Journal of Computer Science, 9 (4): 463-476, 2013
ISSN 1549-3636

doi:10.3844/jcssp.2013.463.476 Published Online 9 (4) 2013 (http://www.thescipub.com/jcs.toc)

NELDER-MEAD METHOD WITH LOCAL SELECTION USING NEIGHBORHOOD AND MEMORY FOR STOCHASTIC OPTIMIZATION

Noocharin Tippayawannakorn and Juta Pichitlamken

Department of Industrial Engineering, Faculty of Engineering,
Kasetsart University, Bangkok, 10900, Thailand

Received 2013-01-18, Revised 2013-05-04; Accepted 2013-05-10

ABSTRACT

We consider the Nelder-Mead (NM) simplex algorithm for optimization of discrete-event stochastic simulation models. We propose new modifications of NM to reduce computational time and to improve quality of the estimated optimal solutions. Our means include utilizing past information of already seen solutions, expanding search space to their neighborhood and using adaptive sample sizes. We compare performance of these extensions on six test functions with 3 levels of random variations. We find that using past information leads to reduction of computational efforts by up to 20%. The adaptive modifications need more resources than the non-adaptive counterparts for up to 70% but give better-quality solutions. We recommend the adaptive algorithms with using memory with or without neighborhood structure.

Keywords: Nelder-Mead Simplex, Adaptive Nelder-Mead Simplex, Continuous Stochastic Optimization, Neighborhood Search, Local Selection

1. INTRODUCTION

An Optimization via Simulation (OvS) is the problem of finding possible set of input variables or decision variables that give maximum or minimum objective function values. In addition, a simulation optimization also aims at minimizing computational resources spent while maximizing the information obtained in a simulation experiment (Carson and Maria, 1997). We are interested in the OvS problems that have stochastic objective functions and continuous decision variables (Alon *et al*., 2005; Henderson and Nelson, 2006; Olafsson and Kim, 2002; Swisher *et al*., 2004) for OvS surveys.

Many OvS tools are developed for unconstrained continuous problems. Most of them are based on the random search method that takes objective function values from a set of sample points and uses that information to select the next points. Various techniques differ in the choice of sampling strategies (Andradottir, 2006). A point-based strategy involves sampling points in a neighborhood of the current solution, e.g., the Stochastic Ruler (Alrefaei and Andradottir, 2005) and the Simulated Annealing (Press *et al*., 2007). A set-based strategy generates a set of candidate solutions from a subset of the feasible region, e.g., the Nested Partitions Method (Shi and Olafsson, 2009) and the Nelder-Mead Simplex (Nelder and Mead, 1965). A population-based strategy creates a collection of candidate solutions using some properties of the previously visited solutions; for example, the Genetic Algorithm (Holland, 2000) and the Evolutionary Strategies (Beyer and Schwefel, 2002).

We focus on the Nelder-Mead (NM) simplex algorithm (Nelder and Mead, 1965), which is originally developed for unconstrained deterministic optimization. It demonstrates wide versatility and ease of use such that it is implemented in MATLAB as a function fminsearch. The NM is also robust with respect to small random variations in the observed objective function values; therefore, it is used for optimizing stochastic problems as well (Tomick, 1995; Humphrey and Wilson, 2000) However, in the case that

Corresponding Author: Noocharin Tippayawannakorn, Department of Industrial Engineering, Faculty of Engineering.
Kasetsart University, Bangkok, 10900, Thailand

variability in the objective function values are sufficiently large, the NM may terminate before reaching the global optima. For this, Barton and Ivey (1996) propose the algorithm RS9 that improve NM performance for stochastic problems by increasing the shrink parameter and recalculating every point of shrink simplex.

In this study, we propose variants of the NM by utilizing past information and/or proximate points. Specifically, we incorporate:

- Information collected since the search begins and
- Search neighborhood

With numerical experiments, we show that our algorithms provide better solutions while requiring less computational efforts than the original NM.

Generally, an OvS problem can be defined as follows: The objective is to determine an optimal solution, x*, that minimizes the unknown objective function, μ: $\Theta \rightarrow \mathbb{R}$ over a continuous feasible region, $\Theta \in \mathbb{R}^d$; that is, $x^* = \arg\min_{x\in\Theta} \mu(x)$, where, $x \in \Theta$ is a vector of d decision variables and x is called a solution. The objective function $\mu(x)$ cannot be observed directly; thus, it is estimated with stochastic simulation, i.e. Equation 1:

$$\mu(x) = E_{\xi_x}\left[G(x,\xi_x)\right] \quad (1)$$

where, $G(x,\xi_x)$ is a simulation output evaluated at **x** and ξ_x is an unbiased random element with $E[\xi_x] = 0$ and $V[\xi_x] = \sigma_x^2$. The estimate of $\mu(x)$, $\hat{\mu}(x)$, is a sample mean of m independent simulation outputs:

$$\hat{\mu}(x) = G(x) = \frac{1}{m}\sum_{i=1}^{m} G(x,\xi_i) \quad (2)$$

This study is organized as follows: Section 2 introduces the original form of the NM simplex algorithm, its existing variants, our extensions and describes design of numerical experimens. Section 3 shows of numerical results. Section 4 discusses the results. Ultimately, we conclude in section 5.

2. MATERIALS AND METHODS

2.1. The Nelder-Mead Simplex Algorithm

2.1.1. Original NM

The first of the simplex methods is due to Spendley *et al.* (1962) for deterministic problems. They assume that any point in the domain of search can be constructed by taking a linear combination of the edges adjacent to any given vertices. The original simplex consists of the reflection of one vertex through the centroid of the opposite face. Sometimes a sequence of reflections brings the search back to where it starts. Nelder and Mead (1965) add expansion and contraction moves to accelerate the search and a shrink step is introduced to decrease the lengths of edges adjacent to the current best vertex by half, in case that none of the steps brings acceptable improvement to the original simplex. **Figure 1** illustrates 2-dimensional trial points for a simplex consisting of x_0, x_1 and x_2. The solid lines simplex is initialized. Other line-style simplexes show various simplex operations, e.g., an expansion point is x_2^E, a reflection point is x_2^R, internal and external contraction points are $x_2^{C_i}$ and $x_2^{C_e}$, respectively and C is the centroid of the 2 best points. By the default setting of fminsearch (the NM implementation in MATLAB), a single simulation output (m = 1 in Equation 2) is an estimate of an objective function $\hat{\mu}(x)$.

The overall logical steps of the NM algorithm are shown in **Fig. 2** and it can be explained in more details as follows.

2.2. Initialization: Create an Initial Simplex

- Select a starting point $x_0 \in \Theta$, a vector of d dimensions
- Form an initial simplex of d+1 points, by defining:

$$x_i = [x_1, x_2, \ldots, x_i + s_i, \ldots, x_d], \quad i = 1,2,\ldots,d, \quad (3)$$

where, s_i are the user-specified initial step sizes.

- Estimate the objective function values at each of the d+1 simplex points from m independent simulation outputs by computing its sample averages via Equation 2 to get $\hat{\mu}(x_0), \hat{\mu}(x_1), \ldots, \hat{\mu}(x_d)$ then initialize the iteration number j = 1 and a number of observation of simulation outputs, count = m(d+1)

