continuous variables)
      2.1.2. select the parameter level *randomly* within the neighborhood
   2.2. select the number of samples $N_{samp}$ by a random move. If rand(0, 1) $\leq$ 0.5,
      then
         $N_{samp} = N_{samp} + 5 \times \text{rand}(0, 1)$
      else
         $N_{samp} = N_{samp} - 5 \times \text{rand}(0, 1)$
   2.3. generate $N_{samp}$ samples of the uncertain parameters
   2.4. perform the following loop $N_{samp}$ times
      2.4.1. run the model
      2.4.2. calculate the objective function cost, $f(\mathbf{x},\xi)$
   2.5. evaluate the expected value $E_\xi f(\mathbf{x},\xi)$ and the variance of the cost function
   2.6. generate the weighting function $b(t) = b_0/k^t$
   2.7. calculate the modified objective function:
         $z(S') = \frac{1}{N_{samp}} \sum_{j=1}^{N_{samp}} f(\mathbf{x},\xi^j) + b(t) \frac{2\sigma}{N_{samp}^{0.5}}$
   2.8. let $\Delta E = z(S') - z(S)$
   2.9. if $\Delta E \leq 0$, then accept the move and set $S = S'$; else if ($\Delta E > 0$), accept with a probability $\exp(-\Delta E/T)$
3. set $T = \alpha T$; if $T > T_{freeze}$, return to step 2
4. stop
```

Thus, the stochastic programming problem ($\mathbb{P}3$) can be modified to a new one ($\mathbb{P}4$) that consists of the expected cost function $E_\xi f(\mathbf{x},\xi)$ and the penalty term $b(t)\,\epsilon$ and is given by

$$(\mathbb{P}4) \qquad \min z = E_\xi f(\mathbf{x},\xi) + b(t)\,\epsilon = \frac{1}{N_{samp}} \sum_{j=1}^{N_{samp}} f(x,\xi^j) + b(t)\,\epsilon \qquad (10)$$

The penalty function is composed of the weighting function $b(t)$ and the error bandwidth (confidence interval) ϵ of the sampling method. The weighting function $b(t)$, governed by the cooling schedule, can be expressed in terms of the annealing temperature level (t). At high temperatures, the sample size (N_{samp}) can be small because the algorithm is exploring the functional topology in the configuration space. Thus, at this stage, computational efficiency is more important than solution accuracy. As the system gets cooler, the algorithm searches for the global optimum. Consequently, it is necessary to take more samples to get more accurate and realistic objectives. Based on these properties, an exponential function for $b(t)$ can be devised as

$$b(t) = b_0/k^t \qquad (11)$$

where b_0 is a small constant (e.g., 0.001) and k is a constant (e.g., 0.92) which governs the rate of increase. These two empirical parameters depend on the cooling schedule and must be predetermined through experimentation such that the penalty term is less than 5% of the real objective function.

The error bandwidth of *random* samples can be estimated from the following equation based on classic statistical methods. No matter what the distribution of X, the central limit theorem allows one to calculate the probabilistic error bands on the expected value of the random samples, given by $(\bar{x} - z_{1-\alpha/2}\sigma/\sqrt{N_{samp}},\ \bar{x} + z_{1-\alpha/2}\sigma/\sqrt{N_{samp}})$, where $z_{1-\alpha/2}$ is a standard normal variate such that the range $(-z_{1-\alpha/2}, z_{1-\alpha/2})$ encloses $1-\alpha$ probability of the unit normal distribution. For a higher confidence interval such as $1-\alpha = 0.95$, the standard normal variate is approximately equal to the constant 2, resulting in the following equation:

$$\epsilon_{MCS} \propto 1/\sqrt{N_{samp}} \qquad (12)$$

In our first approach for the development of stochastic annealing algorithms, we have used this error bandwidth in the penalty function, allowing stochastic annealing to control the number of samples.

The main steps of the STA algorithm are summarized in Table 3. A new configuration S' is generated from the current configuration S based on the given generation probability G_{ij} (steps 2.1.1 and 2.1.2). N_{samp} in step 2.2 can be randomly increased or decreased but eventually is governed by the weighting function used in the penalty term. After N_{samp} uncertain samples are determined and generated, the model runs N_{samp} times with different uncertain parameters to find $E_\xi f(\mathbf{x},\xi)$. The new stochastic objective function shown in eq 13 is then used to evaluate the effect of uncertainties on the optimization problems. The remaining steps are the same as the SA algorithm steps.

