Improved genetic algorithms for deterministic optimization and optimization under uncertainty-Part I: Algorithms Development

Xu, W. and Urimila M. Diwekar*
Center for Uncertain Systems, Tools for Optimization and Management (CUSTOM)
Vishwamitra Research Institute, Westmont, IL 60559

*To whom correspondence should be addressed. E-mail: urmila@vri-custom.org Phone: +1 630 515 8772

Abstract

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 Hammersley sequence sampling 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.
Keywords: Hammersley Sequence Sampling, Efficient Genetic Algorithm, Stochastic Genetic algorithm, Hammersley Stochastic Genetic Algorithm.

1. Introduction

Many optimization problems, which include a large number of continuous or discrete design variables fall into the category of integer programming (IP) and mixed integer nonlinear programming (MINLP). Branch and bound (BB), generalized bender's decomposition (GBD), and outer-approximation (OA) (Diwekar, 2003) are generally used for solving IP 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 based on Darwin's natural selection and Mendel's genetics, provides an alternative to the mixed integer programming techniques like the BB, GBD and OA. The class of evolutionary algorithms includes genetic algorithms (GA) (Holland, 1975); genetic programming (GP) (Koza, 1992); evolutionary programming (EP) (Fogel et al. 1966); and evolutionary strategy (ES) (Rechenberg, 1973; Schwefel, 1995). Among all the evolutionary algorithms, GA is the most widely used. GA was first introduced by Holland (1975). It has been receiving increased attention due to a series of successful applications in different disciplines like biology, medicine and different branches of engineering (Tarafder et al., 2004)

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 like 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 like 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 (Michalewicz, 1996), simulated binary crossover operator (SBX) (Deb and Agrawal, 1995), parameter adaptation (Herrera and M. Lozano, 1996), and hybrid genetic algorithm (HGA) (Özdamar, 1999).

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) (Diwekar and Kalagnanam, 1997), 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) (Iman and Conover, 1982). Genetic algorithm especially benefits from these features, since 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 genetic algorithm

The Genetic Algorithm was first developed by Holland (Holland, 1975). In general, there are five components in it (Michalewicz, 1996).

- A genetic representation of solutions to the problem.
- A way to create an initial population of solutions.
- An evaluation function rating solutions in terms of fitness.
- Genetic operators that generate new individuals.
- Values for the parameters of genetic algorithms.

The general procedure of genetic algorithms can be summarized as follows:

```
At t = 0,
    • Generate initial population P(t).
    • Evaluate P(t).
While termination condition is not satisfied, do
    • Recombine P(t) to generate new individuals, i.e., children C(t).
    • Evaluate C(t).
    • Select P(t + 1) from P(t) and C(t).
    • 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 genetic algorithm. 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 (Gen and Cheng, 2000). The initial population *P(0)* can be chosen heuristically or randomly. If it is chosen randomly, a corresponding sampling technique like Monte Carlo sampling (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 hyper-planes already represented earlier in the population. By evaluating new solution strings, GA gathers further knowledge about these hyper-planes. Second, crossover introduces representatives of new hyper-planes into the population. If this new hyper-plane 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. Genetic algorithm parameters like population size, crossover ratio, and mutation ratio are key factors in the trade-off between exploitation and exploration (Holland, 1975).

3. Sampling method.

Monte Carlo Sampling, latin hypercube sampling (Iman and Conover, 1982), and importance sampling (Dantzig and Glynn, 1990) are widely used sampling techniques. Recently, an efficient sampling technique called Hammersley sequence sampling (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 population, showing better uniformity in the multi-dimensional uncertain surface compared to Monte Carlo, latin hypercube, and its variant, the median latin hypercube sampling (MLHS) techniques. The main reason for this is that the Hammesley 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 MCS technique and HSS technique are shown in Fig. 1. It shows that samples generated by HSS technique achieves better uniformity and results in faster convergence to the "true" mean, variance, or fractiles (Diwekar and Kalagnanam, 1997).

4. Hierarchical improvements of Genetic Algorithm.

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 (Kim and Diwekar, 2002).

4.1. Efficient genetic algorithm.

4.1.1. Overview.

Population diversity plays an important role in the performance of genetic algorithm. 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 HSS technique in selection, crossover and mutation rather than random probability functions in these operations results in additional improvements. Generally, these random probabilities generated by pseudo-random number generators like crude Monte Carlo are uniformly distributed. Since 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 towards a particular region of solution space or chromosome, which would trap the genetic algorithm in a local optima. 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. Efficient genetic algorithm (EGA) is developed by implementing these new features. The algorithm is summarized as follows:

