



Thailand Statistician
January 2016; 14(1): 63-81
<http://statassoc.or.th>
Contributed paper

Nelder–Mead Method with Local Selection Using Memory for Discrete Stochastic Optimization

Noocharin Tippayawannakorn* and Jutta Pichitlamken

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

*Corresponding author; e-mail: noocharint@kmutnb.ac.th

Received: 8 June 2015

Accepted: 14 July 2015

Abstract

We consider the Nelder–Mead (NM) simplex algorithm for optimization of discrete–event stochastic simulation models. We propose seven new modified algorithms based on NM to reduce computational time and to improve quality of the estimated optimal solutions. Our modifications include utilizing past information of already seen solutions, using adaptive sample size, and using Restart. We compare the performance of these extensions on four test functions with 5 levels of random variations. We find that using past information leads to reduction of computational efforts by up to 50%. The adaptive modifications need fewer resources than the non-adaptive counterparts by up to 30% but there is no difference in quality. Although the Restart consumes 60% more resources than non–Restart, the Restart gives up to 50% improvement in the optimal solution. In summary, we find that a Restart with memory algorithm gives the best performance either with or without adaptivity.

Keywords: Nelder–Mead simplex, adaptive Nelder–Mead simplex, discrete stochastic optimization, local selection, memory utilization, restart.

1. Introduction

An Optimization via Simulation (OvS) is a problem of finding possible sets of input variables or decision variables that give maximum or minimum objective function values. In addition, OvS also aims at minimizing computational resources spent while maximizing the information obtained in a simulation experiment (Carson and Maria 1997). Most OvS problems in practice involve discrete decision variables with stochastic objective functions (see, for example, Swisher et al. 2004; Ólafsson and Kim 2002; Alon et al. 2005).

Optimization algorithms are important tools for effective use of stochastic simulation to improve real systems. An example of stochastic simulation that is considered in various industrial operations is blending of raw materials to make products (Kumral 2003). The Buffer Allocation Problem (BAP) is well-known in the design of production lines. The objective is to allocate a fixed number of buffer spaces, n , to the $m-1$ niches between successive m machines, in order to measure the performance of the system, such as steady-state throughput. There are two reasons why the BAP is a difficult optimization problem. The first reason is that, for a given buffer allocation, the exact value of an

objective function (for example, the steady-state throughput) is often difficult or impossible to calculate. In fact, complete knowledge of the objective function is only available for relatively small production lines in which the processing times have an exponential distribution. In a more general setting, the BAP is typically a noisy or simulation-based optimization problem, i.e., via discrete-event simulation. The second reason why the BAP is difficult is that finding the optimum of the objective function, even if this function is completely known, comprises a combinatorial optimization problem over a potentially every large set with $\binom{n+m-2}{m-1}$ elements (Alon et al. 2005).

Many OvS tools have been developed for unconstrained problems. Most of them are based on a random search method that takes the 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 (Andradóttir 2006). A point-based strategy involves sampling points in a neighborhood of the current solution, e.g., the Stochastic Ruler (Yan and Mukai 1992) and Simulated Annealing (Kirkpatrick et al. 1983). 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 Ólafsson 1997), Hooke-Jeeves pattern search (Hooke and Jeeves 1961) and the Nelder-Mead Simplex method (Nelder and Mead 1965). A population-based strategy creates a collection of candidate solutions using some properties of the previously visited solutions; for example, Genetic Algorithms (Holland 1992) and Evolutionary Strategies (Schwefel 1993).