The idea for reducing the Markov chain length by using HSS, as done in ESA, is exploited here for the new stochastic annealing algorithm. Additionally, HSS is used in the inner sampling loop for uncertainty analysis. This is likely to reduce the number of samples needed to calculate the objective function accurately. We call this two-level improved stochastic annealing algorithm the ESTA algorithm.

Consider the nonconvex deterministic MINLP problem given in eq 9 that incorporates uncertainties (ξ). This problem results in the following objective function, a stochastic MINLP, to be minimized.

$$f(\mathbf{x},\mathbf{y},\xi) = \sum_{i=1}^{ND}\left(\xi_i x_i - \frac{i}{ND}\right)^2 + \sum_{i=1}^{ND}(\xi_i y_i^2) - \prod_{i=1}^{ND}\cos(4\pi\xi_i y_i) \qquad (13)$$

Because this example problem has a continuous decision vector ($\mathbf{x}$) and a discrete decision vector ($\mathbf{y}$), a coupling of STA and NLP (STA−NLP), similar to SA−NLP, is used.

Table 4. Comparison of STA and ESTA in Terms of the Average Number of NLP Subprogram Calls

ND	STA	ESTA	% savings	Markov chain length
2	6218	3298	47.1	100
10	5670	3265	42.4	100
10	4015	1439	64.2	minimum

Table 4 shows the total NLP subprogram calls using the STA and the new ESTA algorithms. Even for the same Markov chain length (the first two rows), ESTA is approximately 45% more efficient than the STA approach. This reduction is mainly attributed to the faster convergence property of the HSS technique. Further reduction can be achieved when a minimum Markov chain length for each STA is determined (the last row). To find a minimum Markov chain length, the concept of cost of epoch proposed by Skiscim and Golden[23] is applied to determine a *pseudo* thermal equilibrium at each temperature level. A current epoch (summation of certain consecutive cost values) is compared with the previous epoch, and if the difference between epochs is within tolerance, then we assume *pseudo* thermal equilibrium. The minimum Markov chain lengths at each temperature level were 60 for STA and 40 for ESTA, and hence the efficiency improvement reaches up to 65%. This reduction is a combined effect of the uniformity property of the HSS-based random number generator and the faster convergence property of HSS.

3.4. HSTA. The error bandwidth used in the STA and ESTA algorithms presented in the previous section are based on classic statistical methods (eq 12). Classical statistical methods provide good estimates for the bounds (confidence intervals) of the MCS but may not be applicable to other less random, yet uniform, sampling techniques. It has been shown that classic statistical methods used to characterize the error bandwidth for any confidence level of HSS overestimate either the confidence intervals or bounds.[13,24] Hence, the combination of the HSS technique and the classic error bandwidth (ϵ_{MCS}) in the ESTA algorithm is not the most efficient approach, and further improvement is made possible by using a modified error bandwidth that is specific to the HSS technique.

An approach to quantify the error bandwidth for any probabilistic function is well outlined in the references[13,24] and based on concepts from fractal geometry. In this literature, it was established that the relative error bandwidth of the Monte Carlo and HSS techniques shows a scaling relationship ($N_{samp}{}^{d}$) with respect to the number of samples. For the HSS technique, the exponent for the error bandwidth for the mean is found to be −1.8 from comprehensive simulations, and thus the HSS-specific error bandwidth is given by

$$\epsilon_{HSS} \propto \frac{1}{N_{samp}{}^{1.8}} \qquad (14)$$

Note that d of the MCS technique is estimated as −0.5. That is exactly the same value obtained from the classic statistical methods as shown in eq 12.

A new variant of stochastic annealing, HSTA, therefore, incorporates HSS for the generation probability G_{ij}, HSS in the inner sampling loop for N_{samp} determination, and the HSS-specific error bandwidth in the penalty term. To evaluate the efficiency improvement by this new HSTA algorithm, the same probabilistic objective equation (eq 14) is used.