```
At t = 0,
    • Use HSS method to generate initial population P(t), keeping the k-dimensional uniformity property intact.
    • Evaluate P(t).
While termination condition is not satisfied,
    • Recombine P(t) to generate C(t). Use HSS method to generate random moves in selection, crossover and mutation steps, keeping the k-dimensional uniformity property intact.
    • Evaluate C(t).
    • Select P(t + 1) from P(t) and C(t).
    • 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 : $$f(y)=\sum_{i=1}^{ND} y_i^2 , \tag{1}$$

Example 2 : $$f(y)=\sum_{i=1}^{y_1}\left\{(y_1-3)^2+(y_2(i)-3)^2+(y_3(i)-3)^2\right\}, \tag{2}$$

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

where x denotes a vector of continuous variables, y denotes a vector of discrete variables and ND is dimension of the examples. Example 1 is a multi-dimensional parabolic function (Salazar and Toral, 1997) that has one global optimum at zero for all decision variables equals to zero. The second example, a pure combinatorial problem (Painton and Diwekar, 1994) that has one global minimum zero when all y_1, $y_2(i)$, $y_3(i)$ 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 efficient genetic algorithm.

Theoretically, the random seed which is used to propagate samples, should have no impact on the performance of HSS. However, the existence of limitation on sample size makes it impossible for the samples to cover the whole search space. This limitation directly induces a non-overlapping distribution of different set of samples on the k-dimensional space, though each set of sample is uniformly distributed. From Fig. 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 set of samples used. 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 like population size, crossover rate etc. that need to be tuned to reach the best performance. The existence of additional parameters would make genetic algorithm less flexible. To increase the diversity of new populations and decrease the effect of different seeds, the strategy of parameter adaptation (Holland, 1975; Beyer, 1996) is used for random seed adaptation. Since there is no relationship between performance and random seed value, changing random seed schematically 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. The efficiency improvement of EGA.

To demonstrate the efficiency improvement of EGA over Monte Carlo genetic algorithm (MGA), the three examples (1), (2), and (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 Fig. 3 to Fig. 6.

Due to the better uniformity property of HSS, the best solution found by EGA in the first generation should be better than MGA. Fig. 3, 4, 5 show this trend. But Fig. 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. Since MCS is not as uniform as HSS, in some regions MCS has more sample points than HSS, 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 Fig. 3 to Fig. 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 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.

4.2.1. Overview of optimization under uncertainty and stochastic genetic algorithm (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" (Diwekar, 2003). 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).

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

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:

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

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, then it propagates N_{samp} samples for random parameter ζ and optimizes the following approximated problem:

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

Figure 7 shows a general framework of stochastic optimization. It has two loops, inner sampling loop to propagate uncertainties, and outer optimization loop to optimize the probabilistic objective function. Stochastic genetic algorithm (SGA) is used to 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:

$$\min \quad z = E_\zeta f(x,\zeta) + b(t)\varepsilon \tag{7}$$

where $b(t)$ is a weighting function, and ε is the error bandwidth (confidence interval) of sampling method. In the stochastic genetic algorithm, 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 for the sake of computation efficiency, 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:

$$b(t) = \frac{b_0}{k^t} \tag{8}$$

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:

$$\varepsilon \propto \frac{1}{\left(N_{samp}\right)^{\alpha}} \tag{9}$$

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,
    • Generate initial population P(t).
    • Select the number of samples $N_{samp}$ by a random move. If $rand(0,1) \le 0.5$, then
        $N_{samp} = N_{samp} + 10 * rand(0,1)$
      else
        $N_{samp} = N_{samp} - 10 * rand(0,1)$
    • Evaluate P(t) with penalized objective function (7).
While termination condition is not satisfied, do
    • Update $N_{samp}$.
    • Recombine P(t) to generate C(t).
    • Evaluate C(t) with penalized objective function (7).
    • Select P(t + 1) from P(t) and C(t).
    • Set P(t) = P(t + 1).
```

4.2.2. Efficient stochastic genetic algorithm (ESGA).

The inner sampling loop is important when trying to optimize the objective function (6). In this inner loop, a sampling method like Monte Carlo sampling (MCS) or Latin hypercube sampling (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. Hammersley sequence sampling (HSS), which shows both better homogeneities over multivariate parameter space and which uses less number of samples for convergence, is an ideal substitute in the sampling loop. Efficient stochastic genetic algorithm (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 outer optimization loop.

4.2.3. Hammersley Stochastic Genetic Algorithm (HSGA).

