

Improved Genetic Algorithms for Deterministic Optimization and Optimization under Uncertainty. Part I. Algorithms Development

U. M. Diwekar* and W. Xu

Center for Uncertain Systems, Tools for Optimization and Management (CUSTOM), Vishwamitra Research Institute, Westmont, Illinois 60559

This paper proposes three new variants of genetic algorithm to solve deterministic and stochastic optimization problems. In these algorithms, a new and efficient sampling technique, Hammersley sequence sampling (HSS), is utilized in the initial population generation and population updating. Additionally for stochastic optimization problems, HSS is also used for propagation of parametric uncertainties through the model. The better uniformity properties of HSS are exploited in developing the efficient genetic algorithm (EGA) to solve deterministic optimization problems. A case study has been performed in this work to show that EGA has better performance than its traditional counterpart, in which the random number generator from Monte Carlo sampling is commonly employed. For stochastic optimization problems, the Hammersley stochastic genetic algorithm (HSGA) coupled with better confidence interval of the samples has been introduced. Case studies show that the new algorithm outperforms (1) the stochastic genetic algorithm (SGA) which employs Monte Carlo sampling and (2) the efficient stochastic genetic algorithm (ESGA), where HSS is used together with Monte Carlo confidence intervals. This is due to the uniformity and faster convergence properties of HSS utilized in HSGA. The exercise demonstrates that HSGA has the best performance while SGA displays the worst performance. The second part in this series of papers describes two solvent selection models, and solvent selection with and without uncertainty is solved using the new algorithms.

1. Introduction

Many optimization problems, which include a large number of continuous or discrete design variables, fall into the category of integer programming (IP), mixed integer linear programming (MILP), and mixed integer nonlinear programming (MINLP). Branch and bound (BB), generalized bender's decomposition (GBD), and outer-approximation (OA)¹ are generally used for solving IP, MILP, and MINLP problems. However, problems occur when (1) functions do not satisfy convexity conditions, (2) systems have large combinatorial explosion, or (3) the solution space is discontinuous. One probabilistic optimization technique, named evolutionary algorithm (EA), that has been developed on the basis of Darwin's natural selection and Mendel's genetics, provides an alternative to the mixed integer programming techniques such as the BB, GBD, and OA. The class of EAs includes genetic algorithms (GAs),² genetic programming,³ evolutionary programming,⁴ and evolutionary strategy.^{5,6} Among all the EAs, the GA is most widely used. The GA was first introduced by Holland.² It has been receiving increased attention because of a series of successful applications in different disciplines such as biology, medicine, and different branches of engineering.⁷

The basic idea of GA is to start from a population instead of a single point in the potential solution space of a specific problem and allow that population to evolve from generation to generation by genetic operators such as selection, crossover, and mutation until the stopping criteria are satisfied. In the evolving process, GA uses a structured yet randomized information exchange to

form a search direction. The behavior of GA is characterized by a balance between exploitation and exploration. This balance is strongly affected by strategy parameters such as population size, crossover, and mutation rate as well as the mechanism employed for (1) choosing initial population, (2) representing individuals, and (3) performing evolution. The improvements proposed here are based on both adaptation of these features and incorporation of problem-specific properties similar to a scheduling problem. A number of modifications arising from the above considerations have been developed in the last several decades to obtain better performance such as real number encoding,⁸ simulated binary crossover operator,⁹ parameter adaptation,¹⁰ and hybrid genetic algorithm.¹¹

In this paper, a new strategy considering the improvements on population diversity and uniformity of random operations is proposed by applying a new sampling mechanism, Hammersley sequence sampling (HSS),¹² to both deterministic optimization and optimization under uncertainty problems. This new sampling mechanism has been shown to exhibit better uniformity over the multivariate parameter space. Furthermore, it has also been proven that the number of samples required to converge is less than the crude Monte Carlo sampling (MCS) and the variance reduction techniques such as latin hypercube sampling (LHS).¹³ The GA especially benefits from these features, because the calculation of cost functions is expensive due to the fact that it starts from a population instead of a single point.

2. Overview of the GA

The GA was first developed by Holland.² In general, there are five components in it:⁸ (1) a genetic representation of solutions to the problem, (2) a way to create

* To whom correspondence should be addressed. Tel.: (630) 515-8773. E-mail: urmila@vri-custom.org.

an initial population of solutions, (3) an evaluation function rating solutions in terms of fitness, (4) genetic operators that generate new individuals, and (5) values for the parameters of GAs.