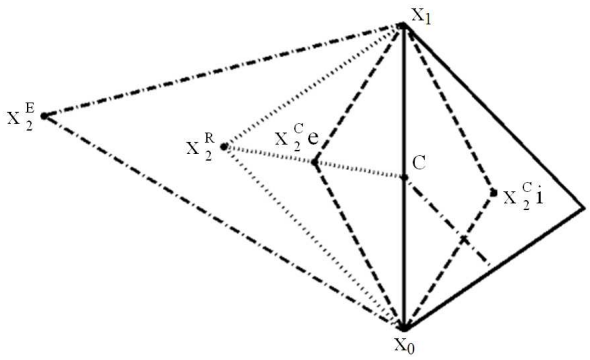

Fig. 1. All simplex operations of the Nelder-Mead simplex

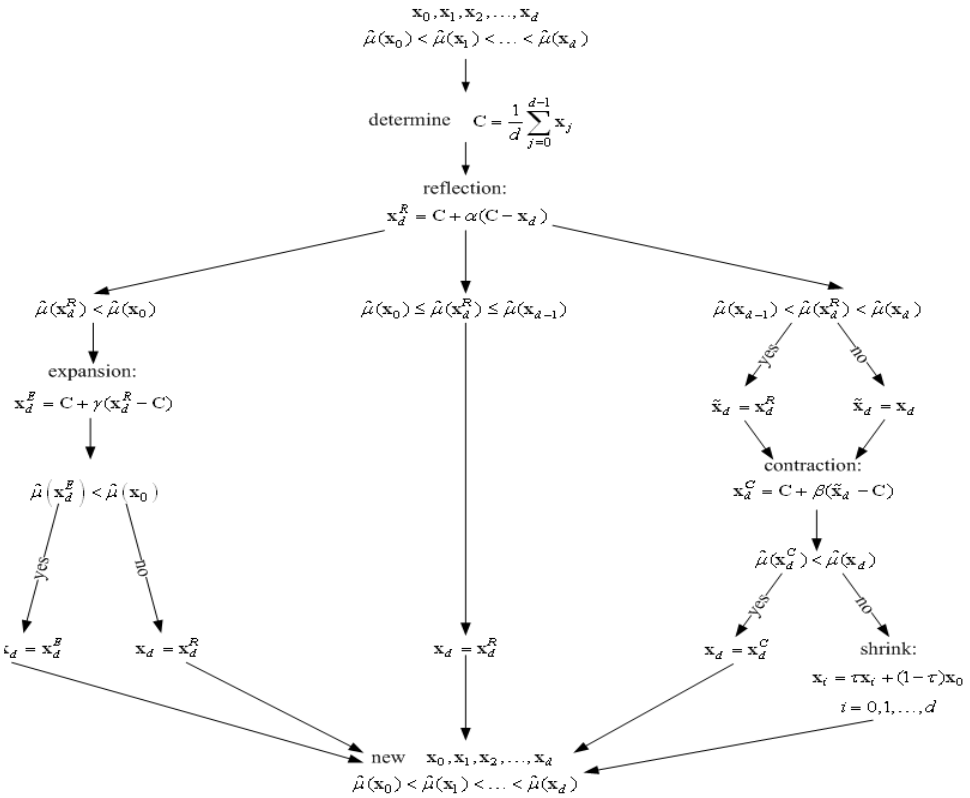

Fig. 2. Flow chart for the j[th] iteration of the Nelder-Mead simplex algorithm

- Re-order these points in a non-decreasing order so that $\hat{\mu}(x_0) \le \hat{\mu}(x_1) \le \ldots \le \hat{\mu}(x_d)$

While $\max_{i=1,2,\ldots,d} |\hat{x}_i - \hat{x}_0| \le \varepsilon_x$ and $\max_{i=1,2,\ldots,d} |\hat{\mu}(\hat{x}_i) - \hat{\mu}(\hat{x}_0)| \le \varepsilon_{\hat{\mu}}$, $j<N_{search}$ and $count < N_{\hat{\mu}}$ are true.

Step 1: Calculate the Reflection Point

A worst point on the simplex (recall that we consider a minimization problem, so the worst point is one with the highest sample mean x_d) x_d is replaced with another point which has a lower objective function. Let x_d^R be the reflection of the worst point and x_d passes through the centroid C of the d-best points. These points are computed as $C=\frac{1}{d}\sum_{j=0}^{d-1} x_j$ and $x_d^R = C+\alpha(C-x_d)$ where, $\alpha<0$ is a reflection parameter; typically 1. Then the objective function value $\hat{\mu}(x_d^R)$ is estimated via m simulation outputs then count = count+m.

Step 2: Update the Simplex

Figure 3 shows an initial simplex with dash lines and an updated simplex with solid lines. The updated simplex depends on the relationship between $\hat{\mu}(x_d^R)$ and $\hat{\mu}(x_0), \hat{\mu}(x_1), \ldots, \hat{\mu}(x_d)$; that is:

- If $\hat{\mu}(x_0) \le \hat{\mu}(x_d^R) < \hat{\mu}(x_{d-1})$, set $x_d \leftarrow x_d^R$ and $\hat{\mu}(x_d) \leftarrow \hat{\mu}(x_d^R)$ as shown in **Fig. 3a**. Then go to Step 4.
- If $\hat{\mu}(x_d^R) < \hat{\mu}(x_0)$, the search continues in the same direction by calculating the expansion point, $x_d^E = C + \gamma(x_d^R - C)$ where $\gamma>0$ is an expansion parameter, typically 2. Then $\hat{\mu}(x_d^E)$ is estimated from m simulation outputs by Equation 2, then count = count + m. The expansion point is accepted when it improves over the best point in the simplex, x_0, as shown in **Fig. 3b**; otherwise, the reflection point is accepted. Go to Step 4.

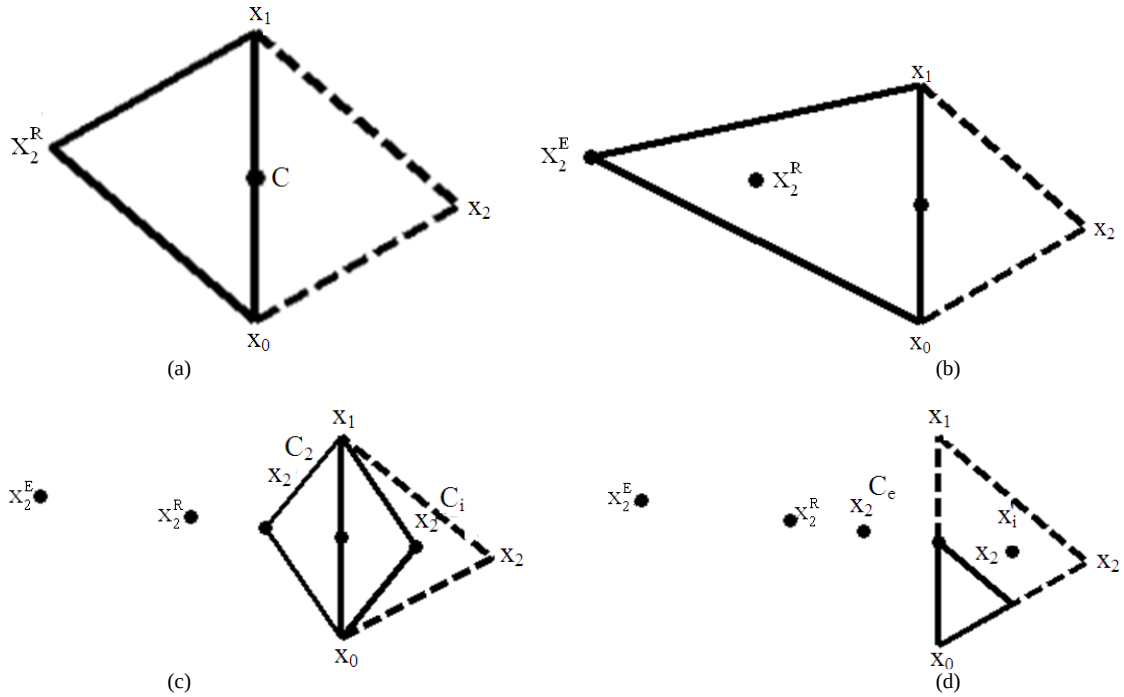

Fig. 3. Operations of the NM simplex method

If $\hat{\mu}(x_{d-1}) \le \hat{\mu}(x_d^R)$, the search reduces the simplex size by calculating the contraction point, $x_d^C = C + \beta(\tilde{x}_d - C)$, where, β>0 is a contract parameter (generally 0.5) and $\tilde{x}_d$ is x_d^R if $\hat{\mu}(x_d^R) < \hat{\mu}(x_d)$ and x_d otherwise. The objective function $\hat{\mu}(x_d^C)$ is estimated then count = count+m. If $\hat{\mu}(x_d^C) \ge \hat{\mu}(x_d)$, go to Step 3; otherwise, the contraction point is accepted, i.e., $x_d \leftarrow x_d^C$. The updated simplex can be one of two solid-line simplexes in **Fig. 3c** depending on which as the optimal $\hat{x}^*$ and $\hat{\mu}(\hat{x}^*)$, respectively, when when the search terminates. Contraction point ($x_2^{c_i}$ or $x_2^{c_e}$) is used. Go to Step 4.