The error bandwidth used in SGA and ESGA is derived from the estimation of the bounds of Monte Carlo sampling using classical statistical methods. But this method overestimates either the confidence intervals or bounds (Chaudhuri and Diwekar, 1999) 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 (Kim and Diwekar, 2004) to quantify the error bandwidth has been developed. The new α value for HSS method is -1.4, so the new HSS-specific error bandwidth is given by:

$$\varepsilon_{HSS} \propto \frac{1}{\left(N_{samp}\right)^{1.4}} \tag{10}$$

With the incorporation of this new error bandwidth in the penalty term, the development of the Hammsersley stochastic genetic algorithm (HSGA) is complete.

4.2.4. The performance of SGA, ESGA, and HSGA.

Example (3) is modified by adding uncertain factors, which leads to a stochastic MINLP,

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

Fig. 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 faster convergence properties of HSS. HSGA outperforms ESGA because of the reduced error bandwidth. The usage of the HSS bandwidth reduces the possibilities of distraction from the optimal objective value, thus make HSGA reach the real objective value faster.

5. Conclusion.

In this paper, the newly developed HSS technique has been applied to genetic algorithm to improve the performance for both deterministic and stochastic optimization problems. Efficient genetic algorithm (EGA) has been developed for solving deterministic optimization problems by