The general procedure of GAs can be summarized as follows: At $t = 0$, (1) generate the initial population $P(t)$ and (2) evaluate $P(t)$.

When the termination condition is not satisfied, (1) recombine $P(t)$ to generate new individuals, that is, children $C(t)$, (2) evaluate $C(t)$, (3) select $P(t + 1)$ from $P(t)$ and $C(t)$, and (4) set $P(t) = P(t + 1)$, where $P(t)$ is parent, $C(t)$ is children, and t is generation. Encoding plays an important role in GA. The original GA uses binary encoding. But with increasing utilization of GA in more complex problems, new encoding methods have been developed, such as real-number encoding, integer or literal permutation encoding, and general data structure encoding.¹⁴ The initial population $P(0)$ can be chosen heuristically or randomly. If it is chosen randomly, a corresponding sampling technique like the MCS method is needed to propagate the initial population. To develop the next generation, genetic operators recombine the old generation to form a new one. Selection, crossover, and mutation are three operators generally used. Selection is a process in which the individuals selected based on their fitness are copied to the next generation. In this process, fitter solutions have a higher chance to contribute to the next generation while the unfit string patterns are phased out. Selection must work to strike a balance between selection pressure and population diversity. Selection plays an important role in exploitation, while crossover and mutation play important roles in exploration. The crossover operator randomly exchanges parts of the genes of two parents to generate two new children. Crossover serves two complementary search functions. First, crossover can provide new information about the hyperplanes already represented earlier in the population. By evaluating new solution strings, GA gathers further knowledge about these hyperplanes. Second, crossover introduces representatives of new hyperplanes into the population. If this new hyperplane is a high-performance area of the search space, the evaluation of new more-fit population will lead to further exploration in this subspace. The mutation operator performs a random gene change. A low level of mutation serves to prevent any given bit position from remaining fixed indefinitely to a single value in the entire population, while a high level of mutation essentially yields a random search.

Termination criteria can be specified as the permissible maximum number of generations or an acceptable approximated solution. The evolution process can also be terminated when there is no obvious change of best individuals found after a fixed number of generations. GA parameters such as population size, crossover ratio, and mutation ratio are key factors in the tradeoff between exploitation and exploration.²

3. Sampling Method

MCS, LHS,¹³ and importance sampling¹⁵ are widely used sampling techniques. Recently, an efficient sampling technique called HSS, based on Hammersley points, has been developed, which uses an optimal design scheme for placing N_{samp} points on a k -dimensional hypercube more uniformly. This scheme ensures that the sample set is more representative of the

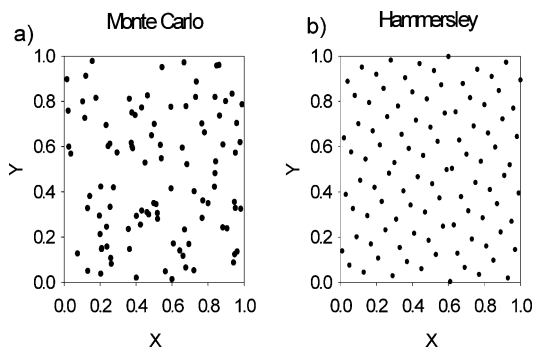


Figure 1. Comparison between (a) the MCS sampling and (b) the HSS sampling techniques.

population, showing better uniformity in the multidimensional parameter surface compared to MCS, LHS, and its variant, the median latin hypercube 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. The sampling results on a unit square using the MCS technique and HSS technique are shown in Figure 1. It shows that samples generated by the HSS technique achieve better uniformity in parameter space and results in faster convergence to the “true” mean, variance, or fractiles.¹²

4. Hierarchical Improvements of the GA

The hierarchical improvements to the GA for both deterministic optimization and stochastic optimization cases include the following aspects: (1) efficient genetic algorithm (EGA), (2) efficient stochastic genetic algorithm (ESGA), and (3) efficient Hammersley stochastic genetic algorithm (HSGA). All of the three improved variants above are based on application of the HSS sampling technique, similar to the work on the improvements to simulated annealing by employing HSS.¹⁶