Step 3: Shrink the Simplex

If the reflection point and contraction point provide no improvement, then the simplex is shrunk toward the best point x_0 as shown in **Fig. 3d**. Compute the new simplex as follows:

$$[x_0, \tau x_1 + (1-\tau)x_0, \tau x_2 + (1-\tau)x_0, \ldots, \tau x_d + (1-\tau)x_0] \quad (4)$$

where, τ is a shrink parameter, typically 0.5. The objective function values of these new points are estimated count = count +md. Then go to Step 4.

Step 4: Re-order the Simplex Points in Ascending Orders

Then let j = j+1.

End while
Return $\hat{x}^*$ and $\hat{\mu}(\hat{x}^*)$.

2.3. Barton and Ivey Stochastic Modification of NM and its Variant (RS9 and ANRS9)

Barton and Ivey (1996) adapt the NM algorithm to accommodate stochastic variations in the objective function values. From empirical results, they see that because the NM algorithm relies on the ranks of the objective function values at the simplex vertices, it can make progress in presence of relatively small randomness which does not change the rank of the function value at the simplex points. However, if the variations in the function value are large enough, it affects the relative rank of the simplex vertices and misleads the algorithm.

Barton and Ivey (1996) recommend the shrinkage coefficient (in Equation 4) of 0.9 instead of the usual 0.5 to increase the extent of reduction after shrink. This change improves the performance effectively for the cases where the original NM fails. Resampling the best point after shrink reduces the frequency of contraction, but this strategy is not effective in improving algorithm performance.

Moreover, Barton and Ivey (1996) apply another stopping criterion for stochastic problems, as suggested by Dennis and Woods (1987). The search terminates when the simplex size is sufficiently small:

$$\frac{\sum_{i=1}^{d}\left\|x_i - x_0\right\|}{\max(1,\left\|x_0\right\|)} \le \varepsilon_x \quad (5)$$

where, $\hat{\mu}(x_0) < \hat{\mu}(x_1) < \ldots < \hat{\mu}(x_d)$, and $\|\bullet\|$ is the Euclidean norm, i.e., $\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_d^2}$. Besides the stopping criteria in Equation 5, their so-called RS9 is the NM with the following modifications: The objective function is estimated with 6 independent simulation outputs (m = 6 in Equation 2); every solution is resampled every time it is encountered (no search history is kept); and the shrinkage coefficient τ is 0.9.

The rescaling operations of the NM algorithm can lead to a too-early termination at a non-optimum if noise is present. Tomick *et al.* (1995) modify the RS9 further to allow the sample sizes to adjust adaptively to the observed noise in the solution space, called ANRS. Suppose that m^j is the minimum number of observations taken at each new trial point during the j^{th} iteration and $m^0 = 6$:

$$m^{j+1} = \begin{cases} \lfloor bm^j \rfloor & \text{if } S^2/(d\sigma^2) \le \chi^2_{d,\alpha} \\ m^j & \text{otherwise} \end{cases} \quad (6)$$

where, b = 1.25 is a factor to increase the sample size, d is a size of the decision variable x, $\lfloor x \rfloor$ is the largest integer smaller than x:

$$S^2 = \frac{\sum_{i=1}^{d+1}\left[\sum_{k=1}^{m^j}\hat{\mu}(x_{ik})\right]^2}{m^j} - \frac{\left[\sum_{i=1}^{d+1}\sum_{k=1}^{m^j}\hat{\mu}(x_{ik})\right]^2}{(d+1)m^j}}{d}$$

is the mean square treatments from Analysis of Variance (ANOVA), σ^2 is the variance of white noise, ξ_x in Equation 1 and $\chi^2_{d,\alpha}$ is an α upper percentile of the chi-square distribution with d degree of freedom.

2.4. New Variants of the Nelder-Mead Simplex Algorithm

We are motivated by several general-purpose optimization algorithms for deterministic problems that are based on a neighborhood search; for example, the very large scale neighborhood search (Pichitlamken and Nelson, 2003), neighborhood search based on tabu search and complete local search with memory for solving the uncapacitated facility location problem (Pichitlamken *et al.*, 2006). At each iteration, the search iteratively moves from the current solution to one of its neighbors which is better than itself and any other solutions in the neighbor-hood. Neighborhood search strategy and statistical selection of the best are used in OvS in Tomick *et al.* (1995) followed by a framework for OvS in More *et al.* (1981).

First, we define an "already-visited solution" as x that is not "too far" from the one already seen v, $\|v-x\|_\infty = \max_{1\le i\le d}|v_i - x_i| = e_v$, where, $\|.\|_\infty$ is the uniform norm, $\|x\|_\infty = \max\{|x_1|,|x_2|,\ldots,|x_d|\}$, and the neighbor distance $0 \le e \le \varepsilon_x$ where $\varepsilon_x = 10^{-4}$, similar to the terminating criterion of the fminsearch. We select the neighbor v which provides the minimum e_v as distinguishable from x. For an example of 2-dimensional problem, suppose x = (1.00018,2.50101), v_1 = (1.00018, 2.50110) and v_2 = (1.0011,2.50101), v_3 = (1.00015, 2.601010) then e_1 = max {|1.00018-1.0018|,|2.50101-2.50110}= 9×10^{-5}, e_2 = max (|1.0018-1.0011|,|2.50101|) = 7×10^{-5}, e_3 = max(|1.0018-1.00015|,|2.50101-2.60101) = 0.1. Since e_3 is greater than ε_x, v_3 does not belong to N(x) and we select v_2 to represent x because $e_2 \le \varepsilon_x$ and it has the smallest uniform-norm distance from x.

We define the neighborhood of solution x = $[x_1,x_2,\ldots,x_d]$ as N(x) consisting of all already-seen solutions which lie inside the region of:

$$[x_1-\varepsilon,x_2,\ldots,x_d],[x_1+\varepsilon,x_2,\ldots,x_d],[x_1,x_2-\varepsilon,\ldots,x_d],$$
$$[x_1,x_2+\varepsilon,\ldots,x_d],\ldots,[x_1,x_2,\ldots,x_d+\varepsilon] \quad (7)$$

where, ε is the user-specified maximum neighborhood distance. We exclude any neighbors that lie outside the feasible space Θ, or ones which are further than ε from any given x.

The aim of using past information is to save simulation effort by avoiding sampling at every encounter. In our implementation of the NM search, we compare the sample averages of all candidates and select the one with the smallest average as the winner.

We propose two NM-based algorithms with memory as follows.

2.5. The Nelder-Mead Selection with Memory (NMSM)

Simulation outputs that have been obtained for revisited solutions and their candidate solutions are kept in a database and they replace new sampling. Nevertheless, for already-seen solutions, NMSM adds one simulation output every time it is encountered so as to protect the search from unusually good or bad history.

For each $\mathbf{x} \in \{x_0, x_1, x_2, \ldots, x_d, x^R, x^E, x^C\}$

If $\mathbf{x} \in V^{j-1}$, % embedded NMSM Algorithm

Add one new simulation output and calculate cumulative sample means as a new $\hat{\mu}(x)$ and let $count = count + 1$.

Else % $x \notin V^{j-1}$

For every $\mathbf{x}' \in \mathbf{N}(\mathbf{x})$, compute a new $\hat{\mu}(x')$ as follow.

If there is any revisited neighbor $x' \in V^{j-1}$, use their cumulative simulation outputs.

Else there is any first encounter $x' \notin V^{j-1}$, generate m simulation outputs, then register $\mathbf{x}'$ into the set of V^j. Let count = count + md.

End if

If there exist many nearest neighbors $\mathbf{x}'$ in $\mathbf{N}(\mathbf{x})$, select a solution that give the minimum $\hat{\mu}(x')$ to be $\mathbf{x}^j$ and add a new observation then calculate cumulative sample means as a new $\hat{\mu}(\mathbf{x}')$ then let $count = count + 1$.

Else % there exist only one $\mathbf{x}'$ in $\mathbf{N}(\mathbf{x})$

Use a generated new $\hat{\mu}(x')$ and collects $\mathbf{x}'$ into the set of V^j.

