Journal of the Applied Mathematics, Statistics and Informatics (JAMSI), **6** (2010), No. 2

A NEW EFFICIENT STOCHASTIC ALGORITHM FOR OPTIMIZATION PROBLEMS AND ITS IMPROVEMENT

BEHROUZ FATHI VAJARGAH AND MOHADESEH KANAFCHIAN

Abstract

This paper presents two stochastic algorithms against finite difference algorithm for optimizing a desired function based on Halton random numbers sequences and compares their results.

Mathematics Subject Classification 2000: 78M50

Key Words: Optimization, FD algorithm, RD algorithm, RD-Halton, Gradient approximation, Quasi-Random generator

1. INTRODUCTION

Optimization is the science of selecting the best of many possible decisions in a real-life environment. In other sciences such as engineering, physical and social sciences, we encounter problems involving the optimization of some mathematical objective function (e.g., as in optimal control, system design and planning, model fitting, and performance evaluation from system test data).

Physical systems tend to a state of minimum energy. The molecules at an isolated chemical system react with each other until the total potential energy of their electrons is minimized. Rays of light follow paths that minimize their travel time.

All the examples mentioned above, can be expressed as an optimization problem. Most real-world problems are complex and have many variables. So, to solve the optimization problem, it is very important to use a method that does not require information about the objective function, even for very complex functions.

The solution of optimization problem corresponds to a vector of parameters at which the gradient of the objective function with respect to parameters being optimized is zero. So some deterministic algorithms such as gradient algorithm require information about the gradient of objective function. If gradient information is conveniently available, it is generally to one's advantage to use this information in the optimization process. But the main problem occurs when the gradient information is not readily available or objective function is too complicated and it is difficult or impossible to directly obtain the gradient of objective function. In this case, the problem can not be solved with the gradient algorithm because the gradient algorithm relies on measurements of the gradient of the objective function.

2. GRADIENT ALGORITHM

Consider the following deterministic minimization problem:

$$\min_{x \in D \subset R^n} g(x) = g(x^*) = g^* \tag{1}$$

where $g(x)$ is a real-valued bounded function which is defined on a closed bounded domain $D \subset R^n$. We assume that objective function $g(x)$ is a function of x where $x \in D \subset R^n$, $n \geq 1$. It is assumed that g achieves its minimum value at a unique point x^*.

In this paper, we consider our attention to minimization problem because maximizing $g(x)$ is equivalent to minimizing $-g(x)$.

Problem (1) is equivalent to finding the minimum point x^* such that

$$\nabla g(x^*) = \left. \frac{\partial g}{\partial x} \right|_{x=x^*} = 0 \quad .$$

Therefore, most deterministic algorithms for solving minimization problem require the gradient information. One of these algorithms is gradient algorithm.

In path search methods, we choose an initial solution vector as the starting point, from which the solution moves certain step size forwarding in a direction so as to improve the objective function
value. Usually, the direction is determined by the gradient of the function.

Gradient algorithm is one of the most important methods in the family of path search algorithms. It traces a sequence of points in the search space that converges to the point with zero gradient of the objective function.

Suppose that $g(x)$ is a differentiable function. For simplicity assume that the set $D = R^n$. According to the gradient algorithm, we approximate the point x^* step by step. If on the k^{th} iteration we have reached point $\hat{x}_k$, then the next point $\hat{x}_{k+1}$ is chosen as

$$\hat{x}_{k+1} = \hat{x}_k - \alpha_k \nabla g(\hat{x}_k) \tag{2}$$

where $\nabla g(x) = \left[\frac{\partial g(x)}{\partial x_1}, \ldots, \frac{\partial g(x)}{\partial x_n} \right]^T$ is the gradient of $g(x)$ and $\frac{\partial g(x)}{\partial x_i}$, $i = 1, 2, \ldots, n$ are the partial derivatives and $\alpha_k > 0$ is the step size of iteration k.

Gradient algorithm relies on values of the gradient of objective function with respect to x. In the other word, for solving problem (1), gradient algorithm is based on the

simple principle that from a given starting point the best direction to go is the one that produces the largest (maximum) reduction in the objective function value.

α_k play a critical role in the algorithm, often determining whether the algorithm converges or diverges. Steps that are too large or too small may prevent the algorithm from converging to x^*. Usually, α_k are constant. Let ε represents a small positive number, such that algorithm stops when $\left\| \nabla g(\hat{x}_k) \right\| < \varepsilon$.