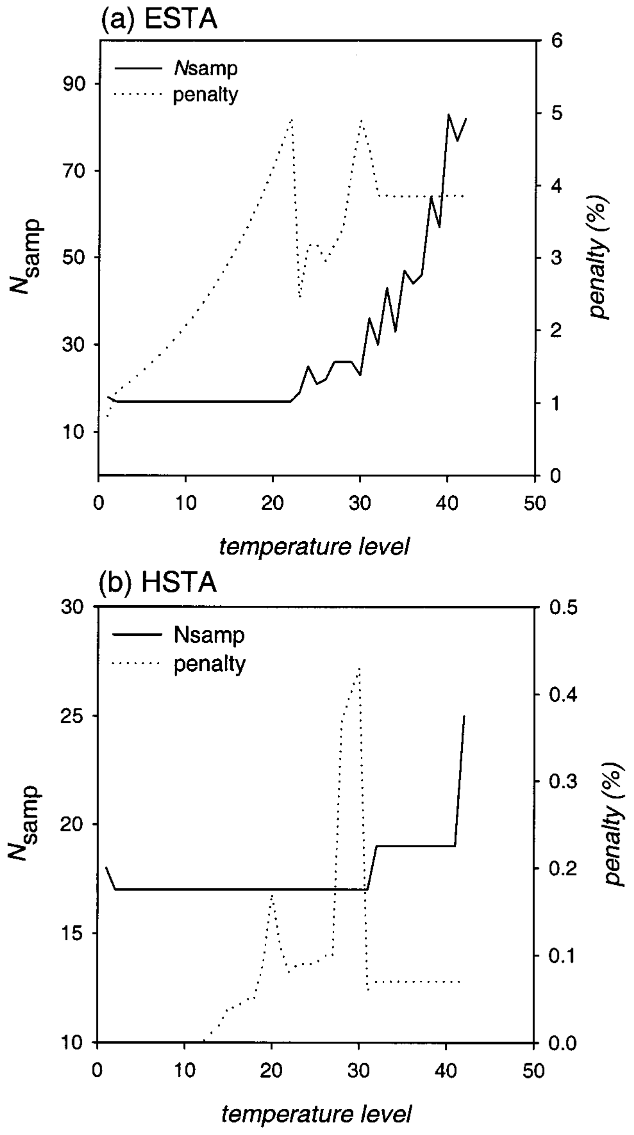

Figure 6. Trajectories of N_{samp} and penalty percentage of the ESTA and HSTA algorithms.

Table 5. Comparison of Levels of Algorithm Improvements (ND = 10)

algorithm	problem	total moves	% savings
SA + fixed N_{samp}[a]	ℙ3	274200	
ESA + fixed N_{samp}	ℙ3	170000	38.0
STA	ℙ4	5670	97.9
ESTA	ℙ4	3265	98.8
HSTA	ℙ4	1793	99.3

[a] Fixed N_{samp} = 100.

Figure 6 shows trajectories of N_{samp} and the penalty term in a percentage of the objective function for the ESTA and HSTA algorithms. From this figure we can see that N_{samp} is significantly reduced when stochastic annealing based algorithms are used. A further decrease in N_{samp} is observed for the HSTA algorithm because of a reduced error bandwidth. Note that there is a large difference in the penalty percentage values between ESTA and HSTA. Table 5 shows results comparing the hierarchical improvements from stochastic optimization with fixed N_{samp} to the newest HSTA algorithm. In the table, problem ℙ3 is the stochastic optimization with fixed N_{samp}, while problem ℙ4 is the stochastic optimization with automatically varying N_{samp}. The fixed N_{samp} is 100 in this comparison, while the varying N_{samp} at

ℙ4 algorithms spans from 15 to 90. The base case for comparison is SA with fixed N_{samp}, in which the objective function in SA is modified to ℙ3. The base case requires 274 000 moves to find the optimal solution **x***. This stochastic programming can be improved up to four levels: ESA with fixed N_{samp}, STA, ESTA, and HSTA. A large improvement in computational efficiency can be achieved if ESA is used instead of SA. We can further see that there is significant improvement when we change the problem type from ℙ3 to ℙ4. The improvement is over 97% and is mainly attributed to the use of the penalty term and the properties of the HSS sampling technique. Among the ℙ4 algorithms, HSTA is 68% faster than the basic STA algorithm.

4. Conclusion

This paper presented hierarchical improvements in the SA-based algorithms for solving large-scale discrete optimization problems under uncertainty. At first, the deterministic SA was modified to ESA, which exploited the uniformity property of the HSS technique, and resulted in a shorter Markov chain length at each temperature level. The same concept was then applied to the stochastic annealing algorithm, resulting in the new ESTA algorithm that provided the tradeoff between computational efficiency and solution accuracy by automatically determining N_{samp}. In addition, the faster convergence property of HSS is also incorporated in this algorithm. The efficiency is improved further in the HSTA algorithm by using the HSS-specific error bandwidth in the penalty term of the probabilistic objective functional. For stochastic MINLP problems, HSTA was coupled with the NLP algorithm. For the test problems considered in this study, the combined improvement of all of the steps in the hierarchy was shown to result in a 99.3% savings in computational time over the traditional stochastic optimization algorithms. Therefore, the HSTA algorithm can be a useful tool for large-scale combinatorial stochastic programming problems. A real world problem of solvent selection under uncertainty for acetic acid extraction is presented in part 2 of this series.