In this paper, we focus on the Nelder-Mead (NM) simplex algorithm (Nelder and Mead 1965). This algorithm was originally developed for solving unconstrained deterministic optimization problems. It demonstrates wide versatility and ease of use. It is implemented in MATLAB as a function *fminsearch*. Since the NM is also robust with respect to small random variations in the observed objective function values, it is now widely used for optimizing stochastic problems (Tomick 1995; Humphrey and Wilson 2000). For example, it has been used to study a food reffluent treatment model for plant design (Bonello et al. 2002), a microsimulation model for breast cancer screening (Branco 2006), and a quality assurance program of manufactured parts (Gosavi and Phatakwal 2006). However, in the case that variability in the objective function values are sufficiently large, the NM may terminate before reaching the global optima. To overcome this problem, Barton and Ivey (1996) proposed the algorithm RS9 that improved NM performance for stochastic problems by increasing the shrink parameter and recalculating every point of the shrink simplex. Tippayawannakorn and Pichilamken (2013), have proposed variants of the NM for the continuous stochastic optimization by utilizing past information of already seen solutions, expanding search space to the neighborhood of already seen solutions and using adaptive sample sizes. Their results show that using past information reduces computational effort by up to 20%, that adaptive variants need more resources but give better-quality solutions. However, searching in the neighborhood of already seen solutions was found not to save computation effort and not to improve estimates of optimization.

In this work, we extend Tippayawannakorn and Pichilamken (2013) to discrete search spaces. We incorporate past information and adaptive sample size into the Nelder-Mead with discrete stochastic solution space. We also incorporate the use of restart to escape from getting stuck in local minima. With numerical experiments, we show that our algorithms provide better solutions while requiring less computational effort than the original NM.

Generally, the OvS problem can be defined as follows (Andradóttir 2006): The objective is to determine an optimal solution, $\mathbf{x}^*$, that minimizes an unknown objective function, $\mu: \Theta \rightarrow \mathbb{R}$ over a continuous feasible region, $\Theta \in \mathbb{R}^d$; that is,

$$\mathbf{x}^* = \arg \min_{\mathbf{x} \in \Theta} \mu(\mathbf{x}),$$

where $\mathbf{x} \in \Theta$ is a vector of d decision variables, and $\mathbf{x}$ is called a solution. The objective function $\mu(\mathbf{x})$ cannot be observed directly; thus, it is estimated with stochastic simulation, i.e.

$$\mu(\mathbf{x}) = E_{\xi_{\mathbf{x}}} [G(\mathbf{x}, \xi_{\mathbf{x}})], \quad (1)$$

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

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

This paper is organized as follows: Section 2 describes the original form of the NM simplex algorithm and its existing variants. Section 3 discusses our extensions. Section 4 consists of numerical experiments. Section 5 contains and discussion and conclusions.

2. The Nelder–Mead Simplex Algorithm

2.1. Original Nelder–Mead simplex (NM)

The simplex method for optimization of deterministic problems was first proposed by Spendley et al. (1962). 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) added 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. Flow charts for the Nelder–Mead method are shown in Figures 1 and 2. Figure 1 illustrates 2-dimensional trial points for a simplex consisting of $\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$. The solid lines show the initial simplex. Other line-style simplexes show various simplex operations, e.g., an expansion point is $\mathbf{x}_2^E$, a reflection point is $\mathbf{x}_2^R$, internal and external contraction points are $\mathbf{x}_2^{C_i}$ and $\mathbf{x}_2^{C_e}$, respectively, and C is the centroid of the 2 best points.

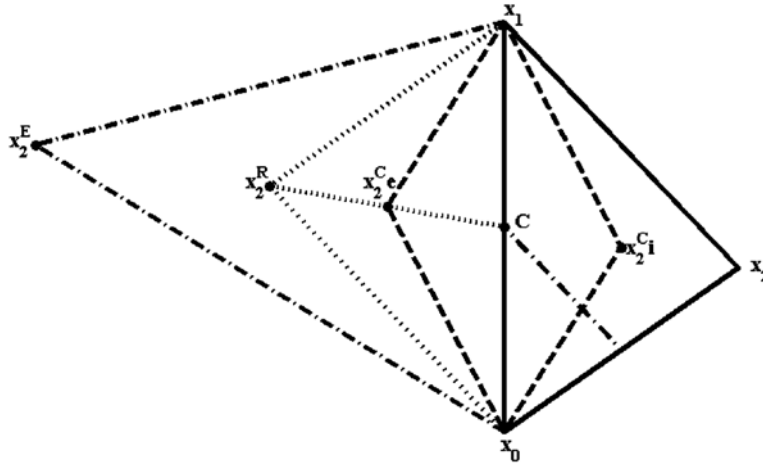


Figure 1 All simplex operations of the Nelder–Mead simplex

By the default setting of *fminsearch* (the NM implementation in MATLAB), a single simulation output ($m = 1$ in (2)) is an estimate of the objective function $\hat{\mu}(\mathbf{x})$.