End if

End for

End if

End for

Fig. 4. NMSMN algorithm

Let V^j be a collection of visited solutions during the j^{th} iteration where $V^0 = \phi$. If $x \notin V^j$ then $V^j = V^{j-1} \cup x$. The NM is changed as follows: before generating new simulation outputs $G(x,\xi_i)$ for $x \in \{x_0, x_1, x_2, \ldots, x_d, x^R, x^E, x^C\}$, we check whether they have been visited. If $x \in V^j$, we use their past simulation outputs and generate only one new observation. The historical and one additional observations are used to calculate a new $\hat{\mu}(x)$ of a revisited solution. Otherwise, the NMSM generates new m observations for calculating new $\hat{\mu}(x)$ then registers x into the visited set.

2.6. The Nelder-Mead Selection with Memory and using Neighborhood (NMSMN)

The NMSMN is the NMSM integrated with a neighborhood. It constructs a neighborhood for every vertex of the simplex and estimates their objective function values. The best solution in the neighborhood replaces the original vertex. The advantages of the NMSMN are that they utilize past information of previous encounters and it also augments the search area to their neighborhood. Let $N(x^j)$ be a neighbor set of x^j with $\varepsilon = 0.01$ in Equation 7. Thus V^j is a collection of visited solutions and their neighborhood during the j^{th} iteration where $V^0 = \phi$. If $x \notin V^j$, $V^j = V^{j-1} \cup N(x^j)$. The NM are changed as follows: For $x \in \{x_0, x_1, x_2, \ldots, x_d, x^R, x^E, x^C\}$, before generating new simulation outputs, we check whether they are already visited solutions by the algorithm in **Fig. 4**.

2.7. The Adaptive Nelder-Mead Selection with Memory (ANSM)

The ANSM is NMSM, but when search reaches the step of updating the simplex, the $\hat{\mu}^j(x_1)$, $\hat{\mu}^j(x_2)$, …, $\hat{\mu}^j(x_{d+1})$ are estimated by the NMSM where Step 4 is modified as follows:

Step 4: Re-Order the Simplex Points In ascending orders then compare all updated expected objective functions of simplex points for determining the m^{j+1} by Equation 6. Let j = j +1.

2.8. The Adaptive Nelder-Mead Selection with Memory and using Neighborhood (ANSMN)

This modification applies ANSM, to the NMSMN. It combines the advantage of utilizing memory of revisited solution and their neighbors and efficiently spending resources to ensure further progress approaching to minimum objective function value.

2.9. Numerical Experiments

In section 2.3.1, we describe a set of test functions and their starting solutions. Section 2.3.2 explains the main figures of merit that we use to evaluate and compare the performance of many modifications of the NM. Section 2.3.3 discusses the empirical test setup.

2.10. Test Functions

We test the unconstrained optimization algorithm on a set of six deterministic test functions; that is:

$$G(x,\xi_x) = g(x) + \xi_x, \tag{8}$$

As ξ_x is an unbiased random element with $E[\xi_x] = 0$ and $V[\xi_x] = \sigma_x^2$ and g(x) is deterministic test problems. These test functions are 2 dimensional (d = 2). Our standard deviations, σ_x, are 0.75, 1.00 and 1.25 times g(x*) as defined in Equation 8. Common random numbers are used. Some of the selected functions have appeared in previous studies. For example, test functions 2-5 are classical test functions produced by More *et al.* (1981). They were also used in Humphrey and Wilson (2000) and Barton and Ivey (1996) for optimization of noisy responses. Test function 6 is adapted from Neddermeijer *et al.* (2000). Each of these deterministic test functions has a unique optimum.

2.11. Test Function 1: Paraboloid Function

The paraboloid function is defined as:

$$g(x) = \sum_{i=1}^{d} x_i^2 + 1$$

The starting point is given by x = [d,d,…,d]. The optimal of function value g* =1 is achieved at point x* = [0,..,0]. **Figure 5** depicts the polynomial function for case d = 2. This function is concave, symmetric and having only one minimum point. It is easy to optimize if no noise exists. However, when noise is present, optimization is difficult.

2.12. Test Function 2: Variably Dimensioned Function

The variably dimensioned function is given by:

$$g(x) = \sum_{i=1}^{d+2} [f_i(x)]^2 + 1$$

where $f_i(x) = x_i - 1$ for i = 1,…,d, $f_{d+1}(x) = \sum_{j=1}^{d} j(x_j - 1)$ and $f_{d+2}(x) = \left[\sum_{j=1}^{d} j(x_j - 1)\right]^2$. The starting point is given by x = $[x_1, x_2, \ldots, x_d]$, where $x_j = 1-(j/d)$, j = 1,2,…,d. The optimal function value g* = 1 is achieved at point x* = [1,…,1]. **Figure 6** depicts the variably dimensioned function for d = 2. The search area is U-curve, which is a crossed flat area. There are numerous local minima in the region of flat area but only one unique global minima exist.

2.13. Test Function 3: Trigonometric Function

The trigonometric function is defined as:

$$g(x) = \sum_{i=1}^{d} [f_i(x)]^2 + 1$$

where for i=1,...,d, $f_i(x) = d - \sum_{j=1}^{d} \cos(x_j - 1) + i[1 - \cos(x_i - 1)]$ $-\sin(x_i-1)$. The starting point is x = [1/d,…,1/d]. The optimal of function value g* = 1 is achieved at point x* = $[1+2\pi k_1,..,2\pi k_d]$ where k_j = 0±1, ±2,… for j = 1,..,d. **Figure 7** illustrates the trigonometric function for d = 2. This function is a sine curve and multi-modal minima.

2.14. Test Function 4: Extended Rosenbrock

The extended Rosenbrock function is defined as:

$$g(x) = \sum_{i=1}^{d} [f_i(x)]^2 + 1$$

where, for i=1,..,d/2, $f_{2i-1}(x) = 10(x_{2i} - x_{2i-1}^2)$ and $f_{2i}(x) = (1-x_{2i-1})$. The starting point is x = [-1.2,1,…, -1.2,1]. The optimal of function value g* = 1 occurs at x* = [1,…,1].

Figure 8 depicts the extended Rosenbrock function for the case of d = 2. This function is a non-convex function. The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial. However, it is difficult to converge to the global minimum.

2.15. Test Function 5: Brown's Almost-Linear Function

The Brown's almost-linear function is given by:

$$g(x) = \sum_{i=1}^{d} [f_i(x)]^2 + 1$$

where $f_i(x) = x_i + \sum_{j=1}^{d} x_j - (d+1)$ for i = 1,...,d-1 and $f_d(x) = \left(\prod_{j=1}^{d} x_j\right) - 1$. The solution x = [1/2,…,1/2] is used as the starting point. The optimal function value g* = 1 is achieved at the point x* = $[\lambda,..,\lambda,\lambda^{1-d}]$ where λ satisfies $d\lambda^d-(d+1)\ \lambda^{d-1}+1 = 0$. Humphrey and Wilson (2000) compute the value of λ is 0.5 for d = 2. **Figure 9** illustrates the Brown's almost-linear function for d = 2. The function is not linearly separable and has the basic form of a nonlinear least squares problem.

2.16. Test Function 6: Symmetrical Gaussian Function

The symmetrical Gaussian function is defined as:

$$g(x) = 2 - \exp\left\{-\frac{1}{15000}\sum_{i=1}^{d} [f_i(x)]^2\right\}$$

where, $f_i(x) = 100-x_i$ for i = 1,..,d. The starting point is x = [70,…,70]. The optimal of function value g* = 1 is achieved at the point x* = [100,..,100]. **Figure 10** depicts the symmetrical Gaussian function for d = 2. If any starting point is in area of blended curve, it converges to a unique global minimum point. On the other hand, if any staring point is in flat area, it is difficult to reach the minimum point.