capitalizing on the better uniformity property of 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 efficient stochastic genetic algorithm (ESGA), both faster convergence and the uniformity properties of HSS technique have been exploited. In Hammersley stochastic genetic algorithm (HSGA), the HSS-specific error bandwidth has ben applied to the penalty term of the probabilistic objective function. HSGA has been proved to converge faster than stochastic genetic algorithm (SGA) and Efficient stochastic genetic algorithm (ESGA).

References

Bayer, H. G., 1996. Toward a theory of evolution strategies: Self-adaptation. Evolutionary Computation, 3, 311-347.

Chaudhuri, P., Diwekar, U., M., 1999. Synthesis approach to the determination of optimal waste blends under uncertainty. AIChE J., 45, 1671-1687.

Dantzig, G., Glynn, P., 1990. Parallel processors for planning under uncertainty. Annals of Operations Research., 22, 1-21.

Deb, K., Agrawal, R., B. 1995. Simulated binary crossover for continuous search space, Complex Systems, 9, 115-148.

Diwekar, U., M., Kalagnanam, J., R. 1997. Efficiency sampling technique for optimization under uncertainty. AIChE J., 43, 440-447.

Diwekar, U. M., 2003. Introduction to applied optimization. Applied optimization, 80, Kulwer Academic Publishers, Massachusetts.

Fogel, G. B., Owens, A.J., Walsh, M. J., 1966. Artificial Intelligence Through Simulated Evolution. Willey, New York.

