IPSJ SIG Technical Report

An Improvement of the Stochastic Algorithm for Solving the Sum-of-Ratios Problem

Yangyang Hu[*1], Shinya Watanabe[*1,*2],
Yongkang Ji[*1] and Jianming Shi[*1,*2]

There are many applications of Sum-of-Ratios (SOR) problem in the fields of engineering and economy. Theoretically, the SOR problem is $\mathcal{NP}$-hard. Most existing deterministic algorithms are of branch-and-bound. When the number of terms of ratios is greater than 30, the SOR problem can not be solved by these algorithms within a reasonable time. On the other hand, recently, the stochastic algorithm has been well developed to find an ε-optimal solution to the SOR problem. We first improve such an algorithm by using *line search* method, give some theoretical results for the convergence of the proposed algorithm, and we apply the modified algorithm to solving the SOR problem. The results of computational experiments we conducted show that the modified algorithm is quite efficient than its ancestor.

1. Introduction

It is no need to mention that the importance of global optimization comes from primarily the increasing needs of applications in engineering, finance, computational chemistry, bioinformatics, medicine and many other areas.

An algorithm called *Pure Adaptive Search* (PAS)[7] gives that for convex programs the computational complexity of the algorithm increases at most linearly in the dimension of the problem. These surprising results of PAS can be extended for solving global optimization problems under the *Lipschitz* condition[10]. Although the theoretical result of linear time complexity for global optimization is interesting in itself, there is no better alternative for efficiently generating uniform points in the region, so *Improving Hit-and-Run* (IHR) algorithm[1),6),11)] are proposed to generate a sequence of random points by proving a random direction and a uniform random point in the intersection of that direction and the region.

In this paper, we 1) propose an improvement IHRLS of IHR by using line search algorithm (LS); 2) give some theoretical convergent results; 3) do an empirical study on the performance of the modified algorithm. As an application we solve the Sum-of-Ratios (SOR) problem using two dynamic-multistart versions of DMIHR and DMIHRLS of IHR and IHRLS, respectively.

The paper is organized as follows. In Section 2, we briefly review the background of relative topics and existing results we need through the paper. Algorithms IHRLS and DMIHRLS will be proposed in Section 3. The convergence results are also provided in this section. In Section 4, we investigate the efficiency of the modified algorithm by conducing numerical experiments and report the results. Finally, some conclusions including further work will be remarked in Section 5.

*1 College of Information and Electronic Engineering, Muroran Institute of Technology, Mruoran, Japan
*2 Corresponding authors. {sin, shi}@mmm.muroran-it.ac.jp

2. A Brief Review of Background

In this section, we give a brief review of stochastic algorithm, line search method and the Sum-of-Ratios problem. Hereafter we use the background to develop an improvement of the algorithm using line search and its applications to Sum-of-Ratios problem.

2.1 Stochastic Algorithms

We start with the following optimization problem (P).

$$(P) \quad \left| \begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in S \end{array} \right.$$

In this paper, we assume that

- the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function
- S is a nonempty compact subset in $\mathbb{R}^n$
- problem (P) has an optimal solution x_* with the optimal value y_*, that is, $y_* = f(x_*) \le f(x), \forall x \in S$.
- the maximal value $\max\{f(x) \,|\, x \in S\}$ is given and denote by $\max(P)$.

Therefore the optimal solution set

$$S_{y_*} = \{x \in S \,|\, f(x) \le y_*\}$$

is nonempty.

A stochastic algorithm cannot guarantee to find an exact global optimal solution. For a given tolerance ε such an algorithm can give the probability of a value of function that is not greater than $y_* + \varepsilon$ in the following fashion.

$$p(f(X) \leq y_* + \varepsilon, X \in S) \geq 1 - \alpha, \tag{1}$$

where $\alpha \in (0, 1)$ is a user-determined parameter, X is generated randomly and uniformly in the algorithm. In this paper, a random variable is written in uppercase.

Pure Adaptive Search (PAS) has some amusing theoretical properties. This algorithm is the base of this research. The framework of PAS is described as follows, where, as well as through this paper, Step 0 contributes to the initial settings of parameters.

Pure Adaptive Search (PAS)

Step 0. Uniformly generate a point $X_0 \in S$. Set $Y_0 := f(X_0)$ and $k := 0$.

Step 1. Set $S_{Y_k} := \{x \in S \mid f(x) \leq Y_k\}$. Uniformly generate $X_{k+1} \in S_{Y_k}$.