4.1. Efficient GA. 4.1.1. Overview. Population diversity plays an important role in the performance of GA. The uniformity property of the HSS technique can be used in this step to avoid initial populations clustered in a small region of the potential solution space. Applying the HSS technique in selection, crossover, and mutation rather than random probability functions in these operations results in additional improvements. Because the HSS method shows more uniformity in generating samples over k -dimensional hypercube, its application here ensures a more uniform exploration and exploitation of solution space instead of a bias toward a particular region of solution space or chromosome, which would trap the GA in a local optimum. Another important issue that needs to be addressed here is that it is imperative that one maintain the k -dimensional uniformity property of HSS by generating N quasi-random numbers needed in each generation simultaneously for all probabilities instead of one quasi-random number at each time for N times in each generation. EGA is developed by implementing these new features. The algorithm is summarized as follows:

At $t = 0$, (1) use the HSS method to generate initial population $P(t)$, keeping the k -dimensional uniformity property intact, and (2) evaluate $P(t)$.

When the termination condition is not satisfied, (1) Recombine $P(t)$ to generate $C(t)$. Use the HSS method to generate random moves in selection, crossover, and mutation steps, keeping the k -dimensional uniformity

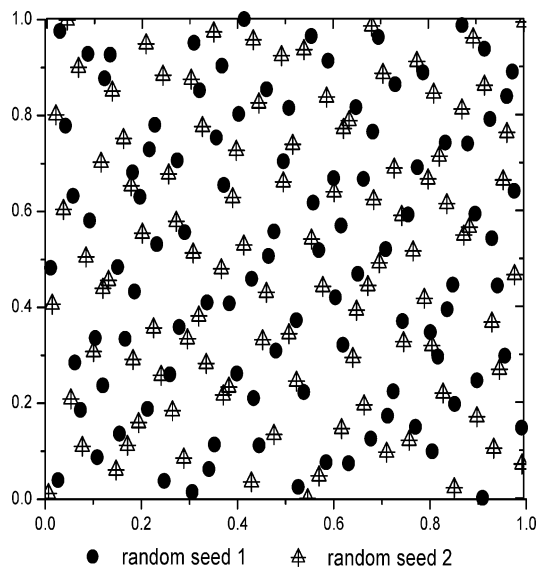


Figure 2. Effect of different seed on the HSS generated samples.

property intact. (2) Evaluate $C(t)$. (3) Select $P(t+1)$ from $P(t)$ and $C(t)$. (4) Set $P(t) = P(t+1)$.

The following three examples are used in this paper to evaluate the performance of the newly developed genetic algorithm EGA.

Example 1:

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

Example 2:

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

Example 3:

$$f(x, y) = \sum_{i=1}^{ND} \left(x_i - \frac{i}{ND} \right)^2 + \sum_{i=1}^{ND} y_i^2 - \prod_{i=1}^{ND} \cos(4\pi y_i) \quad (3)$$

where x denotes a vector of continuous variables, y denotes a vector of discrete variables, and ND is the dimension of the examples. Example 1 is a multidimensional parabolic function¹⁷ that has one global optimum at 0 for all decision variables equal to 0. The second example, a pure combinatorial problem,¹⁸ has one global minimum 0 when all y_1 , $y_2(i)$, and $y_3(i)$ are equal to 3. The third example is a MINLP problem that has one global minimum -1 .

4.1.2. Effect of Random Seed on EGA. Theoretically, the random seed which is used to propagate samples should have no impact on the performance of HSS. However, we have modified the HSS, and this modified HSS fluctuates with random seed. The existence of limitation on sample size makes it impossible for the samples to cover the whole search space. This limitation directly induces a nonoverlapping distribution of a different set of samples on the k -dimensional space, though each set of samples is uniformly distributed. From Figure 2 we can observe the difference in the two sets of samples generated from different random seeds. The performance of EGA depends on the specific value of each sample and, hence, will have a different convergence path due to the different sets of samples used.

Table 1. Effect of Seed on the Performance of EGA for Example 3 with $ND = 5$

	random seed				dynamic seed
	425 001	888 868	25 689	256	
objective value	-0.9994	-0.9983	-0.9970	-0.9958	-0.9992
generation	22	200	200	200	124

Table 2. Comparison of EGA and MGA with Examples 1–3^a

example	ND	total number of generations	optimal value/generation		theoretical optima
			MGA	EGA	
1	10	150	2/150	0/40.6	0
1	20	200	26/200	0/195.2	0
2		200	1/200	0/16.4	0
3	5	200	-0.9987/200	-0.9991/88.6	-1

^a The results of EGA are the average of five runs with the dynamic seed value.