Gen, M., Cheng, R., 2000. Genetic algorithms & Engineering optimization. Wiley-Interscience publication, USA.

Herrera, F. and M. Lozano. 1996. Adaptation of genetic algorithm parameters based on fuzzy logic controllers, in Herrera, F. and J. Verdegay, editors, Genetic Algorithms and Soft Computing, 95-125.

Holland, J. H., 1975. Adaptation in Natural and Artifical System, University of Michigan Press, Ann Arbor, MI.

Iman, R. L., Conover, W. J. 1982. Small-sample sensitivity analysis technique for computer models, with an application to risk assessment. Communications in Statistics – Part A, Theory and Methods, 17, 1749-1842.

Kim, K.-J., Diwekar, U. M. 2002 Efficient Combinatorial Optimization under Uncertainty. 1. Algorithmic Development. Ind. Eng. Chem. Res., 41, 1276-1284.

Kim K. J., and Diwekar, U. M. 2004. Characterizing Sampling Error for Optimization Under Uncertainty: A Fractal Geometry Approach. submitted to Operations Research.

Koza, J. R., 1992. Genetic programming: on the programming of computers by means of natural selection. In: Complex Adaptive Systems. MIT Press, Cambridge, MA.

Michalewicz, Z., 1996. Genetic Algorithm + Data structure = Evolution Programs, 3rd edition, Springer-Verlag, New York.

Özdamar, L. 1999. A genetic algorithm approach to a general category project scheduling problem. IEEE Trans, Syst., Man, Cybern.-Part C: Application and reviews, 29, 44-59.