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

Let $count = 0$.

Initialization: Create an initial simplex.

a. Select a starting point $\mathbf{x}_0 \in \Theta$, a vector of d dimensions. Then initialize the iteration number $j = 1$.

b. Form an initial simplex of $d + 1$ points, by defining

$$\mathbf{x}_i^j = [x_1, x_2, \dots, x_i + s_i, \dots, x_d], \quad i = 1, 2, \dots, d, \quad (3)$$

where s_i are the user-specified initial step sizes.

c. Estimate the objective function values at each of the $d + 1$ simplex points from m independent simulation outputs by computing sample averages via (2) to get $\hat{\mu}(\mathbf{x}_0^j)$,

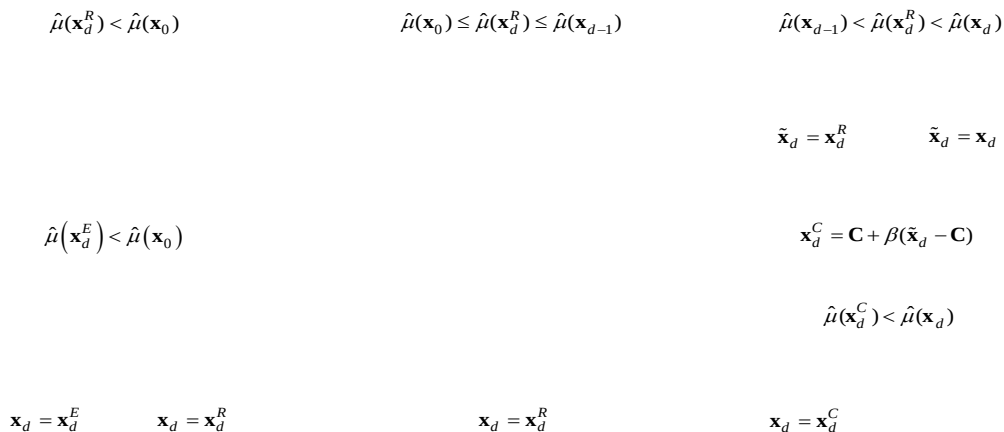


Figure 2 Flow chart for the j^{th} iteration of the Nelder–Mead simplex algorithm

$\hat{\mu}(\mathbf{x}_1^j), \dots, \hat{\mu}(\mathbf{x}_d^j)$. Then update the number of observations of simulation outputs,
 $count = count + m(d+1)$.

d. Re-order the simplex points in a non-decreasing order so that $\hat{\mu}(\mathbf{x}_0^j) \leq \hat{\mu}(\mathbf{x}_1^j) \leq \dots \leq \hat{\mu}(\mathbf{x}_d^j)$.

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

Step 1: Calculate the reflection point. Find the worst point on the simplex (recall that we consider a minimization problem, so the worst point is one with the highest sample mean $\mathbf{x}_d$). Replace the worst point $\mathbf{x}_d$ with another point which has a lower objective function value as follows. Let $\mathbf{x}_d^R$ be the reflection of the worst point in the centroid $\mathbf{C}$ of the other $d-1$ points. These centroid and reflected point are computed as

$$\mathbf{C} = \frac{1}{d} \sum_{k=0}^{d-1} \mathbf{x}_k,$$

$$\mathbf{x}_d^R = \mathbf{C} + \alpha(\mathbf{C} - \mathbf{x}_d),$$

where $\alpha > 0$ is a reflection parameter; typically 1. Then the objective function value $\hat{\mu}(\mathbf{x}_d^R)$ is estimated via outputs of m simulations. Then set $count = count + m$.