The essential part of algorithm (2) is the gradient at each point $\hat{x}_k$. If one of the following cases occurs, we cannot calculate the gradient and the problem (1) can not be solved with gradient algorithm.

1) The gradient information is not readily available.
2) Objective function $g(x)$ is too complicated and it is difficult or impossible to directly obtain the gradient of $g(x)$.
3) $g(x)$ is not differentiable.
4) Analytic expression of $g(x)$ is not given explicitly (only the values of $g(x)$ can be observed at each point on a given domain.

Then we must approximate the gradient of objective function $g(x)$ and we use the gradient algorithm with the gradient approximation instead of the direct gradient.

3. GRADIENT-BASED APPROXIMATION ALGORITHM

All of these gradient –based approximation algorithms, which will be expressed, are based on only values of objective function. A main advantage of such algorithms is that they do not require the knowledge of objective function being minimized that is required in gradient algorithm. The general form of these algorithms for solving problem (1) is:

$$\hat{x}_{k+1} = \hat{x}_k - \alpha_k \hat{\nabla} g(\hat{x}_k), \quad \alpha_k > 0 \tag{3}$$

where $\hat{\nabla} g(\hat{x}_k)$ is the approximation (or estimate) of the gradient $\nabla g(\hat{x}_k)$.

Now, we express the two general forms of gradient-based on approximation algorithms and we discuss their advantages and disadvantages.

3.1. Finite difference gradient algorithm

In finite difference (FD) approximation of gradient algorithm, each component of x_k is perturbed one-at-a-time and corresponding values g are obtained; each component of gradient vector is calculated by differencing the values of objective function $g(x)$ and then dividing by the corresponding difference interval [3]. Typically, the i^{th} component

of $\hat{\nabla} g(\hat{x}_k)$ (i =1, 2,…, n) for a two-side FD approximation is given by

$$\hat{\nabla} g_i(x_k) = \frac{g(\hat{x}_k + \beta_k e_i) - g(\hat{x}_k - \beta_k e_i)}{2\beta_k}, \quad \beta_k > 0 \tag{4}$$

where e_i denotes the vector with component i equals to 1 and all other components equal to 0. Algorithm (2) by using two-side FD approximation is called finite difference gradient algorithm.

For different k, the pairs $\{\alpha_k, \beta_k\}$ are gains (or gain sequences). Under appropriate conditions on α_k and β_k, the finite difference gradient algorithm converges to x^*.

3. 2. Convergence conditions

Let $g_i'(x)$, $g_i''(x)$, and $g_i'''(x)$ represent the first, second, and third derivatives of g with respect to the i^{th} component of x. Suppose that the $g_i'''(x)$ are continuous for all i and $x \in D \subset \Re^n$. Then by a third-order form of Taylor's theorem, we have

$$g(\hat{x}_k + \beta_k e_i) = g(\hat{x}_k) + \beta_k g_i'(\hat{x}_k) + \frac{1}{2}\beta_k^2 g_i''(\hat{x}_k) + \frac{1}{6}\beta_k^3 g_i'''(\bar{x}_k^{(i+)}) \tag{5}$$

$$g(\hat{x}_k - \beta_k e_i) = g(\hat{x}_k) - \beta_k g_i'(\hat{x}_k) + \frac{1}{2}\beta_k^2 g_i''(\hat{x}_k) - \frac{1}{6}\beta_k^3 g_i'''(\bar{x}_k^{(i-)}) \tag{6}$$

where $\bar{x}_k^{(i+)}$ and $\bar{x}_k^{(i-)}$ denote points on the line segments between $\hat{x}_k$ and $\hat{x}_k \pm \beta_k e_i$. Therefore, by using (5) and (6) in (4), we have

$$\begin{aligned}
\hat{\nabla} g_i(\hat{x}_k) &= \frac{g(\hat{x}_k + \beta_k e_i) - g(\hat{x}_k - \beta_k e_i)}{2\beta_k} \\
&= g_i'(\hat{x}_k) + \frac{1}{6}\frac{\beta_k^3 g_i'''(\bar{x}_k^{(i+)}) - \beta_k^3 g_i'''(\bar{x}_k^{(i-)})}{2\beta_k} \\
&= g_i'(\hat{x}_k) + O(\beta_k^2)
\end{aligned}$$

Therefore

$$\hat{\nabla} g(\hat{x}_k) = \nabla g(\hat{x}_k) + O(\beta_k^2)$$

So, β_k are small positive numbers that usually get smaller as k gets larger.