The effect of random seed is tested on example 3 and summarized in Table 1, which shows that a different random seed for HSS produces a different convergence performance. There are already several parameters such as population size, crossover rate, and so forth, that need to be tuned to reach the best performance. The existence of additional parameters would make the GA less flexible. To increase the diversity of new populations and decrease the effect of different seeds, the strategy of parameter adaptation^{2,19} is used for random seed adaptation. Because there is no relationship between performance and random seed value, changing the random seed at each generation would not outperform a randomly changing seed value. To simplify the algorithm while keeping the random seed as diverse as possible, the dynamic seed is used, which for simplicity takes the value of the system time. It shows that it may not ensure the best performance, but it is better than the average case. And further, it avoids the time-consuming testing process.

4.1.3. Efficiency Improvement of the EGA. To demonstrate the efficiency improvement of EGA over the traditional Monte Carlo genetic algorithm (MGA), the three examples (1–3) have been used as case studies. Table 2 presents the comparison results in terms of a fixed number of generations. The convergence paths of both EGA and MGA are presented separately in Figures 3–6.

As a result of the better uniformity property of HSS, the best solution found by EGA in the first generation should be better than in the MGA. Figures 3–5 show this trend. But Figure 6 shows that the best initial solution for MGA for example 3 with ND equal to 5 is 31, while the best initial solution found by EGA is 89.08. Because MCS is not as uniform as HSS, in some regions MCS has more sample points than HSS and, thus, MCS would have a better chance to find the best solution when it lies in these regions. At the same time, the use of HSS on the genetic operation produces more uniform operation on the population. Such unbiased operations maintain a better balance between keeping the fittest string pattern and diversity. In all the case studies shown in Figures 3–6, the fast convergence of EGA is observed while MGA is trapped in a local optimum before reaching maturity. All the above observations prove that GA benefits from the uniformity property of

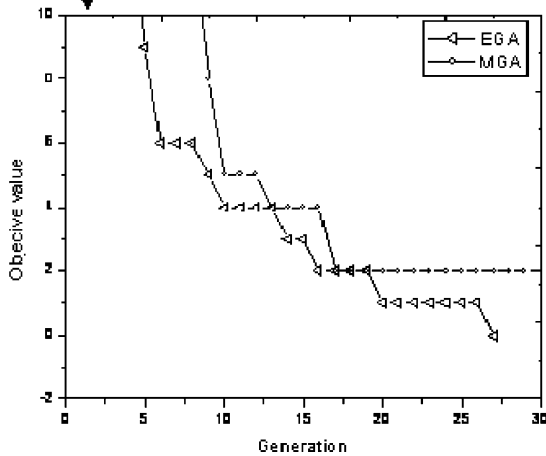
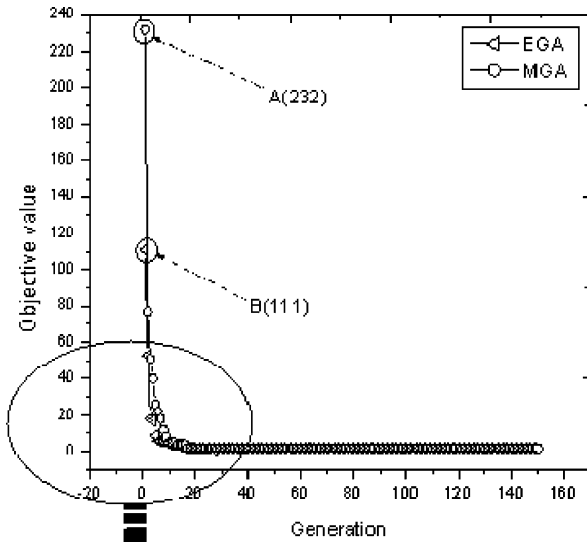


Figure 3. Convergence path of EGA and MGA for example 1 with ND = 10. Point A is the best initial solution in the first generation of MGA, while point B is the best initial solution found in the first generation of EGA. The total generation is 150.

HSS, by producing more diverse individuals and operations. Further, the figures show that when the dimension of the problem size increases from ND = 10 to ND = 20, the difference in performance between EGA and MGA is magnified.