Step 2: Update the simplex. Figure 3 shows an initial simplex with dashed lines and an updated simplex with solid lines. The updated simplex depends on the relationship between $\hat{\mu}(\mathbf{x}_d^R)$ and $\hat{\mu}(\mathbf{x}_0^j), \hat{\mu}(\mathbf{x}_1^j), \dots, \hat{\mu}(\mathbf{x}_{d-1}^j)$; that is,

- If $\hat{\mu}(\mathbf{x}_0^j) \leq \hat{\mu}(\mathbf{x}_d^R) < \hat{\mu}(\mathbf{x}_{d-1}^j)$, set $\mathbf{x}_d^j \leftarrow \mathbf{x}_d^R$ and $\hat{\mu}(\mathbf{x}_d^j) \leftarrow \hat{\mu}(\mathbf{x}_d^R)$ as shown in Figure 3(a). Then go to Step 4.
- If $\hat{\mu}(\mathbf{x}_d^R) < \hat{\mu}(\mathbf{x}_0^j)$, the search continues in the same direction by calculating the expansion point, $\mathbf{x}_d^E = \mathbf{C} + \gamma(\mathbf{x}_d^R - \mathbf{C})$, where $\gamma > 0$ is an expansion parameter, typically 2. Then $\hat{\mu}(\mathbf{x}_d^E)$ is estimated from outputs of m simulations by (2), then set $count = count + m$. The expansion point is accepted if it improves over the best point in the simplex, $\mathbf{x}_0$, as shown in Figure 3(b); otherwise, the reflection point is accepted. Then; go to Step 4.
- If $\hat{\mu}(\mathbf{x}_{d-1}^j) \leq \hat{\mu}(\mathbf{x}_d^R)$, the search reduces the simplex size by calculating the contraction point,

$$\mathbf{x}_d^C = \mathbf{C} + \beta(\tilde{\mathbf{x}}_d - \mathbf{C}),$$

where $\beta > 0$ is a contract parameter (generally 0.5), and $\tilde{\mathbf{x}}_d$ is $\mathbf{x}_d^R$ if $\hat{\mu}(\mathbf{x}_d^R) < \hat{\mu}(\mathbf{x}_d^j)$ and $\mathbf{x}_d^j$ otherwise. The objective function $\hat{\mu}(\mathbf{x}_d^C)$ is estimated then $count = count + m$. If $\hat{\mu}(\mathbf{x}_d^C) \geq \hat{\mu}(\mathbf{x}_d^j)$, go to Step 3; otherwise, the contraction point is accepted, i.e.,

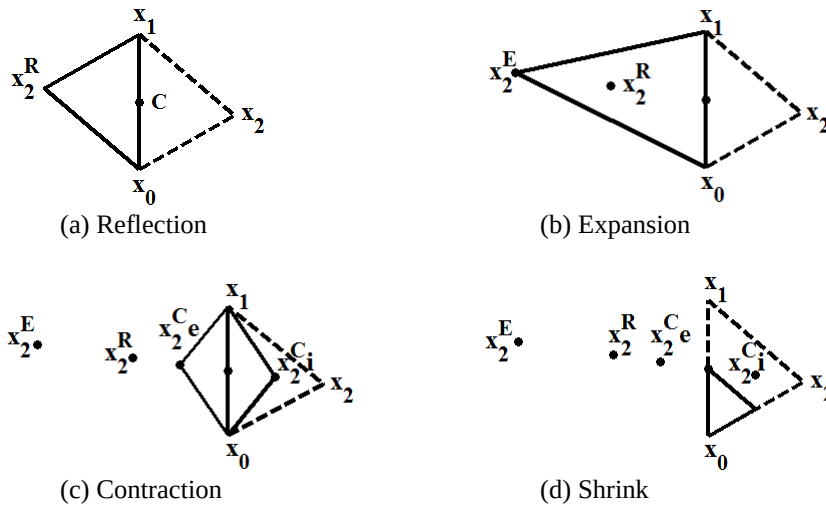


Figure 3 Operations of the NM simplex method

$\mathbf{x}_d^j \leftarrow \mathbf{x}_d^C$. The updated simplex can be one of the two solid-line simplexes in Figure 3(c) depending on which as the optimal $\hat{\mathbf{x}}^{*j}$ and $\hat{\mu}(\hat{\mathbf{x}}^{*j})$, respectively. When the search terminates, contraction point ($\mathbf{x}_2^{C_i}$ or $\mathbf{x}_2^{C_e}$) is used. Then; 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 $\mathbf{x}_0$ as shown in Figure 3(d). Compute the new simplex as follows:

$$[\mathbf{x}_0^j, \tau \mathbf{x}_1^j + (1-\tau)\mathbf{x}_0^j, \tau \mathbf{x}_2^j + (1-\tau)\mathbf{x}_0^j, \dots, \tau \mathbf{x}_d^j + (1-\tau)\mathbf{x}_0^j], \quad (4)$$

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 of $\hat{\mu}(\hat{\mathbf{x}})$. Then let $j = j + 1$.