Under following conditions on gain sequences, the finite difference gradient algorithm converges to x^* as $k \to \infty$.

$$\alpha_k > 0, \alpha_k \to 0, \ \beta_k \to 0, \ \beta_k > 0, \sum_{k=0}^{\infty} \alpha_k = \infty, \sum_{k=0}^{\infty} \alpha_k \beta_k < \infty, \sum_{k=0}^{\infty} \frac{\alpha_k^2}{\beta_k^2} < \infty$$

The choice of these gain sequences is critical to the performance of the algorithm. Common forms for the sequences are $\alpha_k = \frac{a}{(k+1+A)^{\alpha}}$ and $\beta_k = \frac{c}{(k+1)^{\gamma}}$ where the coefficients a, c, α and γ are strictly positive and $A \geq 0$. The user must choose these coefficients, a process usually based on a combination of the theoretical restrictions above, trial-and-error numerical experimentation, and basic problem knowledge. In some cases, it is possible to partially automate the selection of the gains [2].
Note that the number of objective function measurements needed per iteration of FD approximation grows with n. For example, let $n = 2$. From (4), we have

$$\hat{\nabla} g(\hat{x}_k) = \left[\hat{\nabla} g_1(\hat{x}_k), \hat{\nabla} g_2(\hat{x}_k) \right]^T$$

$$i = 1, 2 \quad \Rightarrow \quad \begin{cases} \hat{\nabla} g_1(\hat{x}_k) = \dfrac{g(\hat{x}_k + \beta_k e_1) - g(\hat{x}_k - \beta_k e_1)}{2\beta_k} \\ \hat{\nabla} g_2(\hat{x}_k) = \dfrac{g(\hat{x}_k + \beta_k e_2) - g(\hat{x}_k - \beta_k e_2)}{2\beta_k} \end{cases}$$

In two-dimensional space, measurements of $g(x)$ needed to calculate the first and second components of gradient vector per iteration are $g(\hat{x}_k \pm \beta_k e_1)$ and $g(\hat{x}_k \pm \beta_k e_2)$. So the number of measurements needed to calculate each component of gradient vector is equal to search space dimension.
The major disadvantage of FD algorithm is that it requires $2n$ measurement of objective function value per iteration. One of the main shortcomings of FD algorithm is its inefficiency in high-dimensional problems. Then Kushner and Clark Presented algorithm that it needs only two measurements of the objective function per iteration.

3. 3. Random search double trial algorithm

In random direction approximation of gradient, all component of $\hat{x}_k$ are randomly perturbed together in two separate directions. The i^{th} component of $\hat{\nabla} g(\hat{x}_k)$ (i =1, 2 ,…, n) for a two-side RD approximation is given by

$$\hat{\nabla} g_i(\hat{x}_k) = d_i \frac{g(\hat{x}_k + \beta_k d_k) - g(\hat{x}_k - \beta_k d_k)}{2\beta_k}, \quad \beta_k > 0 \tag{7}$$

At the k^{th} iteration we generate a random vector $d_k = (d_1, ..., d_n)^T$ that it continuously and uniformly is distributed on the $n-$dimensional unit sphere [1].
Algorithm (2) by using two-side RD approximation is called random search double trial algorithm.

RD approximation needs only two measurements per iteration. With the comparison between FD and RD approximation, we have

$$\frac{Number\ of\ measurment\ of\ the\ objective\ function\ in\ RD\ approximation}{Number\ of\ measurment\ of\ the\ objective\ function\ in\ FD\ approximation} \rightarrow \frac{1}{n}$$

So, the number of measurements in algorithm with RD approximation is much less than FD gradient algorithm when n is large. The convergence conditions for the random search double trial algorithm are similar to the convergence conditions for the FD gradient algorithm.

The major disadvantage of algorithms with FD and RD approximation are their slow speed of convergence.

It is very important to generate (or should say simulate) the random numbers that they are covered the most numbers on [0,1] .

Now instead of generating random numbers $d_1, d_2, \ldots, d_n$ in algorithm (7), we simulate it by Halton quasi random numbers algorithm. Let us introduce Halton quasi random numbers.