4.2. Stochastic Genetic Algorithm (SGA). 4.2.1. Overview of Optimization under Uncertainty and SGA.

Optimization problems involving uncertainties in the data or model are commonly cited as stochastic programming problems and are divided into categories such as “wait and see”, “here and now”, and “chance constrained optimization”.²⁰ While formulating the optimization problems under uncertainty, the objective function and constraints are expressed in terms of probabilistic representations (e.g., expected value, variance, fractiles, or most likely values).

$$\begin{aligned}
 \min z &= P_1[f(x, \zeta)] \\
 \text{s.t. } P_2[h(x, \zeta)] &= 0 \\
 P_3[g(x, \zeta)] &\leq 0 \\
 x &\in X, \quad \zeta \in \Xi
 \end{aligned} \quad (4)$$

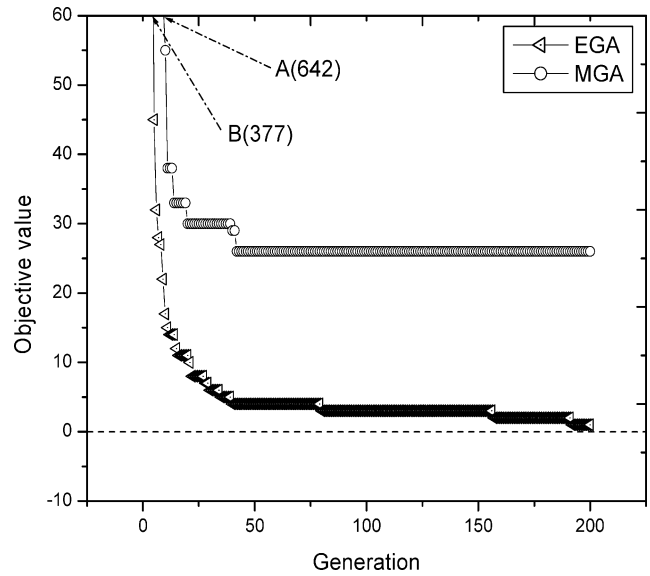


Figure 4. Convergence path of EGA and MGA for example 1 with ND = 20. Point A is the best solution in the first generation of MGA, while point B is the best solution found in the first generation of EGA. The total generation is 200.

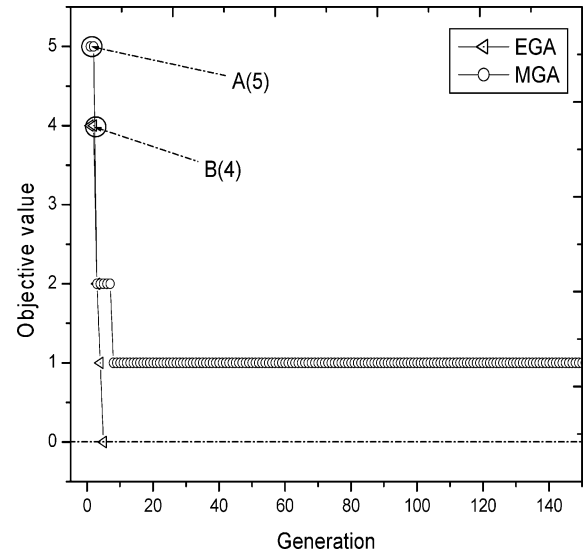


Figure 5. Convergence path of EGA and MGA for example 2. Point A is the best solution in the first generation of MGA, while point B is the best solution found in the first generation of EGA. The total generation is 200.

Here x is a vector of decision variables of domain X , and ζ is a vector of uncertain parameters of domain Ξ . The objective function, equality, and inequality constraints are defined by a set of probability functions P_1 , P_2 , and P_3 . The probability function P_i represents a cumulative distribution function such as the expected value, mode, variance, or fractiles. If P_i is the expected value, the above optimization problem becomes

$$\min z = E_{\zeta}[f(x, \zeta)] \quad (5)$$

where E_{ζ} is the mathematical expectation with respect to ζ . In this case, the main difficulty of stochastic programming stems from evaluating the uncertain functions and their expected values. A generalized method to propagate the uncertainties employs a sampling technique. Once the sampling method is determined, it propagates N_{sam} samples for random param-