Step 2. Set $Y_{k+1} := f(X_{k+1})$. Terminate if a terminal criterion is satisfied. Otherwise, $k := k + 1$. Go to **Step 1**.

One of theoretical properties of PAS is that for a given $y \in$ that is a real number between y_* and $\max(P)$ the probability of objective function value Y_k that is generated by PAS and is less than or equal to y at iteration k can be calculated as follows[2].

$$p(Y_k \leq y) = \sum_{i=0}^{k} \frac{p(X \in S_y)(\ln(1/p(X \in S_y)))^i}{i!}, \tag{2}$$

where $p(X \in S_y)$ is the probability that X obtained by PAS is in set S_y.

Recall that *Lipschitz* condition K of $f(x)$ holds over S if and only if $|f(x) - f(y)| \leq K \parallel x - y \parallel$ holds for all $x, y \in S$. When such Lipschitz condition K and the diameter D of S are given, the probability $p(X \in S_y)$ in (2) is bounded as follows[10].

$$p(X \in S_y) \geq \left(\frac{y - y_*}{KD}\right)^n, \tag{3}$$

where n is the dimension of x.

A multistart version of PAS is proposed recently[5]. With such a multistart strategy, X obtained from the multistart algorithm has a probability $p(f(X) \leq y_* + \varepsilon, X \in S)$ (refer to (1)) that is provided below[2].

$$p_\varepsilon := 1 - \prod_{k=0}^{j} \left(1 - \sum_{i=0}^{s_k} \frac{(\varepsilon/KD)^n(\ln(KD/\varepsilon)^n)^i}{i!}\right), \tag{4}$$

where s_k is a number of points obtained through PAS on improving level sets in the kth restart execution, which is not predetermined at the beginning, but determined during the execution. It implies that if $p_\varepsilon \geq 1 - \alpha$ is satisfied then the function value at point X obtained through the algorithm is expected less than or equal to $y_* + \varepsilon$ with a probability at least $1 - \alpha$, that is,

$$p(f(X) \leq y_* + \varepsilon, X \in S) \geq p_\varepsilon \geq 1 - \alpha \tag{5}$$

In other words, the condition can serve as a stopping criteria. When this condition is satisfied, we can stop the algorithm and obtain that $f(X) \leq y_* + \varepsilon$ with a probability $1 - \alpha$ at least. Note that all values embedded in (4) are available, so it is calculable! More detailed information about condition (5) for implementation can be found in the references of this paper and therein.

Although the results in (2) as well as (4) are quite general and shirking, as we mentioned in Section 1, PAS is not easy to implement. The difficulties of implementing PAS come from primarily that generating uniformly in S_{Y_k}.

To circumvent such difficulties, Hit-and-Run is employed to serve an approximation of implementing PAS. This method is called Improving Hit-and-Run (IHR)[11] and works as follows.

Improving Hit-and-Run (IHR)

Step 0. Initialize $X_0 \in S, Y_0 := f(X_0)$, and set $k := 0$.

Step 1. Generate a random direction D_k uniformly on the surface of the unit hypersphere.

Step 2. If $L_k = \{X_k\}$, go to **Step 1**.
Generate a candidate point $W_{k+1} := X_k + \lambda D_k$ by sampling uniformly over the line set

$$L_k := \{x \in S : x = X_k + \lambda D_k, \lambda \textit{ is a real scalar}\}$$

Step 3. Update the current point X_{k+1} with the candidate point if it is improving, i.e., set

$$X_{k+1} = \begin{cases} W_{k+1} & \text{if } f(W_{k+1} < Y_k) \\ X_k & \text{otherwise} \end{cases}$$

and set $Y_{k+1} = f(X_k + 1)$.

Step 4. If a stopping criterion is met, **stop**. Otherwise, $k := k + 1$ and return to **Step 1**.

Lemma 1 [11] Suppose that all level sets S_{Y_k} generated in IHR are elliptical in shape, then the expected number of evaluations of $f(x)$ needed to achieve an ε-optimal solution is at most $O(n^{5/2})$.

Proof: Note that $S_{Y_k} := \{x \in S \mid f(x) \leq Y_k\}$ and that ε-optimal solution x_*^ε is defined as

$$f(x_*^\varepsilon) \leq y_* + \varepsilon.$$

Set $y = y_* + \varepsilon$ then the desired result follows from Corollary 3.6 in[11]. ■

A mutlistart version of IHR has been proposed[2] and works as follows.