Halton's sequence is defined as below [4]:

$$X(m) = (\varphi_2(m), \varphi_3(m), ..., \varphi_{pd}(m)),$$

Where *pd* shows the d^{th} prime number in the above definition and, where

$$\varphi_r(m) = a_0 r^{-1} + a_1 r^{-2} + ... + a_{n-1} r^n + ...$$

$a_i \in \{0,1,2,\ldots,r-1\}$ and $m = a_0 r^0 + a_1 r^1 + \ldots + a_n r^n + \ldots$

4. Numerical Results

The Rosenbrock function is a non-convex function used as a performance test problem for optimization algorithms. It is also known as Rosenbrock's valley or Rosenbrock's banana function.

The global minimum is inside a long, narrow, parabolic shaped flat valley. It is defined by

$$g(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

where $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$. It has a global minimum at $x^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ where $g(x^*) = 0$.

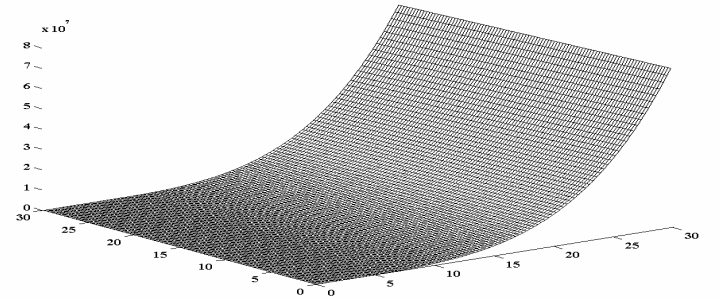

Fig. 1. Plot of Rosenbrock function

Consider obtaining the global minimum of $g(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2,\ 0 \le x_1, x_2 \le 10$, by FD gradient algorithm, random search double trial algorithm based on rand function in Matlab environment and random search double trial algorithm based on Halton's sequence using 600, 2000, 10000, 50000 iterations. We choose starting point as $\hat{x}_0 = \begin{pmatrix} 0.9 \\ 1.2 \end{pmatrix}$, then we obtain $g(\hat{x}_0) = 15.21999999999999$.

Here for all employed algorithms, we set $\alpha = 0.602$ and $\gamma = 0.101$ and $A = 0.1 \times N$ in gain sequences.

Also, we set $a = 0.0086$ and $c = 0.001$ for $N = 600$, $a = 0.0207$ for $N = 2000$, $a = 0.054$ for $N = 10000$ and $a = 1.2$ for $N = 50000$. Also, for 2000, 10000, 50000 iterations we use $c = 0.00001$.

The results listed in the following table.

Table I. Comparison of RD and RD -Halton results with FD Algorithm results

N	$\hat{x}$		$g(\hat{x})$
600	*FD*	$\begin{pmatrix}1.04847958367509\\1.09952026738638\end{pmatrix}$	0.00235471496232
	RD-Mean of 50 replications	$\begin{pmatrix}1.04034619352909\\1.08250357759871\end{pmatrix}$	0.00242712860231
	RD-Halton	$\begin{pmatrix}1.00146395193742\\1.00293562265496\end{pmatrix}$	$2.146264034315441\times10^{-6}$
2000	*FD*	$\begin{pmatrix}1.04114147898139\\1.08413618469146\end{pmatrix}$	0. 00169520070338
	RD-Mean of 50 replications	$\begin{pmatrix}1.04450817536657\\1.09172361543401\end{pmatrix}$	0. 00203372695894
	RD-Halton	$\begin{pmatrix}1.00098813037547\\1.00198078756024\end{pmatrix}$	$9.776621783789740\times10^{-7}$
10000	*FD*	$\begin{pmatrix}1.01554248457459\\1.03138815379595\end{pmatrix}$	$2.419484776790686\times10^{-4}$
	RD-Mean of 50 replications	$\begin{pmatrix}1.02860283688368\\1.05837508202401\end{pmatrix}$	$8.304624616890636\times10^{-4}$
	RD-Halton	$\begin{pmatrix}0.99958686813872\\0.99917234259710\end{pmatrix}$	$1.709226564864338\times10^{-7}$
50000	*FD*	$\begin{pmatrix}0.99999979665523\\0.99999959249688\end{pmatrix}$	$4.141529321924881\times10^{-14}$
	RD-Mean of 50 replications	$\begin{pmatrix}0.99999997874792\\0.99999995741856\end{pmatrix}$	$4.522481344918939\times10^{-16}$