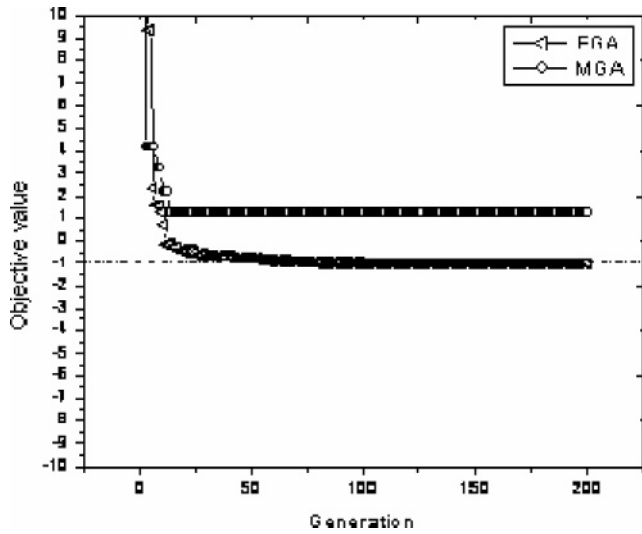


Figure 6. Convergence path of EGA and MGA for example 3 with ND = 5. Point A is the best solution in the first generation of MGA, while point B is the best solution found in the first generation of EGA. The total generation is 200.

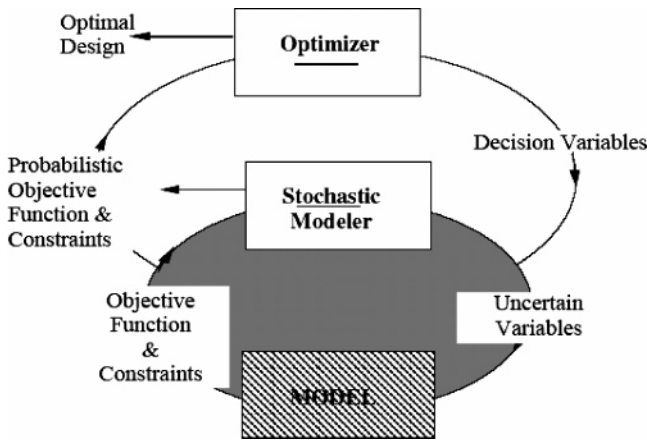


Figure 7. Framework of stochastic optimization.

eter ζ and optimizes the following approximated problem:

$$\min z = \frac{1}{N_{\text{samp}}} \sum_{j=1}^{N_{\text{samp}}} f(x, \zeta^j) \quad (6)$$

Figure 7 shows a generalized framework for stochastic optimization consisting of two loops; the inner sampling loop propagates uncertainties, and the outer optimization loop optimizes the probabilistic objective function. SGA is used in the outer loop to optimize a probabilistic objective function, which in our case is the expected value. The objective function includes the expected value of the objective function and a penalty term with respect to sample errors. The corresponding penalized objective function is as follows:

$$\min z = E_f f(x, \zeta) + b(t)\epsilon \quad (7)$$

where $b(t)$ is a weighting function and ϵ is the error bandwidth (confidence interval) of the sampling method. In the SGA, the optimizer obtains not only the decision variables but also the number of samples required for the stochastic model. The weighting function can be expressed as a function of generation. At the beginning of the search, accuracy is not essential; thus, fewer samples are needed, increasing the computation ef-

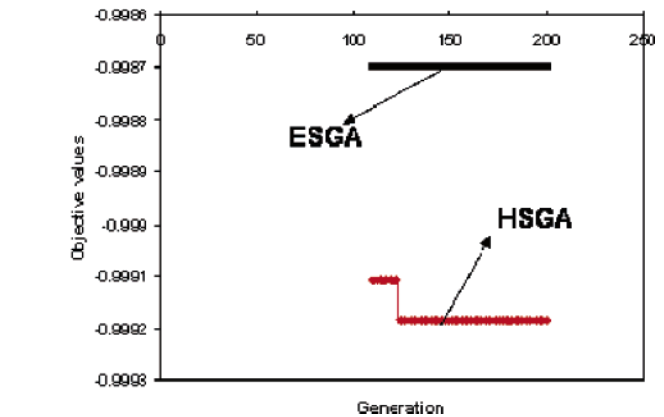
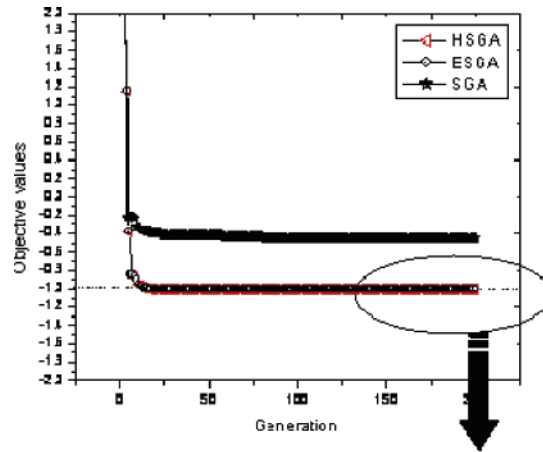