Dynamic Multistart Improving Hit-and-Run (DMIHR)

Step 0. Set parameters ε, α and maximum number θ of function evaluations for a single run of IHR. Calculate D, K. Set n and $j = 1$.

Step 1. Execute θ iterations for IHR. Account the number s_j of points sampled uniformly on the improving level sets, also record the best objective value $\bar{Y}_j = f(\bar{X}_j)$ and the best solution $\bar{X}_j$ in the jth run.

Step 2. Update the current best object function values by taking $\min_j\{\bar{Y}_j\}$ and its associated current best solution.

Step 3. Calculate P_ε defined in (4).

Step 4. If $p_\varepsilon \geq 1 - \alpha$, **stop**. Otherwise, $j := j + 1$ go to to **Step 1**.

2.2 Line Search

Suppose that the objective $f(x)$ in problem (P) is differentiable on S. When a point x_k and a direction D_k are given at iteration k, a line segment l_k can be defined as follows.

$$l_k := \{x \mid (x = x_k + \lambda_k D_k, \lambda_k > 0) \text{ and } (x \in S)\}. \tag{6}$$

The objective function $f(x)$ restricted on l_k turns into the following $\phi(\cdot)$

$$\phi(\lambda_k) = f(x_k + \lambda_k D_k),\ \ \lambda_k > 0, x_k + \lambda_k D_k \in S \tag{7}$$

$$\phi'(\lambda_k) = \nabla f(x_k + \lambda_k D_k), \tag{8}$$

where ∇ stands for a gradient. We are interested in shrinking l_k to the smaller intervals that include optimal points of $\phi(\lambda)$ on l_k and finding a point in the interval. To this end, we use the following *Wolfe conditions* (9) and (10) to find the potential intervals.

$$f(x_k + \lambda_k D_k) \leq f(x_k) + c_1 \lambda_k (\nabla f_k)^\top D_k \tag{9}$$

$$(\nabla f(x_k + \lambda_k D_k))^\top D_k \geq c_2 (\nabla f_k)^\top D_k \tag{10}$$

with $0 < c_1 < c_2 < 1$ and are predetermined by the user, $\top$ stands for the transpose of a vector or matrix.

The following condition (11) can be used to impose α to lie at least a neighborhood of a local minimizer or stationary point of ϕ on l_k, if needed.

$$|\nabla f(x_k + \lambda_k D_k)^T D_k| \leq c_2 |\nabla f_k^T D_k| \tag{11}$$

Next we describe the *Line Search Algorithm*[4] with help of *Zoom* subroutine following immediately.

Line Search Algorithm (LS)

Step 0. Set $\lambda_0 = 0$ and $i = 1$, choose $\lambda_1 > 0$ and λ_{max} according to the feasible region.

Step 1. If $[\phi(\lambda_i) > \phi(0) + c_1 \lambda_i \phi'(0)]$ or $[\phi(\lambda_i) \geq \phi_(\lambda_{i-1})$ and $i > 1]$

$\lambda_\star =$ **zoom**$(\lambda_{i-1}, \lambda_i)$ and **stop**;

Step 2. If $\mid \phi'(\lambda_i) \mid \leq -c_2 \phi'(0)$

$\lambda_\star = \lambda_i$, and **stop**;

Step 3. If $\phi'(\lambda_i) \geq 0$

$\lambda_\star =$ **zoom**$(\lambda_i, \lambda_{i-1})$ and **stop**;

Step 4. Choose $\lambda_{i+1} \in (\lambda_i, \lambda_{\max})$. If a stopping criterion hold, **stop**. Otherwise, $i := i + 1$ and return to **Step 1**.

Now we describe *Zoom* algorithm below. In each iteration we try to get an acceptable $\lambda\star$ or replace the endpoint by λ_j which is between λ_{lo} and λ_{hi}.

Zoom Algorithm

Step 0. Interpolate (using quadratic, cubic, or bisection) to find a trial step λ_j between λ_{lo} and λ_{hi}.

Step 1. Evaluate $\phi(\lambda_j)$

Step 1.1. If $\phi(\lambda_j) > \phi(0) + c_1\lambda_i\phi'(0)$ or $\phi(\lambda_j) \geq \phi(\lambda_{lo})$

$\lambda_{hi} = \lambda_j$;

else, Evaluate $\phi'(\lambda_j)$

Step 1.2. If $\mid \phi'(\lambda_j) \mid \leq -c_2\phi'(0)$

Set $\lambda_\star = \lambda_j$ and **stop**;

Step 1.3. $\mid \phi'(\lambda_j) \mid (\lambda_{hi} - \lambda_{lo}) \geq 0$

$\lambda_{hi} = \lambda_{lo}$,$\lambda_{lo} = \lambda_j$

Step 2. If $(\lambda_{hi} - \lambda_{lo})$ is small enough, **stop**. Otherwise, return to **Step 1**.