Painton, L. A., Diwekar, U., M., 1994. Synthesizing optimal design configurations for a brayton cycle power plant. Computers and Chemical Engineering, 18, 365-381.

Rechenberg, I., 1973. Evolutionstrategie: Optimieriung technischer systeme nach prinzipien der biologischen evolution, Frommann-Holzboog, Stuttgart, Germany.

Salazar, R., Toral, R., 1997. Simulated annealing using hybrid Monte Carlo. Journal of Statistical Physics, 89, 1047-1060.

Schwefel, H., 1995. Evolution and Optimum Seeking, Wiley, New York.

Tarafder, A., Rangaiah, G.P. and Ray, Ajay K.,2005. Multiobjective optimization of an industrial styrene monomer manufacturing process. Chemical Engineering Science, 60, 347-363.

List of Figures:

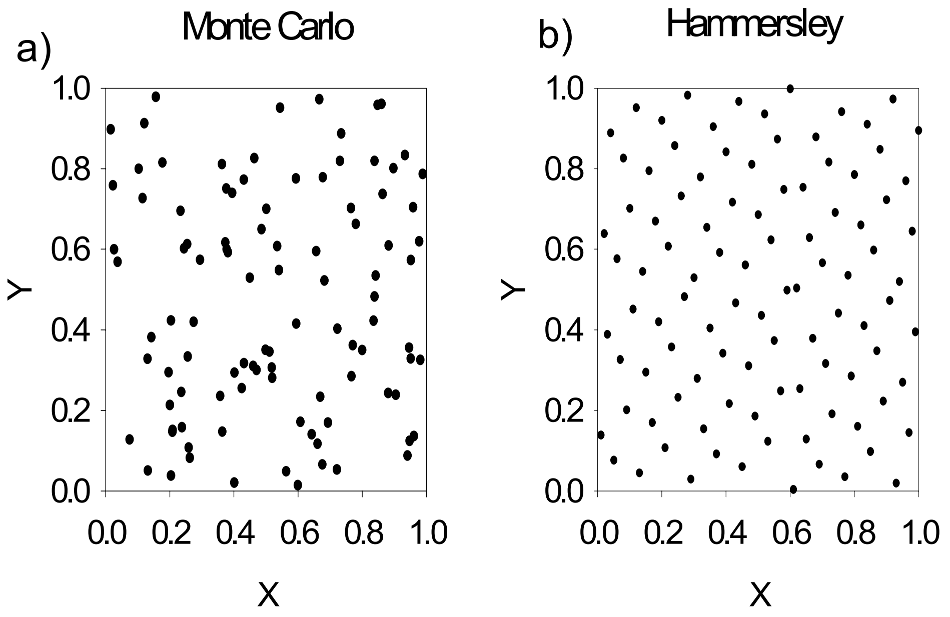

Fig. 1: Comparison between (a) MCS sampling and (b) HSS sampling technique.

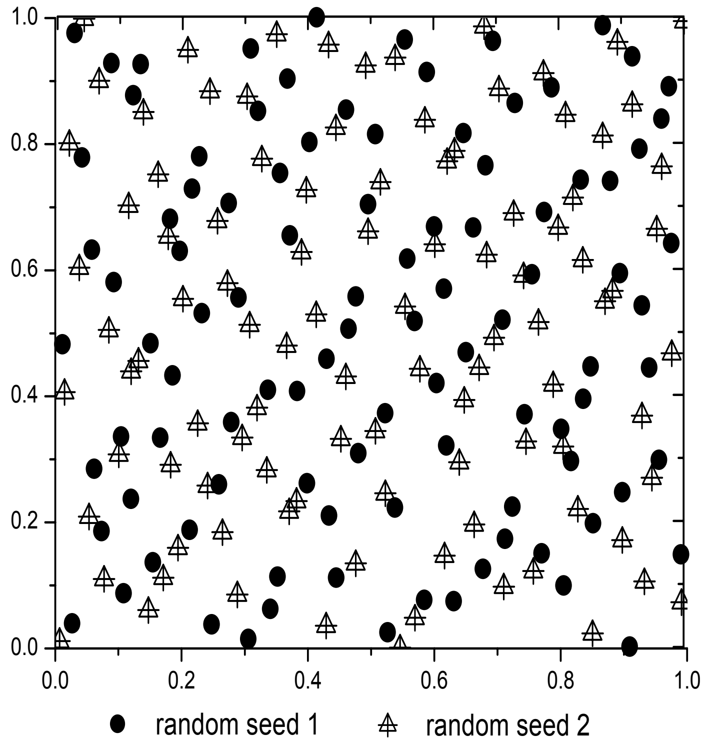

Fig. 2: The effect of different seed on the HSS generated samples.

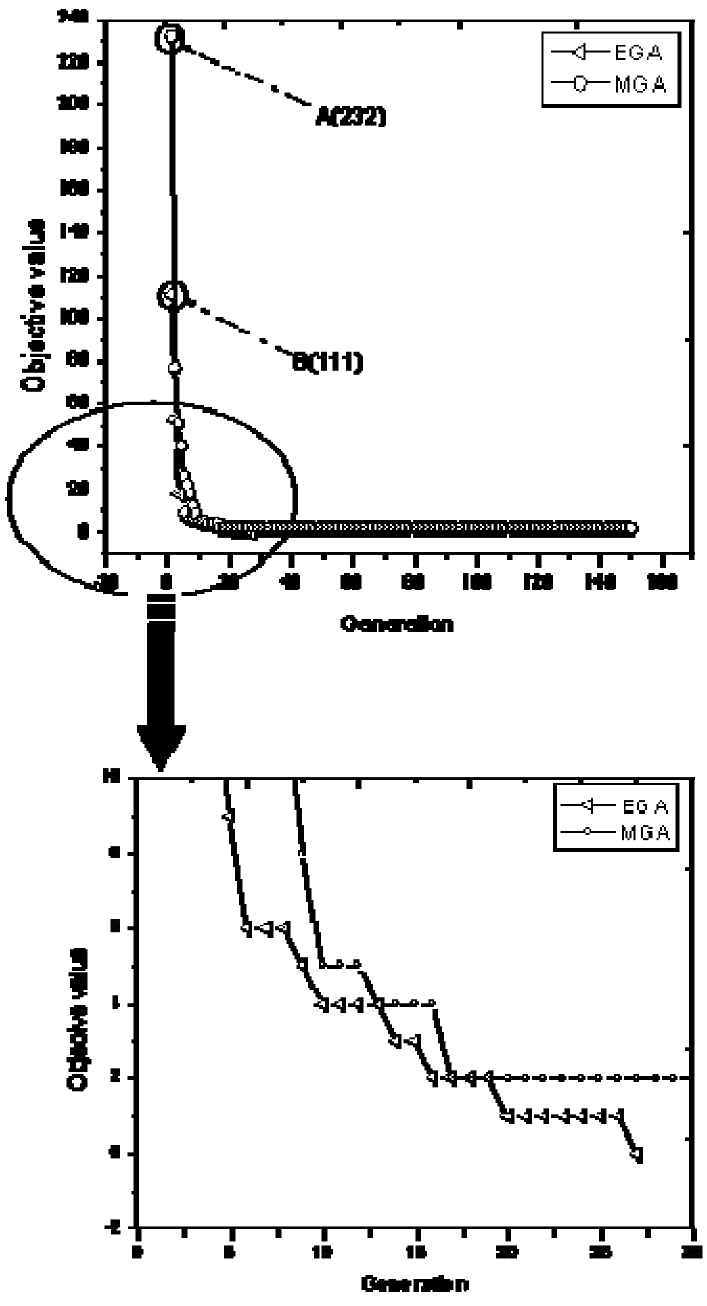

Fig. 3: The convergence path of EGA and MGA for Example 1 with ND = 10 of a single run. A is the best initial solution in the first generation of MGA, while B is the best initial solution found in the first generation of EGA. The total number of generations is 150.

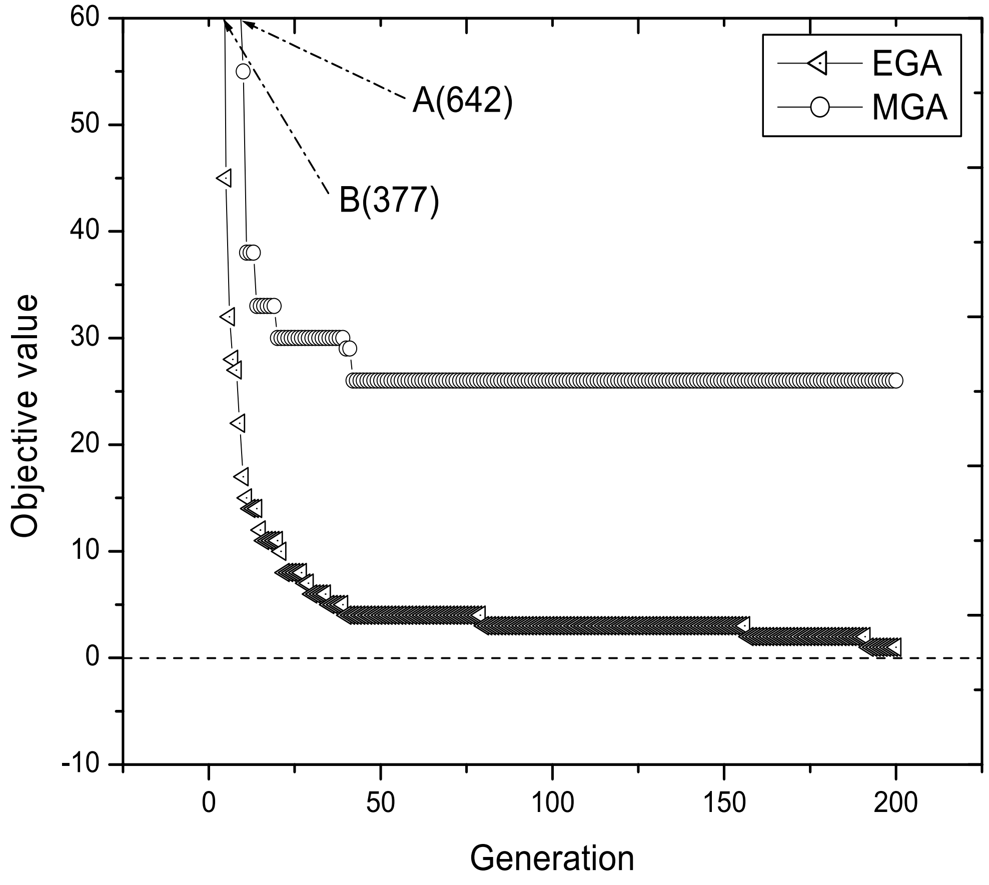

Fig. 4: The convergence path of EGA and MGA for Example 1 with ND = 20 of a single run. A is the best solution in the first generation of MGA, while B is the best solution found in the first generation of EGA. The total number of generation is 200.