	RD-Halton	$\begin{pmatrix} 0.99991724854152 \\ 0.99983439330801 \end{pmatrix}$	$6.849027622007471 \times 10^{-9}$

Table II. Comparison of RD and RD -Halton errors with FD Algorithm error

N	$\frac{\left\|\hat{x}-x^*\right\|}{\left\|\hat{x}_0-x^*\right\|}$		$\frac{\left\|g(\hat{x})-g(x^*)\right\|}{\left\|g(\hat{x}_0)-g(x^*)\right\|}$
600	*FD*	0.49506673598548	$1.547118897714699 \times 10^{-4}$
	RD-Mean of 50 replications	0.49935642414869	$1.594696847772847 \times 10^{-4}$
	RD-Halton	0.01467040261710	$1.410160337920790 \times 10^{-7}$
2000	*FD*	0.41884409670453	$1.113798096833359 \times 10^{-4}$
	RD-Mean of 50 replications	0.45594296359848	$1.336220078144072 \times 10^{-4}$
	RD-Halton	0.00989941513193	$6.423535994605614 \times 10^{-8}$
10000	*FD*	0.15663875800515	$1.589674623384157 \times 10^{-5}$
	RD-Mean of 50 replications	0.29071540994948	$5.456389367208042 \times 10^{-5}$
	RD-Halton	0. 00413689427320	$1.123013511737411 \times 10^{-8}$

50000	*FD*	2.036702664347351 $\times 10^{-6}$	2.721109935561684 $\times 10^{-15}$
	RD- Mean of 50 replications	2.128299800409542 $\times 10^{-7}$	2.971406928330447 $\times 10^{-17}$
	RD- Halton	8.279297109360839 $\times 10^{-4}$	4.500018148493742 $\times 10^{-10}$

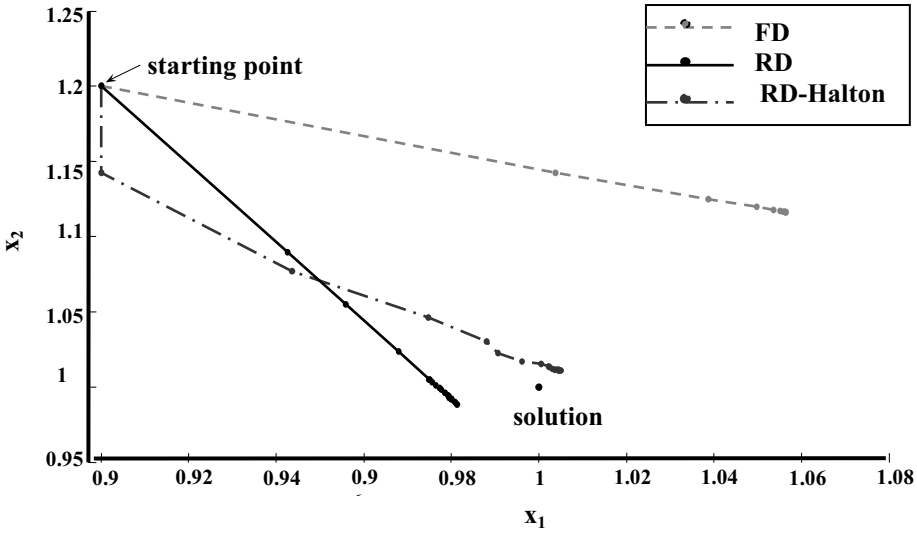

Fig. 2. Search paths for **FD, RD** and **RD-Halton** Algorithms with 20 iterations