Note that in Line Search, $\phi'(0) < 0$ because it starts with a descent direction.

2.3 The Sum of Ratios Problem

Sum-of-Ratios (SOR) problem can be defined as follows

$$\text{(SOR)} \quad \left| \begin{array}{ll} \text{minimize} & \displaystyle\sum_{s=1}^{q} \frac{g_s(x)}{h_s(x)} \\ \text{subject to} & x \in S \end{array} \right. \tag{12}$$

where $g_s(x), h_s(x) : \mathbb{R}^n \rightarrow \mathbb{R}$ and $s = 1, 2, \ldots, q$. Generally, g, h can be linear, quadratic, or more general functions. The SOR problem has many applications, such as the transportation problem, government contracting problem, portfolio optimization, optimal clustering problems, etc[8),9)].

Up to now, we have been lacking in investigating the performance of IHR and its multistart version to solve the SOR and related problems when we apply Line Search. An improvement of IHR described in next section and the results of numerical experiments reported in Section 4 are an attempt to contribute the issue.

3. An Improvement of IHR and DMIHRLS

Considering an improvement of IHR, we replace X_k in IHR by x_k that is obtained through algorithm LS. It has been observed that Line Search has a high ability to find a minimizer of $\phi(\cdot)$ deterministically. By taking a small value of c_2, the condition (11) results in finding an interior minimizer or a stationary point on l_k. The advanced improvement described herein is inspired by the excellent ability of global search of Line Search. Based on these observations, now we propose a new algorithm, which is basically a stochastic one.

Improving Hit-and-Run with Line Search Algorithm (IHRLS)

Step 0. Initialize $X_0 \in S, Y_0 = f(X_0)$, set $k = 0$.

Step 1. Generate a random descent direction D_k uniformly distributed on the surface of the unit hypersphere.

Step 2. Generate a candidate point $W_{k+1} = X_k + \lambda_\star D_k$ using line search as follows.

Step 2.0. Set $\lambda_0 = 0$ and $i = 1$, choose $\lambda_1 > 0$ and λ_{max} according to the feasible region.

Step 2.1. If $[\phi(\lambda_i) > \phi(0) + c_1\lambda_i\phi'(0)]$ or $[\phi(\lambda_i) \geq \phi_(\lambda_{i-1})$ and $i > 1]$

$\lambda_\star = \mathbf{zoom}(\lambda_{i-1}, \lambda_i)$ and go to **Step 3**.

Step 2.2. If $|\phi'(\lambda_i)| \leq -c_2\phi'(0)$

$\lambda_\star = \lambda_i$, and go to **Step 3**.

Step 2.3. If $\phi'(\lambda_i) \geq 0$

$\lambda_\star = \mathbf{zoom}(\lambda_i, \lambda_{i-1})$ and go to **Step 3**.

Step 2.4. If $DK/2^i \leq \varepsilon$, go to **Step 3**. Otherwise, choose $\lambda_{i+1} \in (\lambda_i, \lambda\text{max})$, $i := i + 1$, go to **Step 2**.

Step 3. Update the current point X_{k+1} with the candidate point if it is improving the function value, i.e., set

$$X_{k+1} = \begin{cases} W_{k+1} & \text{if } f(W_{k+1}) < Y_k \\ X_k & \text{otherwise} \end{cases}$$

and set $Y_{k+1} = f(X_{k+1})$.

Step 4. If a stopping criterion is met, **stop**. Otherwise, $k := k + 1$, and go to **Step 1**.

Denote a neighborhood of $\bar{x}$ by

$N_\delta(\bar{x}) := \{y \in \mathbb{R}^n \mid \parallel y - x \parallel \leq \delta\}$

and a set of local minimizers by

$S_{\bar{x}}(\delta) := \{z \in S \,|\, \exists N_\delta(\bar{x}) \text{ such that } f(z) \leq f(x), \forall x \in S \cap N_\delta(\bar{x})\}$

and the set of the stationary point in S by S_s.

Assumption 1 : *Problem* (P) *has finitely many local minimizers and stationary points.* Under Assumption 1 we have the following lemma.

Lemma 2 Suppose that Assumption 1 holds, then a probability that there does not exist a descent direction D_k generated randomly and uniformly in Step 1 is zero.