2.17. Search Performance Measures

When the search terminates, optimal solutions can be estimated in at least 3 ways: The solution on-hand, the most frequently visited solution, or the solution with the best cumulative averages (Banks, 1998; Andradottir, 1999). Our preliminary experiments find that the solutions on hand outperform other estimates for optimal solutions. Motivated by Humphrey and Wilson (2000), we evaluate the search performance via the average of the following performance measures over many replications:

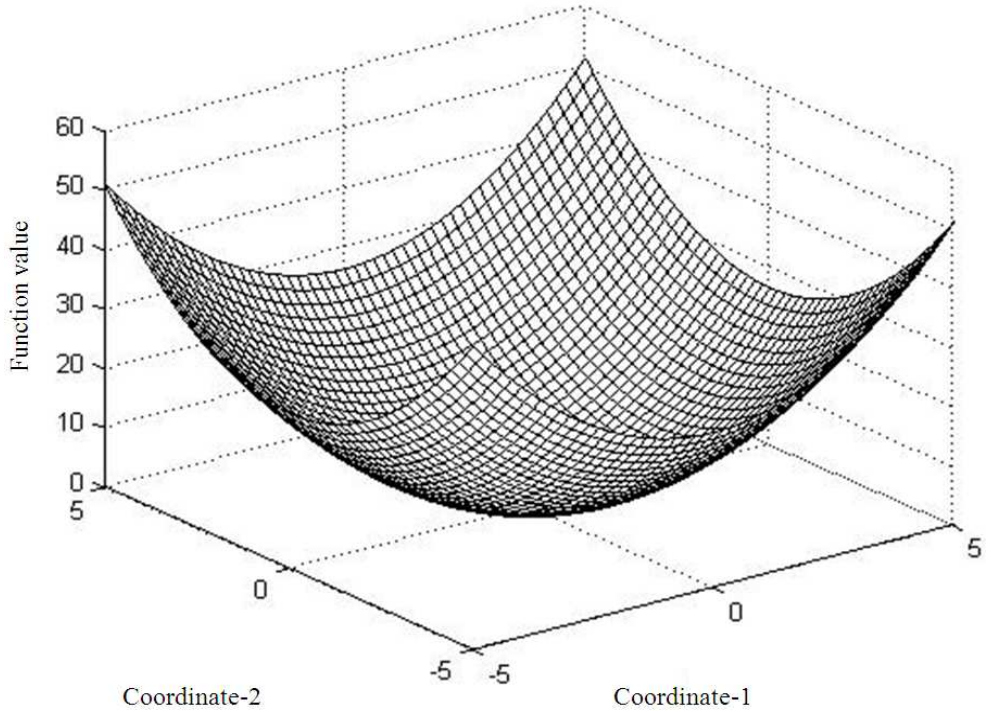

Fig. 5. Paraboloid function for d = 2

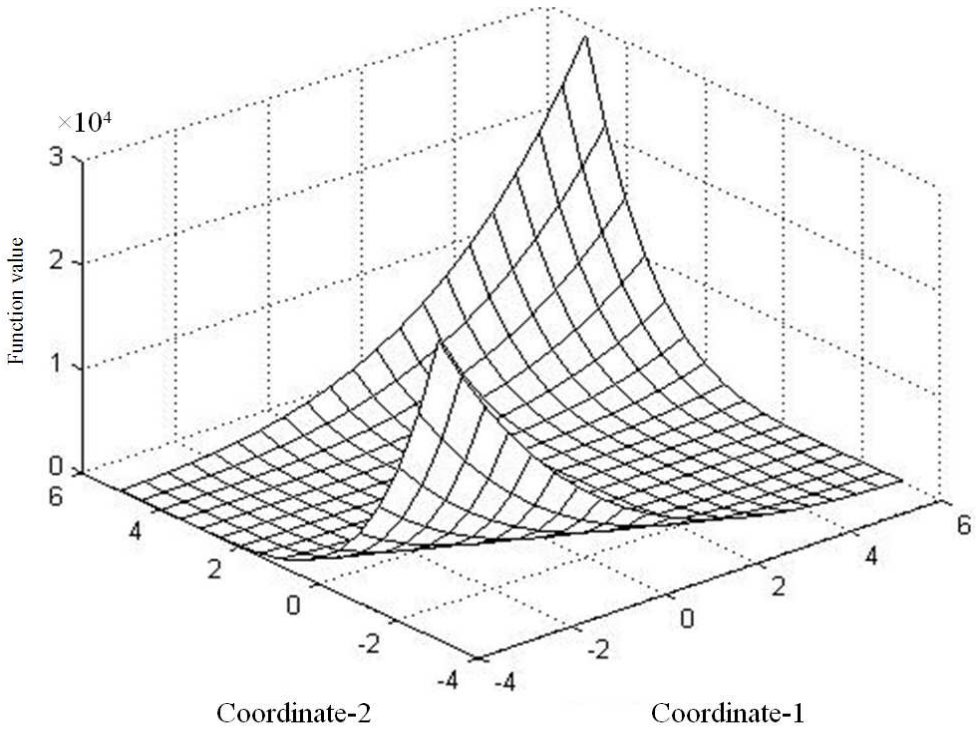

Fig. 6. Variably dimensioned function for d = 2

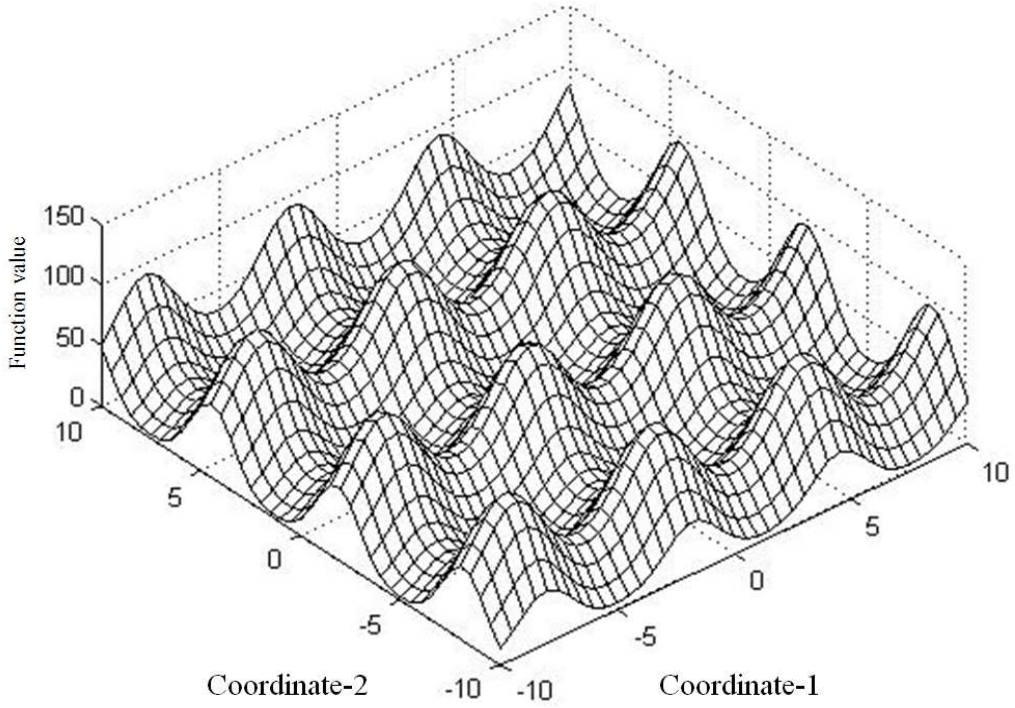

Fig. 7. Trigonometric function for d = 2

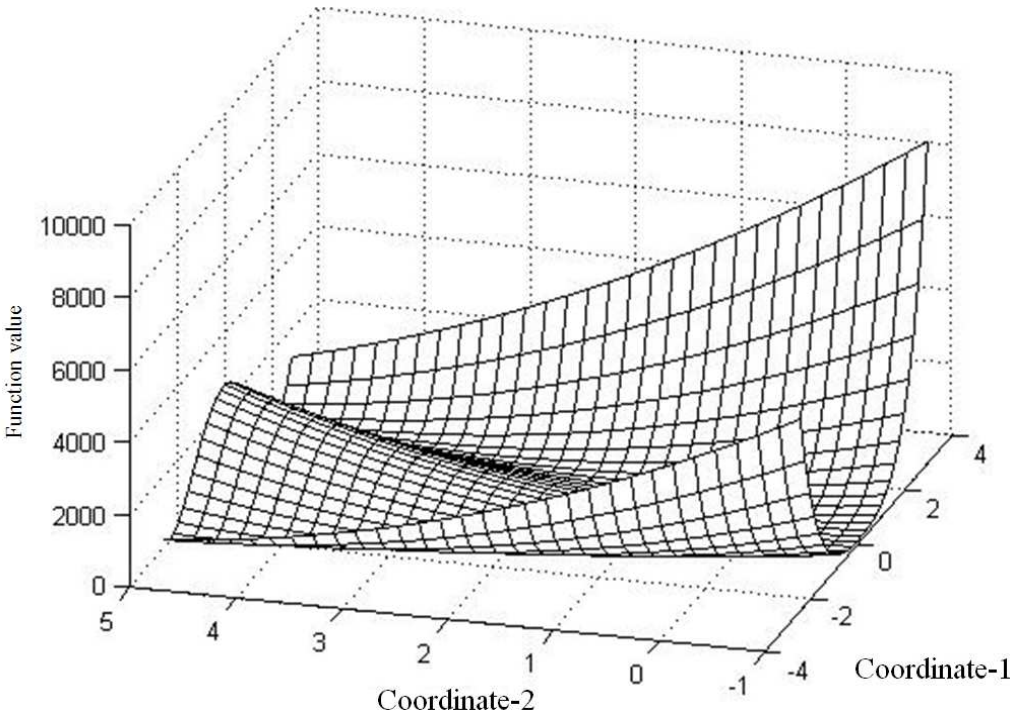

Fig. 8. Extended rosenbrock function for d = 2

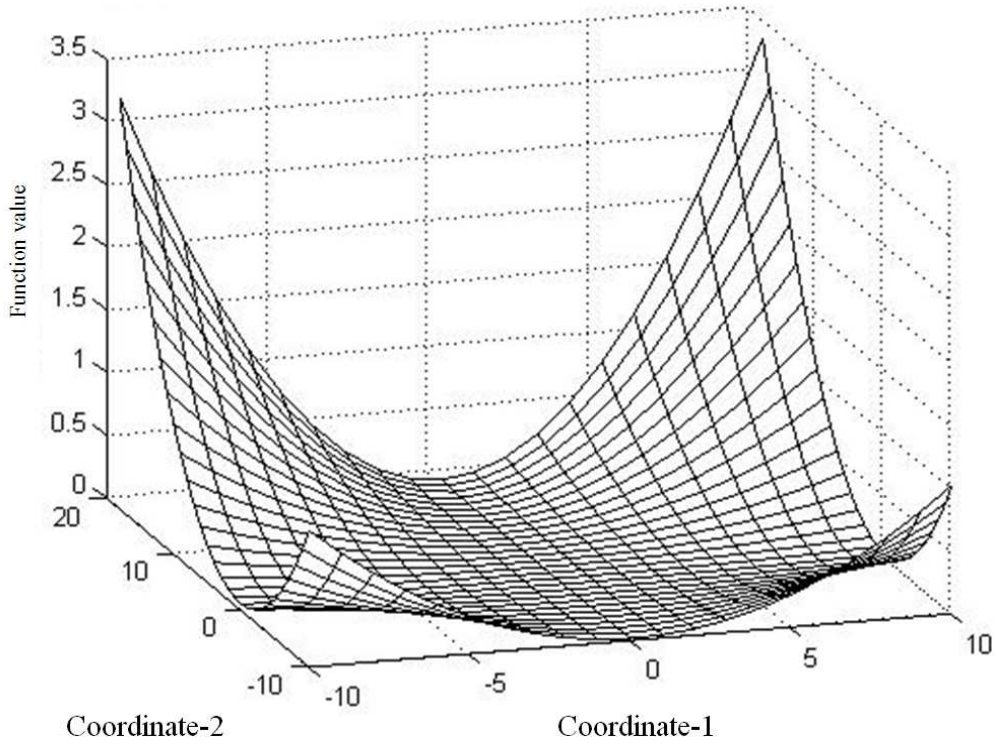

Fig. 9. Brown's almost-linear function for d = 2

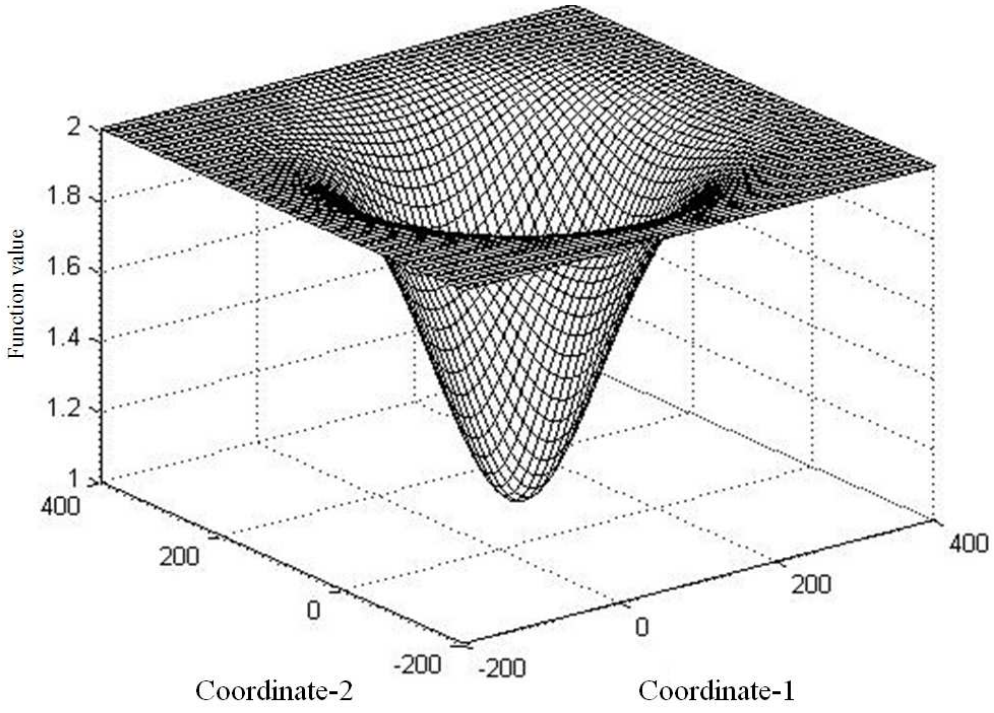

Fig. 10. Symmetrical gaussian function for d = 2

2.18. Logarithm of the Total Number of Simulation Outputs

To measure the computation work performed by a simulation optimization procedure, we compute:

$$L \equiv \ln(\text{total number of simulation outputs})$$

It provides at best a rough indication of the total computational work required by a simulation procedure.

2.19. Deviation of the Best Estimated Optimal Function Value from the True Optimal Value

For a measure of accuracy of the best result delivered by a simulation optimization procedure, we consider Equation 9:

$$D = \left| \frac{\mu(\hat{x}^*) - \mu(x^*)}{\mu(x^*)} \right| \quad (9)$$

This measure cannot be employed for Test Function 1 since each coordinate of the true optimum for the paraboloid function is equal to zero.

2.20. Empirical Test Setup