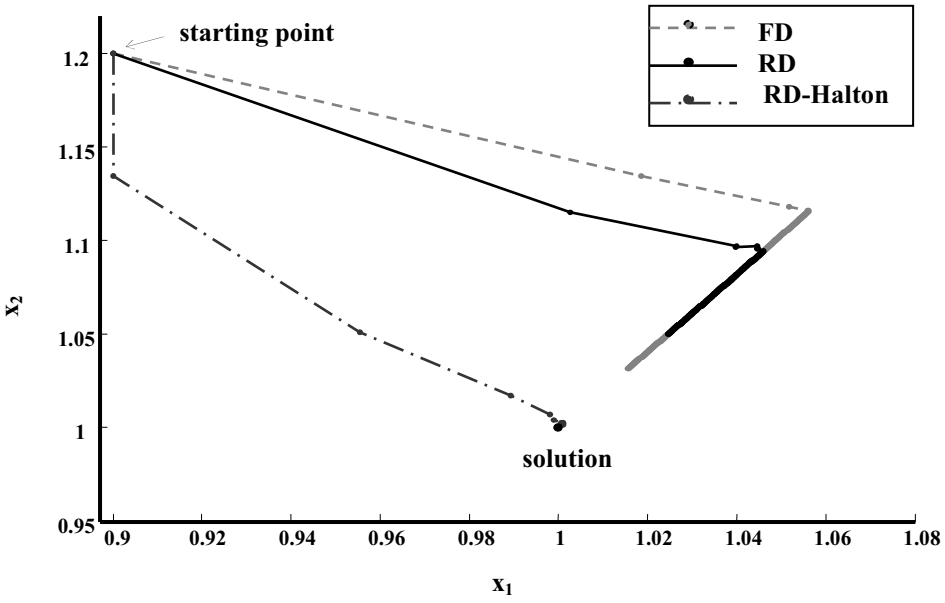

Fig. 3. Search paths for FD, RD and RD-Halton Algorithms with 10000 iterations

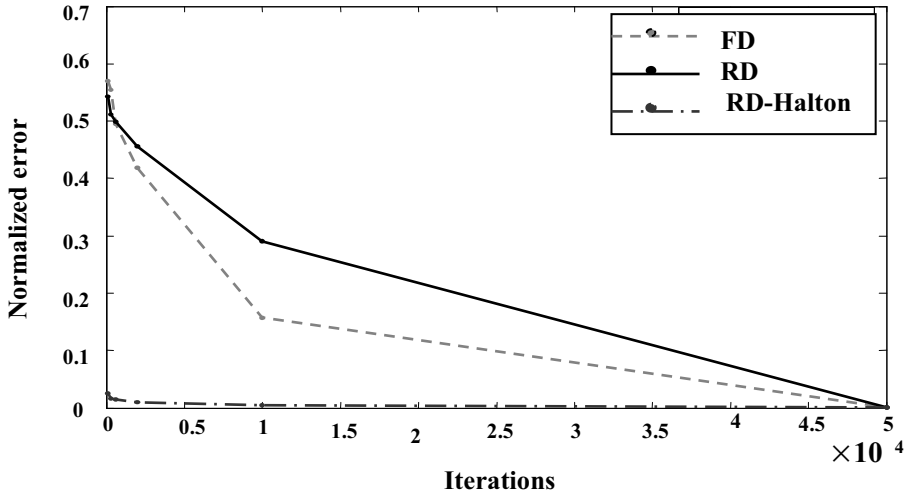

Fig. 4. Normalized errors of $\hat{x}$ for FD, RD and RD-Halton Algorithms

Conclusion

Stochastic algorithm based on Halton quasi random numbers generator (RD-Halton) employed for obtaining accurate results in optimization problems. Based on presented results in Table II and Fig 3, we conclude that the RD-Halton has better performances than RD and FD algorithms. This performance of RD-Halton method increases with

increasing number of random numbers, too. However, FD and RD algorithms have more accurate results than RD-Halton just for $N = 50000$ but this cannot be valid for other N values (i.e. $N = 600, 2000, 10000$).

REFERENCES

[1] XU, Z, AND YU-HANG DAI, 2008, A stochastic approximation frame algorithm with adaptive directions, *Numer. Math. Theor. Meth. Appl 1*, 460-474 .

[2] RUBINSTEIN, R. Y. 1981. *Simulation and the Monte Carlo method*, John Wiley & Sons ; New York.

[3] SPALL, J. C. 2003. *Introduction to stochastic search and optimization. Estimation, Simulation and Control*, John Wiley & Sons ; New Jersey .

[4] SPALL, J. C. 1994. Developments in stochastic optimization algorithms with gradient approximations based on function measurements, *Proceedings of the Winter Simulation Conference*, In J. D. Tew, M. S. Manivannan, D. A. Sadowski, and A. F. Seila (Eds.), 207-214.

BEHROUZ FATHI VJARGAH
MOHADESEH KANAFCHIAN
Department of Mathematics,
Faculty of Sciences,
University of Guilan,
P.O. BOX 1914, Rasht,
Iran

Received June 2010