Proof: It is easy to see that the *Lebesgue measure* of both $S_{\bar{x}}(\delta)$ and S_s is 0 due to Assumption 1. So a probability that a line l_k meets $S_{\bar{x}}(\delta) \cup S_s$ is zero. That implies the desired result. ■

Lemma 3 Suppose that a bisection is used in Zoom in Step 2 and that the conditions for (3) holds, then Step 2 terminates within

$$I_{\text{step2}} := \lceil \ln(DK/\varepsilon) \rceil$$

iterations, where $\lceil \cdot \rceil$ is the ceiling function.

Proof: The Lipschitz condition yields that after I_{step2} iterations

$$|f(x) - f(y)| \leq \frac{DK}{2^{I_{\text{step2}}}}$$

for all $x, y \in L_k$. So if $I_{\text{step2}} \geq \lceil \ln(DK/\varepsilon) \rceil$ is satisfied then a stopping creterion at Step 2.4 is met. ■

Theorem 1 Suppose that Assumtion 1 and conditions in Lemma 1 and 3 are satisfied then the expected number of evaluations of $f(x)$ needed to approximate an ε-optimal solution is at most $O(n^{5/2} \ln(DK/\varepsilon))$

Proof: It follows from Lemma 1 that the target expected number is not great than

$$O(n^{5/2}).$$

That is, the number of k in IHRLS is not greater than $O(n^{5/2})$. Note that the iterations in Step 2 will not exceed $\ln(DK/\varepsilon))$. This implies the desired assertion. ■

The IHR procedure in DMIHR can be replaced by IHRLS to make a multstart version, which is called herein DMIHRLS algorithm and works as follows.

DMIHR with Line Search (DMIHRLS)

Step 0. Replace IHR by IHRLS and execute Step 0 of DMIHR.
Step 1. Replace IHR by IHRLS and execute Step 1 of DMIHR.
Step 2. Execute Step 2-4 of DMIHR.

4. Numerical Experiments

In this section we conduce the numerical experiments to compare the efficiency our improvement and its ancestor, namely, DMIHRLS and DMIHR. The predetermined parameters($\alpha = 0.01$, $\epsilon = 0.01$ and 100 sets of the algorithms for each θ) we used in our experiments are the same as they used[2], where theses examples and datasets are used for examining DMIHR algorithm.

Example 1[3] The first example is the sum-of-linear-ratios optimization problem.

$$(EX_1) \left| \begin{array}{ll} \text{maximize} & \dfrac{3x_1 + x_2 - 2x_3 + 0.8}{2x_1 - x_2 + x_3} + \dfrac{4x_1 - 2x_2 + x_3}{7x_1 + 3x_2 - x_3} \\ \text{subject to} & x_1 + x_2 - x_3 \leq 1 \\ & -x_1 + x_2 - x_3 \leq -1 \\ & 12x_1 + 5x_2 + 12x_3 \leq 1 \\ & 12x_1 + 12x_2 + 7x_3 \leq 1 \\ & -6x_1 + x_2 + x_3 \leq -4.1 \\ & -x_1, -x_2, -x_3 \leq 0 \end{array} \right.$$

The optimal value 2.4714 is known, that is the objective function value at the point $(x_1, x_2, x_3) = (1, 0, 0)$.

For the both of DMIHR and DMIHRLS, Table 1 shows that as θ increases the number of restarts gradually decreases and the number of improving points per restart becomes larger and larger. When the θ is large enough, we do not need restart any more. It means the DMIHR is just a simple IHR and DMIHRLS is only an IHRLS. Table 1 tells that algorithm DMIHRLS has a higher ability to obtain a better solution than DMIHR. Figure 1 indicates that our algorithm DMIHRLS finds a better solution robustly, while the lines of DMIHR are tossed up-and-down especially for a θ less than around 70. We observe that the function of Example 1 is relative simple, so line search finds a better solution easily than a random search even for a smaller θ.