Acknowledgment

We greatly appreciate the National Science Foundation for funding this research (Goali Project CTS-9729074).

Literature Cited

(1) Dantzig, G.; Glynn, P. Parallel processors for planning under uncertainty. *Ann. Oper. Res.* **1990**, *22*, 1.

(2) Higle, J.; Sen, S. Design of environmentally benign processes: Integration of solvent design and separation process synthesis. *Comput. Chem. Eng.* **1999**, *23*, 1395.

(3) Shapiro, A.; Homem-De-Mello, T. On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs. *INFORMS J. Optim.* **2000**, *11*, 70.

(4) Kalagnanam, J. R.; Diwekar, U. M. An efficient sampling technique for off-line quality control. *Technometrics* **1997**, *39*, 308.

(5) Diwekar, U.; Kalagnanam, E. Efficient sampling technique for optimization under uncertainty. *AIChE J.* **1997**, *43*, 440.

(6) Iman, R. L.; Conover, W. J. Small-sample sensitivity analysis techniques for computer models, with an application to risk assessment. *Commun. Stat., Part A: Theory Methods* **1982**, *17*, 1749.

(7) Nemhauser, G. L.; Wolsey, L. A. *Integer and Combinatorial Optimization*; Wiley: New York, 1988.

(8) Geoffrion, A. Generalized Benders decomposition. *J. Optim. Theory Appl.* **1972**, *10*, 237.

(9) Duran, M.; Grossmann, I. E. An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Math. Program.* **1986**, *36*, 307.

(10) Kirkpatrick, S.; Gelatt, C. D., Jr.; Vecchi, M. P. Optimization by simulated annealing. *Science* **1983**, *220*, 671.

(11) Painton, L. A.; Diwekar, U. M. Stochastic annealing for synthesis under uncertainty. *Eur. J. Oper. Res.* **1995**, *83*, 489.

(12) Chaudhuri, P.; Diwekar, U. M. Process synthesis under uncertainty: A penalty function approach. *AIChE J.* **1996**, *42*, 742.

(13) Chaudhuri, P.; Diwekar, U. M. Synthesis approach to the determination of optimal waste blends under uncertainty. *AIChE J.* **1999**, *45*, 1671.

(14) van Laarhoven, P.; Aarts, E. *Simulated Annealing: Theory and Applications*; Reidel Publishing Co.: Dordrecht, The Netherlands, 1987.

(15) Diwekar, U.; Rubin, E. Stochastic modeling of chemical processes. *Comput. Chem. Eng.* **1991**, *15*, 105.

(16) Knuth, D. E. *The Art of Computer Programming: Fundamental Algorithms*; Addison-Wesley: Reading, MA, 1973; Vol. 1.

(17) Morgan, G. M.; Henrion, M. *Uncertainty—A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*; Cambridge University Press: Cambridge, England, 1990.

(18) Salazar, R.; Toral, R. Simulated annealing using Hybrid Monte Carlo. *J. Stat. Phys.* **1997**, *89*, 1047.

(19) Szu, H.; Hartley, R. Fast simulated annealing. *Phys. Lett. A* **1987**, *3*, 157.

(20) Birge, J. R.; Louveaux, F. *Introduction to Stochastic Programming*; Springer: New York, 1997.

(21) Painton, L. A.; Diwekar, U. M. Synthesizing optimal design configurations for a brayton cycle power plant. *Comput. Chem. Eng.* **1994**, *18*, 369.

(22) Narayan, V.; Diwekar, U.; Hoza, M. Synthesizing optimal waste blends. *Ind. Eng. Chem. Res.* **1996**, *35*, 3519.

(23) Skiscim, C. C.; Golden, B. L. Optimization by simulated annealing: A preliminary computational study for the TSP. *NIHE Summer School on Combinatorial Optimization*; NIHE: Dublin, 1983.

(24) Diwekar, U. M. An efficient approach to optimization under uncertainty. *Comput. Optim. Appl.* **2002**, submitted for publication.

Received for review February 21, 2001
Revised manuscript received September 6, 2001
Accepted November 29, 2001

IE0101689