Figure 8. Convergence path of SGA, ESGA, and HSGA for the stochastic example or eq 11 with ND = 5.

iciency, while with the evolution exploitation becomes dominant. In this case, more samples are needed to ensure the accuracy of the results. From the above analysis, an exponential weighting function can be derived:

$$b(t) = b_0/k^t \quad (8)$$

where b_0 is a small constant (e.g., 0.001), k is a constant (e.g., 0.92) and t is the generation number. If the generation size is considerable, to ensure the penalty term does not exceed 5% of the real objective function t should be divided by a constant value like 100. The error bandwidth can be estimated from classical statistical methods, which leads to the following formula:

$$\epsilon \propto \frac{1}{(N_{\text{samp}})^\alpha} \quad (9)$$

where α is sampling-method-related constant. The corresponding α value for a crude Monte Carlo method is 0.5. The SGA algorithm is summarized as follows:

At $t = 0$, (1) generate initial population $P(t)$ and (2) select the number of samples N_{samp} by a random move. If $\text{rand}(0, 1) \leq 0.5$, then

$$N_{\text{samp}} = N_{\text{samp}} + 10 \text{ rand}(0, 1)$$

else

$$N_{\text{samp}} = N_{\text{samp}} - 10 \text{ rand}(0, 1)$$

(3) Evaluate $P(t)$ with penalized objective function 7.

When the termination condition is not satisfied, (1) update N_{samp} , (2) recombine $P(t)$ to generate $C(t)$, (3) evaluate $C(t)$ with penalized objective function (7), (4) select $P(t+1)$ from $P(t)$ and $C(t)$, and (5) set $P(t) = P(t+1)$.

4.2.2. ESGA. The inner sampling loop is important when trying to optimize the objective function 6. In this inner loop, a sampling method like MCS or LHS is used for the uncertain parameters. However, the required number of samples to approximate the "true" mean or variance is large, which would be computationally expensive, and this necessitates the use of an efficient sampling method. HSS, which shows both better homogeneities over multivariate parameter space and which uses a smaller number of samples for convergence, is an ideal substitute in the sampling loop. ESGA uses the same strategy as in EGA, in which HSS is used to produce initial populations and improve the uniformity of selection, crossover, and mutation. In addition, the HSS method is also used for uncertainty analysis in the stochastic model. Thus, in ESGA, HSS is used both in the inner sampling loop and in the outer optimization loop.

4.2.3. HSGA. The error bandwidth used in SGA and ESGA is derived from the estimation of the bounds of MCS using classical statistical methods. But this method overestimates either the confidence intervals or the bounds²¹ for HSS. Thus, a new error bandwidth for HSS needs to be characterized to get more efficient HSGA. A strategy based on the concept of fractal geometry^{1,22} to quantify the error bandwidth has been developed. The new α value for the HSS method is -1.4 , so the new HSS-specific error bandwidth is given by

$$\epsilon_{\text{HSS}} \propto \frac{1}{(N_{\text{samp}})^{1.4}} \quad (10)$$

With the incorporation of this new error bandwidth in the penalty term, the development of the HSGA is complete.

4.2.4. Performance of SGA, ESGA, and HSGA. Example 3 is modified by adding uncertain parameters, which leads to a stochastic MINLP,

$$f(x, y, \zeta) = \sum_i^{\text{ND}} \left(\zeta_i x_i - \frac{i}{\text{ND}} \right)^2 + \sum_i^{\text{ND}} \zeta_i y_i^2 - \prod_i^{\text{ND}} \cos(4\pi \zeta_i y_i) \quad (11)$$

Figure 8 shows the convergence path of SGA, ESGA, and HSGA, in which HSGA achieves the best performance and SGA the worst. The efficiency improvement of ESGA and HSGA over SGA is due to both the improved uniformity and the faster convergence properties of HSS. HSGA outperforms ESGA because of the reduced error bandwidth.