Table 1 Results for Example 1 (DMIHR: A1, DMIHRLS: A2)

	θ	Number of restarts			Number of improving points per restart			Best value found		
		Min	Average	Max	Min	Average	Max	Min	Average	Max
A1	50	1	6.02	29	6	15.78	26	2.2390	2.4095	2.4698
A2	50	1	4.69	19	1	13.55	33	2.1052	2.4291	2.4705
A1	60	1	2.46	9	10	18.33	31	2.1387	2.4010	2.4694
A2	60	1	2.80	12	1	16.67	37	2.2040	2.4446	2.4703
A1	70	1	1.49	5	11	20.71	33	2.1956	2.3900	2.4665
A2	70	1	1.97	13	1	17.15	42	2.0677	2.4463	2.4711
A1	80	1	1.06	2	13	23.02	33	2.1523	2.3973	2.4681
A2	80	1	1.66	6	1	20.59	48	2.1144	2.4444	2.4710
A1	90	1	1.03	2	15	25.91	36	1.9734	2.3943	2.4700
A2	90	1	1.48	5	1	25.08	55	2.2344	2.4562	2.4713
A1	100	1	1.03	2	13	26.92	39	1.8811	2.3735	2.4704
A2	100	1	1.44	4	3	25.37	53	2.1981	2.4572	2.4714
A1	150	1	1.00	1	22	40.08	53	2.1115	2.4231	2.4713
A2	150	1	1.21	5	4	28.72	90	2.0822	2.4602	2.4714
A1	200	1	1.00	1	32	51.02	74	1.9722	2.4255	2.4714
A2	200	1	1.15	3	6	40.09	103	2.3207	2.4637	2.4714

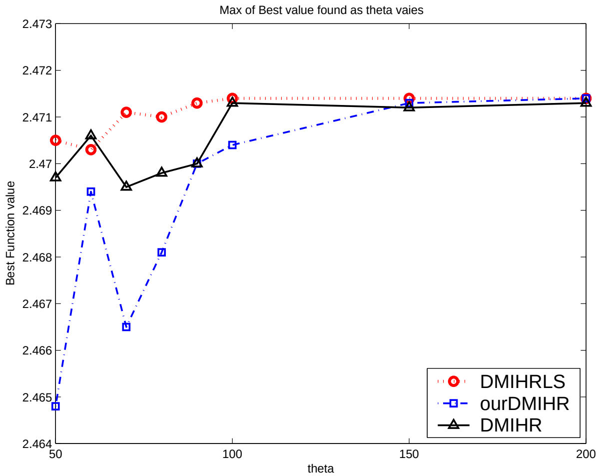

Fig. 1 Best function value for Example 1

5. Concluding remarks

In this research we review the stochastic optimization, Line search and the sum-of-ratios problem. We have proposed a new algorithm which improves its ancestor. We also discuss the convergence of the new algorithm and give an expected number of iterations to get an ε-minimizer. Numerical experiments show that the proposed algorithm find the best solutions better in average, especially for a problem having a relative simple objective or with a larger θ.

References

1) Bélisle, C.J.P., Romeijn, H.E. and Smith, R.L.: Hit-and-Run Algorithms for Generating Multivariate Distributions, *Mathematics of Operations Research*, **18**, 255-266 (1993).
2) Dúr, M., Khompatraporn, C. and Zabinsky, Z.B.: Solving Fractional problems with dynamic multistart improving hit-and-run, *Annals of Operations Research*, **156**, 25-44 (2007).
3) Falk, J.E. and Palocsay, S.W.: Optimizing the Sum of Linear Fractional Functions, in Collection: Recent Advances in Global Optimization, C.A. Floudas and P.M. Pardalos (eds.), 221-258 (1992).
4) Jorge, N. and Stephen J.W.: Numerical Optimization, Springer, Series in Operations Research (1999).
5) Khompatraporn, C.: Analysis and development of stopping creteria for stochastic global optimization, Ph.D. Dissertation, University of Washinton, Seattle, WA (2004).
6) Lovász, L. and Vempala, S.: Hit-and-Run is Fast and Fun, MicroSoft Technical Report, **05** (2003).
7) Patel, N.R., Smith, R.L. and Zabinsky, Z.B.: Pure adaptive search in Monte Carlo optimization, *Mathematical Programming*, **4**, 317-328 (1988).
8) Schaible, S. and Shi, J.: Fractional programming: the sum-of-ratios case, *Optimization Methods and Software*, **18**, 219-229 (2003).
9) Schaible, S.: A note on the sum of a linear and linear-fractional function, *Naval Research Logistics Quarterly* **24**, 691-693 (1977).
10) Zabinsky, Z.B. and Smith, R.L.: Pure Adaptive Search in Global Optimization, *Mathematical Programming*, **53**, 323-338 (1992).
11) Zabinsky, Z.B., Smith, R.L., McDonald, J.F., Romeijn, H.E. and Kaufman, D.E.: Improving Hit and Run for Global Optimization, *Journal of Global Optimization*, **3**, 171-192 (1993).