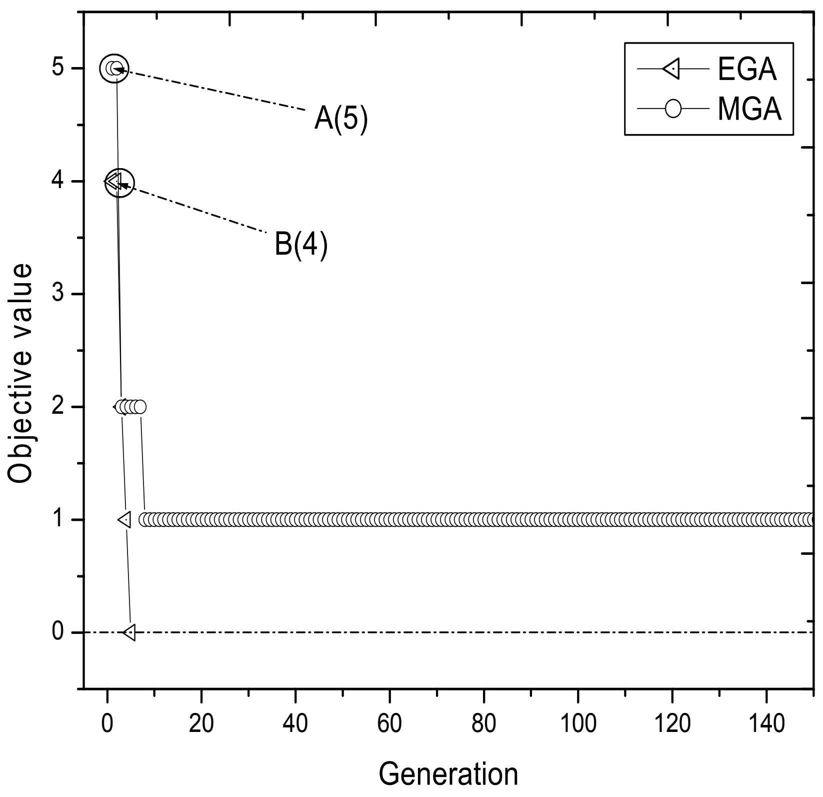

Fig. 5: The convergence path of EGA and MGA for Example 2 of a single run. A is the best solution in the first generation of MGA, while B is the best solution found in the first generation of EGA. The total number of generation is 200.

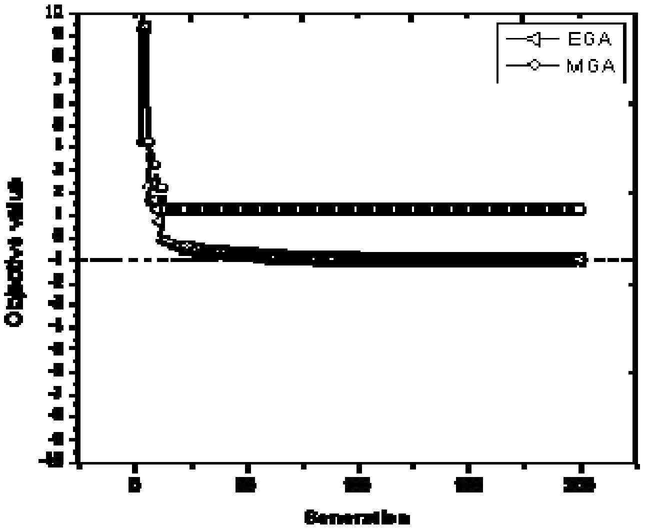

Fig. 6: The convergence path of EGA and MGA for Example 3 with ND=5 of a single run. A is the best solution in the first generation of MGA, while B is the best solution found in the first generation of EGA. The total number of generation is 200.

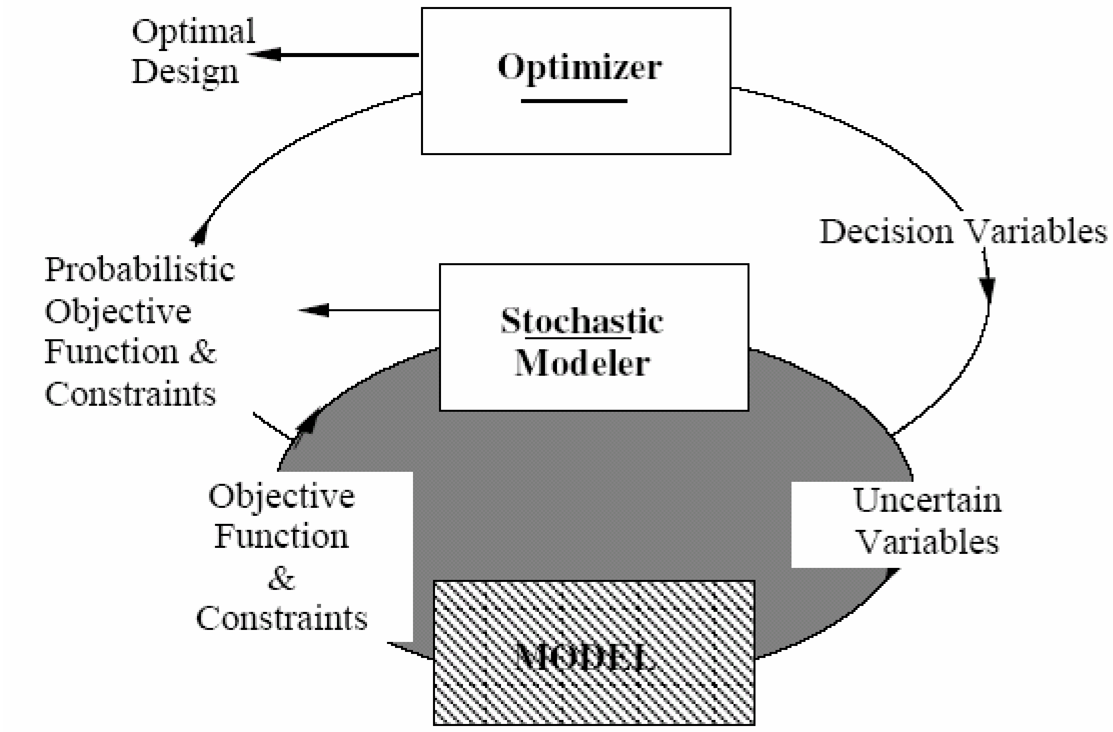

Fig. 7. The framework of stochastic optimization.

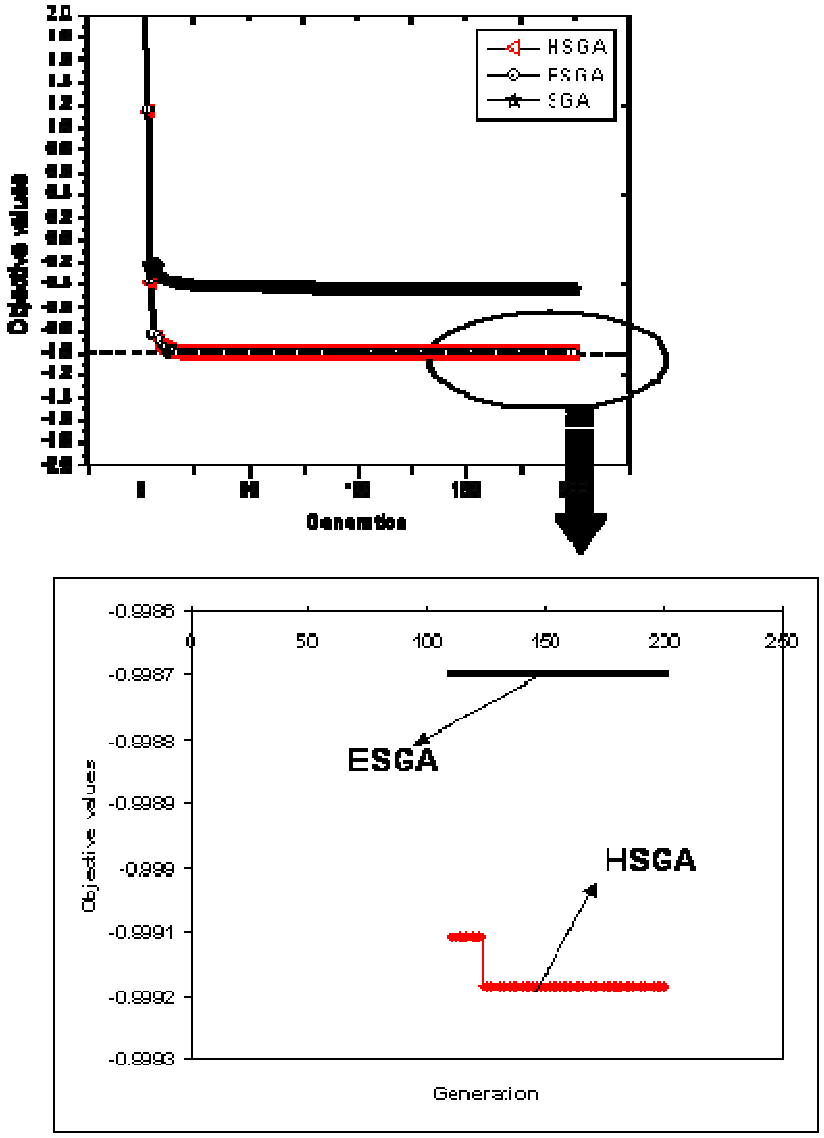

Fig. 8: The convergence path of SGA, ESGA, and HSGA for stochastic example of equation (11) with ND = 5.

List of Tables:

Table 1: The effect of seed on the performance of efficient genetic algorithm for example (3) with ND = 5.

Random seed	425001	888868	25689	256	Dynamic seed
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), (2) and (3). The results of EGA are the average of five runs with dynamic seed value.

			Optimal value/Generation		
	ND	Total number of generations	MGA	EGA	Theoretical optima
Example 1	10	150	2/150	0/40.6	0
	20	200	26 / 200	0 / 195.2	0
Example 2		200	1 / 200	0 / 16.4	0
Example 3	5	200	-0.9987 / 200	-0.9991 / 88.6	-1