Our implementations are run on MATLAB by modifying fminsearch function. The NM coefficients are as follow: $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$ and $\tau = 0.5$. The initial step size, s_i as shown in Equation 3, is 10^{-4}. Minimum deviation ε_x and $\varepsilon_{\hat{\mu}}$ are 10^{-4}. Maximum budget consumption $N_{\hat{\mu}}$ and N_{search} are 10^5. To estimate the objective function, the sample size m is 6. Maximum neighborhood distance ε is 0.01. Three level of standard deviation of random noise ξ_x is {0.75 g (x*), 1.00g (x*), 1.25 g (x*)}. The factor of increasing simulation size b in Tomick (1995) is 1.25. We perform 20 macroreplications (i.e., experiments) for each test problem on 10 search algorithms as follows:

- NM-The original Nelder-Mead Simplex
- RS9-The Barton and Ivey stochastic modification
- ANRS9-Adaptive Nelder-Mead modification
- NMSM-The Nelder-Mead selection with memory
- NMSM+RS9-The NMSM with RS9
- NMSMN-The NMSM with using neighborhood
- NMSMN+RS9-The NMSMN with RS9
- ANSM-Adaptive Nelder-Mead selection with memory
- ANSM+RS9-ANSM with RS9
- ANSMMN-ANSM with using neighborhood
- ANSMMN+RS9-ANSMMN with RS9

3. RESULTS

Table 1 shows the budget consumption of each algorithm until computation budget is exhausted or until the search is unable to get any improvements. **Table 2** contains the average deviation in estimating the objective function values as defined in (9). As expected, when the degree of randomness increases, a given test problem becomes more difficult. Most algorithms fare worse and their estimated optimal solutions are further away from the optimal solutions because all algorithms give smaller $\bar{D}$ at lower random noise. That means $\mu(\hat{x}^*)$ is not far from μ (x*). On the contrary, Test Function 4 is the most difficult to optimize even when the level of randomness is low.