5. Conclusion

In this paper, the newly developed HSS technique has been applied to GA to improve the performance for both deterministic and stochastic optimization problems. EGA has been developed for solving deterministic optimization problems by capitalizing on the better uniformity property of the HSS technique in population initialization and genetic operation. The effect of seed on EGA was tested, which resulted in the use of dynamic seed to increase population diversity and to

decrease the dependence of performance on the random seed. In developing ESGA, both faster convergence and the uniformity properties of the HSS technique have been exploited. In HSGA, the HSS-specific error bandwidth has been applied to the penalty term of the probabilistic objective function. HSGA has been proved to converge faster than SGA and ESGA.

Literature Cited

- (1) Diwekar U. A novel sampling approach to combinatorial optimization under uncertainty, *Comput. Optim. Appl.* **2003**, *24*, 335–371.
- (2) Holland, J. H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, 1975.
- (3) Koza, J. R. Genetic programming: on the programming of computers by means of natural selection. *Complex Adaptive Systems*; MIT Press: Cambridge, MA, 1992.
- (4) Fogel, G. B.; Owens, A. J.; Walsh, M. J. *Artificial Intelligence Through Simulated Evolution*; Wiley: New York, 1966.
- (5) Rechenberg, I. *Evolutionstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*; Frommann-Holzboog: Stuttgart, Germany, 1973.
- (6) Schwefel, H. *Evolution and Optimum Seeking*; Wiley: New York, 1995.
- (7) Tarafder, A.; Rangaiah, G. P.; Ray, A. K. Multiobjective optimization of an industrial styrene monomer manufacturing process. *Chem. Eng. Sci.* **2005**, *60*, 347–363.
- (8) Michalewicz, Z. *Genetic Algorithm + Data structure = Evolution Programs*, 3rd ed.; Springer-Verlag: New York, 1996.
- (9) Deb, K.; Agrawal, R. B. Simulated binary crossover for continuous search space. *Complex Syst.* **1995**, *9*, 115–148.
- (10) Herrera, F.; Lozano, M. Adaptation of genetic algorithm parameters based on fuzzy logic controllers. In *Genetic Algorithms and Soft Computing*; Herrera, F., Verdegay, J., Eds.; Physica-Verlag: Heidelberg, Germany, 1996; pp 95–125.
- (11) Özdamar, L. A genetic algorithm approach to a general category project scheduling problem. *IEEE Trans, Syst., Man, Cybern., Part C: Appl. Rev.* **1999**, *29*, 44–59.
- (12) Diwekar, U. M.; Kalagnanam, J. R. Efficiency sampling technique for optimization under uncertainty. *AIChE J.* **1997**, *43*, 440–447.
- (13) Iman, R. L.; Conover, W. J. Small-sample sensitivity analysis technique for computer models, with an application to risk assessment. *Commun. Stat., Part A: Theory and Methods* **1982**, *17*, 1749–1842.
- (14) Gen, M.; Cheng, R. *Genetic Algorithms & Engineering Optimization*; Wiley-Interscience: New York, 2000.
- (15) Dantzig, G.; Glynn, P. Parallel processors for planning under uncertainty. *Ann. Operat. Res.* **1990**, *22*, 1–21.
- (16) Kim, K.-J.; Diwekar, U. M. Efficient Combinatorial Optimization under Uncertainty. 1. Algorithmic Development. *Ind. Eng. Chem. Res.* **2002**, *41*, 1276–1284.
- (17) Salazar, R.; Toral, R. Simulated annealing using hybrid Monte Carlo. *J. Stat. Phys.* **1997**, *89*, 1047–1060.
- (18) Panton, L. A.; Diwekar, U. M. Synthesizing optimal design configurations for a brayton cycle power plant. *Comput. Chem. Eng.* **1994**, *18*, 365–381.
- (19) Bayer, H. G. Toward a theory of evolution strategies: Self-adaptation. *Evol. Comput.* **1996**, *3*, 311–347.
- (20) Diwekar, U. M. Introduction to applied optimization. *Applied optimization*; Kulwer Academic Publishers: Norwell, MA, 2003, Vol. 80.
- (21) Chaudhuri, P.; Diwekar, U. M. Synthesis approach to the determination of optimal waste blends under uncertainty. *AIChE J.* **1999**, *45*, 1671–1687.
- (22) Kim, K. J.; Diwekar, U. M. Characterizing Sampling Error for Optimization Under Uncertainty: A Fractal Geometry Approach. Presented at INFORMS Annual Meeting, Atlanta, GA, 2003.

Received for review September 9, 2004
Revised manuscript received June 8, 2005
Accepted June 24, 2005