Moreover, using the adaptive strategy with memory gives the smallest $\bar{D}$ although the adaptive feature requires more computational effort. For example, for Test Function 1 and at all random noise levels, ANRS9 consumes more computational resource than RS9, but it rewards with better estimated optimal solutions, i.e., smaller $\bar{D}$. Similar results can be observed between ANSM+RS9 and ANSMN+RS9 in comparison with NMSM and NMSMN, respectively. Considering $\bar{D}$, no algorithms decisively wins at all noise levels, but almost wining algorithms involve the adaptive method. Similarly, when we consider L, no algorithms outperforms completely at all noise levels.

For better comparison, we compare relative ratio of both, L and $\bar{D}$, between pairs of algorithms. The data is divided into 3 sets. Firstly, comparing between the adaptive and non-adaptive methods, e.g., ANRS9/RS9, ANSM+RS9/NMSM +RS9 and ANSMN+RS9/NMSMN, for most test functions and almost all noise levels, the adaptive methods spends 79% more computation effort than the non-adaptive methods, but they give better estimates of the optimal solutions by reducing $\bar{D}$, for up to 50%. For example at σ_x =1.00 g(x*) for Test Function 2, all adaptive algorithms, ANRS9, ANSM+RS9 and ANSMN+ RS9, yield the estimates of the optimal solutions closer to the true optimum than the non-adaptive algorithms, RS9, NMSM+RS9 and NMSMN+RS9 respectively; $\bar{D}$, is reduced approximately by 30% although they spend more computational effort. Except for σ_x = 0.75g(x*) of Test Function 3, the adaptive algorithms with using memory (e.g., ANSM+RS9 and ANSMN+RS9) are slightly less than the corresponding the non-adaptive algorithms (i.e., NMSM+RS9 and NMSMN+RS9, respectively); and for Test Function 4 with σ_x = {0.75 g(x*), 1.00g (x*)}, the adaptive algorithms with using memory (e.g., ANSM+RS9 and ANSMN+RS9) the true optimum, i.e., $\bar{D}$ increases. For Test Functions 1, 3, 5 and 6, $\bar{D}$ increases when the standard deviation of random noise goes up, across all algorithms. For Test Function 2, almost every algorithms also exhibit this pattern except ANSMRS9. Moreover, the results show that for Test Functions 1-3 and 5-6, it is not difficult to find do not give better improvement than their counterparts.

Table 1. Logarithm of computational effort (L)

Factor		Test function					
σ_x	Algorithm	1	2	3	4	5	6
0.75	NM	2.58	2.44	2.46	2.52	2.43	2.42
	RS9	3.75	3.78	3.78	3.73	3.74	3.78
	NMSM	2.38	2.36	2.38	2.32	2.40	2.41
	NMSM+RS9	2.70	2.70	2.64	2.62	2.62	2.52
	NMSMN	2.42	2.41	2.38	2.48	2.32	2.41
	NMSMN+RS9	2.81	2.84	2.80	2.92	2.76	2.62
	ANRS9	4.61	4.48	4.57	4.66	4.62	4.69
	ANSM+RS9	3.08	3.21	2.45	2.89	3.05	2.64
	ANSMN+RS9	3.06	3.26	2.46	3.41	3.22	2.64
1.00	NM	2.55	2.44	2.42	2.55	2.45	2.44
	RS9	3.78	3.78	3.72	3.77	3.70	3.78
	NMSM	2.42	2.29	2.39	2.31	2.36	2.30
	NMSM+RS9	2.71	2.63	2.69	2.60	2.65	2.64
	NMSMN	2.44	2.42	2.37	2.35	2.40	2.30
	NMSMN+RS9	2.65	2.79	2.74	2.86	2.60	2.67
	ANRS9	4.46	4.56	4.68	4.59	4.63	4.70
	ANSM+RS9	3.18	3.18	2.98	3.07	3.05	2.95
	ANSMN+RS9	3.10	3.22	3.13	3.13	3.36	2.95
1.25	NM	2.55	2.42	2.42	2.53	2.47	2.42
	RS9	3.75	3.75	3.75	3.73	3.78	3.78
	NMSM	2.45	2.33	2.34	2.33	2.22	2.28
	NMSM+RS9	2.67	2.52	2.53	2.69	2.60	2.48
	NMSMN	2.45	2.40	2.32	2.51	2.36	2.28
	NMSMN+RS9	2.79	2.82	2.82	2.77	2.69	2.48
	ANRS9	4.65	4.48	4.69	4.59	4.64	4.62
	ANSM+RS9	3.26	3.09	2.97	3.11	2.91	3.19
	ANSMN+RS9	3.31	3.41	3.07	3.41	2.97	3.19

Secondly, comparing between memory and non-memory methods, e.g., NMSM/NM, NMSM+RS9/RS9 and ANSM+RS9/ANRS9, the results show that memory deployment saves on resource consumption, spending on average 77% less than non-memory counterparts for all test functions and noise levels, e.g., NMSM+RS9 and ANSM+RS9 consume less resources for about 30%, than RS9 and ANRS9, respectively. This is because solutions that are less than ε_x apart are classified to be the same. If the search revisits the already seen parts of the solution space, it may use the sampling data from the previous visits, instead of resampling anew. For most test functions and noise levels, on average, utilizing memory reduces deviation $\bar{D}$ by up to 80% of non-utilizing memory, aside from Test Function 4 that involves the adaptive methods. For example, Test Function 5 for all noise levels, NMSM, NMSM+RS9 and ANSMN+RS9 give better optimal solutions by up to 6% to 49% of NM, RS9 and ANRS9, respectively.

The rest of comparing is between neighbor and non-neighbor methods, e.g., NMSMN/NMSM, NMSMN+RS9 /NMSM+RS9 and ANSMN+RS9/ANSM+RS9. Regarding the memory-utilizing property, for all noise levels and almost all test functions, except Test Function 4, the non-adaptive methods which incorporate the neighbor-structure neither saves computation effort nor improves estimates of the optimal solution, e.g., on Test Functions 2 and 6, NMSMN and NMSMN+ RS9 give indistinguishable results on $\bar{D}$ and L from NMSM and NMSM+RS9, respectively. In other words, using neighbor-structure is greedy and misled to a non-optimum compared to the algorithms without neighbor-structure. On the other hand, ANSMN+RS9 provides a better optimal solution and spends less computation effort than ANSM+RS9 by 32% and 29%, respectively. These results show that using neighbor-structure on adaptive algorithms provide improved estimated optimal solutions. The search with good performance when there is no limitation on computational resource is AMSMN+ RS9 because it gives the least $\bar{D}$ at all noise levels for most test functions, except Test Function 4. If computational computational resource is limited, NMSM+RS9 performs better.

Table 2. Average deviation of function value of algorithm from true function value ($\bar{D}$)

Factor		Test function					
σ_x	Algorithm	1	2	3	4	5	6
0.75	NM	0.61	0.57	0.62	0.21	0.58	0.59
	RS9	0.92	0.95	0.97	0.33	0.94	0.92
	NMSM	0.45	0.54	0.61	0.15	0.54	0.54
	NMSM+RS9	0.69	0.70	0.73	0.17	0.68	0.63
	NMSMN	0.53	0.53	0.65	0.14	0.49	0.54
	NMSMN+RS9	0.70	0.72	0.77	0.27	0.66	0.72
	ANRS9	0.32	0.34	0.35	0.26	0.31	0.39
	ANSM+RS9	0.22	0.15	0.26	0.43	0.23	0.33
	ANSMN+RS9	0.20	0.23	0.26	0.38	0.24	0.33
1.00	NM	0.76	0.73	0.81	0.26	0.85	0.81
	RS9	1.22	1.26	1.29	0.56	1.26	1.22
	NMSM	0.69	0.70	0.80	0.21	0.78	0.75
	NMSM+RS9	0.92	0.92	0.94	0.23	0.89	0.99
	NMSMN	0.69	0.67	0.78	0.22	0.74	0.75
	NMSMN+RS9	0.85	0.92	0.96	0.36	0.88	0.87
	ANRS9	0.39	0.40	0.53	0.30	0.47	0.46
	ANSM+RS9	0.27	0.29	0.37	0.39	0.29	0.35
	ANSMN+RS9	0.29	0.28	0.34	0.35	0.27	0.35
1.25	NM	0.98	0.82	0.99	0.39	1.05	1.00
	RS9	1.53	1.60	1.62	0.88	1.62	1.57
	NMSM	0.98	0.96	0.85	0.23	0.84	0.92
	NMSM+RS9	1.05	1.06	1.12	0.56	1.14	1.12
	NMSMN	0.86	0.95	1.06	0.39	0.89	0.92
	NMSMN+RS9	1.25	1.15	1.28	0.47	1.26	1.12
	ANRS9	0.54	0.62	0.63	0.19	0.64	0.62
	ANSM+RS9	0.35	0.28	0.50	0.33	0.33	0.42
	ANSMN+RS9	0.37	0.37	0.50	0.32	0.34	0.42

4. DISCUSSION

We show that the Nelder-Mead algorithm which is designed for deterministic optimization can be modified to accommodate stochastic outputs. Using past information generally decreases computational effort and not jeopardizes the performance significantly. Although all adaptive-with-memory algorithms spends more computational resources, they give better optimal solutions comparing with their corresponding non-adaptive ones. Exploiting neighborhood and utilizing memory are helpful for adaptive algorithms, but not for non-adaptive algorithms. The search performance is improved on either ANSM+RS9 and ANSMN+RS9 for unconstrained resource consumption, or NMSM and NMSMN for economical resource consumption.

5. CONCLUSION

Utilizing past information in continouse optimization saves computational resource. To improve an estimated optimal solutions without limitation of computational effort, uses past information corporates on adaptive method. For our future work, we extend our algorithm to discrete search space and apply simulation to decision making such as queuing or inventory problem.

6. ACKNOWLEDGEMENT

This study was supported by the Industrial Engineering Department, Faculty of Engineering and the Graduate School of Kasetsart University, Thailand. The authors would like to thank Dr. Walailuck Chavanasporn, Mathematics department, King Mongkut University of Technology North Bangkok, for her editorial comments. Some parts of this work were presented previously at the Asia Simulation Conference 2011 in Seoul, Korea (Tippayawannakorn and Pichitlamken, 2012).

7. REFERENCES

Alon, R., S.W. Feigelson, E. Manevich, D.M. Rose and J. Schmitz *et al*., 2005. Alpha4beta1-dependent adhesion strengthening under mechanical strain is regulated by paxillin association with the alpha4-cytoplasmic domain. J. Cell. Biol., 171: 1073-1084. PMID: 16365170

Alrefaei, M.H. and S. Andradottir, 2005. Discrete stochastic optimization using variants of the stochastic ruler method. Naval Res. Logistics, 52: 344-360. DOI: 10.1002/nav.20080

Andradottir, S., 1999. Accelerating the convergence of random search methods for discrete stochastic optimization. Assoc. Comput. Mach. Trans. Model. Comput. Simulat., 9: 349-380. DOI: 10.1145/352222.352225

Andradottir, S., 2006. Chapter 20 An Overview of Simulation Optimization via Random Search. In: Handbooks in Operations Research and Management Science, Henderson, G.S. and B.L. Nelson (Eds.), pp: 617-631.

Banks, J., 1998. Chapter 9 simulation optimization. Principles, Methodol. Adv. Applic. Practice,

Barton, R.R. and J.S. Ivey, 1996. Nelder-Mead simplex modifications for simulation optimization. Manage. Sci., 42: 954-973. DOI: 10.1287/mnsc.42.7.954

Beyer, H.G. and H.P. Schwefel. 2002. Evolution strategies-A comprehensive introduction. J. Natural Comp., 1: 3-52. DOI: 10.1023/A:1015059928466

Carson, Y. and A. Maria, 1997. Simulation optimization: Methods and applications. Proceedings of the 29th Conference on Winter Simulation, Dec. 07-10, IEEE Computer Society Washington, DC, USA., pp: 118-126. DOI: 10.1145/268437.268460

Dennis, J.E. and D.J. Woods, 1987. Optimization on Microcomputers: The Nelder-Mead Simplex Algorithm. In: Society for Industrial and Applied Mathematics, Wouk, A. (Ed.), pp: 116-122.

Henderson, S.G. and B.L. Nelson, 2006. Simulation. 1st Edn., Springer, Dordrecht, pp: 693.

Holland, J., 2000. Building blocks, cohort genetic algorithms, and hyperplane-defined functions. Evolut. Comput., 8: 373-391. DOI: 10.1162/106365600568220

Humphrey, D.G. and J.R. Wilson, 2000. A revised simplex search procedure for stochastic simulation response surface optimization. INFORMS J. Comput. Fall, 12: 272-283. DOI: 10.1287/ijoc.12.4.272.11879

More, J.J., B.S. Garbow and K.E. Hillstrom, 1981. Testing unconstrained optimization software. ACM Trans. Mathemat. Software, 7: 17-41. DOI: 10.1145/355934.355936

Neddermeijer, H.G., G.J. Oortmarssen, N. Piersma, R. Dekker and J.D.F. Habbema, 2000. Adaptive extensions of the nelder and mead simplex method for optimization of stochastic simulation models. Econometric Institute Report, Erasmus University Rotterdam, Econometric Institute.

Nelder, J.A. and R. Mead, 1965. A simplex method for function minimization. Comput. J., 7: 308-313. DOI: 10.1093/comjnl/7.4.308

Olafsson, S. and J. Kim, 2002. Simulation optimization: Simulation optimization. Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers, (WSC '02), ACM Press, pp: 79-84.

Pichitlamken, J. and B.L. Nelson, 2003. A combined procedure for optimization via simulation. ACM Trans. Model. Comput. Simulat., 13: 155-179. DOI: 10.1145/858481.858485

Pichitlamken, J., B.L. Nelson and L.J. Hong, 2006. A sequential procedure for neighborhood selection-of-the-best in optimization via simulation. Eur. J. Operat. Res., 173: 283-298. DOI: 10.1016/j.ejor.2004.12.010

Press, W.H., S.A. Teukolsky, W.T., Vetterling and B.P. Flannery, 2007. Section 10.12. Simulated Annealing Methods In Numerical Recipes: The Art of Scientific Computing. 3rd Edn., Cambridge University Press, New York, ISBN-10: 0521880688 pp: 549-555.

Shi, L. and S. Olafsson, 2009. Nested Partitions Optimization. In: Encyclopedia of Optimization, Christodoulos, A.F. and P.M., Panos, (Eds.), pp: 2533-2539.

Spendley, W., G.R. Hext and F.R. Himsworth, 1962. Sequential application of simplex designs in optimisation and evolutionary operation. Technometrics, 4: 441-461.

Swisher, J.R., P.D. Hyden, S.H. Jacobson and L.W. Schruben, 2004. A survey of recent advances in discrete input parameter discrete-event simulation optimization. IIE Trans., 36: 591-600. DOI: 10.1080/07408170490438726

Tippayawannakorn, N. and J. Pichitlamken, 2012 Nelder-mead method with local selection using neighbor hood and memory for optimization via simulation. Proceedings of the Advanced Methods, Techniques and Applications in Modeling and Simulation, (AMTAMS' 12), pp: 134-143.

Tomick, J.J., 1995. On convergence of the nelder-mead simplex algorithm for unconstrained stochastic optimization. Pennsylvania State University.

Tomick, J.J., S.F. Arnold and R.R. Barton, 1995. Sample size selection for improved Nelder-Mead performance. Proceedings of Winter Simulation Conference, Dec. 3-6, IEEE Xplore Press, Arlington, VA., pp: 341-345. DOI: 10.1109/WSC.1995.478754