End while

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

2.2. Barton and Ivey stochastic modification of NM and its variant (RS9 and ANRS9)

Barton and Ivey (1996) adapted the NM algorithm to accommodate stochastic variations in the objective function values. From empirical results, they found that the NM algorithm relies on the ranks of the objective function values at the simplex vertices. Therefore, it can make progress in the presence of relatively small randomness if this randomness does not change the rank of the function value at the simplex points. However, if the variations in the function value are too large, then the relative ranking of the simplex vertices could change and the algorithm could fail.

Barton and Ivey also recommended that a value of 0.9 for the shrinkage coefficient in Step 3 instead of the usual 0.5 could increase the rate of convergence of the algorithm. This change is also an effective method of improving the performance of the algorithm for cases where the original NM fails. Resampling the best point after the shrinking can reduce the frequency of contraction, but this

strategy was found to be not an effective method of improving algorithm performance.

In their algorithm, Barton and Ivey applied another stopping criterion for stochastic problems, as suggested by Dennis and Woods (1987). This criterion was that the search was terminated when the simplex size was sufficiently small, i.e., when

$$\frac{\sum_{i=1}^d \|\mathbf{x}_i - \mathbf{x}_0\|}{\max(1, \|\mathbf{x}_0\|)} \leq \varepsilon_x, \quad (5)$$

where $\hat{\mu}(\mathbf{x}_0) < \hat{\mu}(\mathbf{x}_1) < \dots < \hat{\mu}(\mathbf{x}_d)$, and $\|\cdot\|$ is the Euclidean norm $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2 + \dots + x_d^2}$. In addition to using the stopping criterion in (5), the so-called RS9 algorithm of Barton and Ivey included the following modifications to the NM algorithm: 1) the objective function was estimated with 6 independent simulation outputs ($m = 6$ in (2)) and 2) every solution was resampled every time it was encountered (no search history was 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) modified the RS9 further to allow the sample size to adjust adaptively to the observed noise in the solution space. Their modified algorithm is called ANRS9. Suppose that m^j is the minimum number of observations taken at each new trial point during the j^{th} iteration, and $m^0 = 6$:

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

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

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

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

2.3. The Nelder-Mead local selection with memory (NMSM) and its variant (ANSM)

Tippayawannakorn and Pichitlamken (2013) use past information 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.

An “already-visited solution” is defined as a new solution $\mathbf{x}$ that is not “too far” from a solution $\mathbf{v}$ that has already been seen, i.e. where

$$\|\mathbf{v} - \mathbf{x}\|_\infty = \max_{i=1,2,\dots,d} |\mathbf{v}_i - \mathbf{x}_i| = e_v,$$

where $\|\cdot\|_\infty$ is the uniform norm,

$$\|\mathbf{x}\|_\infty = \max \{|x_1|, |x_2|, \dots, |x_d|\},$$

and the neighbor distance $0 \leq e_v \leq \varepsilon_x$ where $\varepsilon_x = 10^{-4}$. This definition is similar to the terminating criterion used in the Matlab program *fminsearch*. We select the neighbor $\mathbf{v}$ which provides the minimum norm e_v from $\mathbf{x}$. For an example of a 2-dimensional problem, suppose $\mathbf{x} = (1.00018, 2.50101)$, $\mathbf{v}_1 = (1.00018, 2.50110)$ and $\mathbf{v}_2 = (1.00011, 2.50101)$, $\mathbf{v}_3 = (1.00015, 2.60101)$ then $e_1 = \max\{|1.00018 - 1.00018|, |2.50101 - 2.50110|\} = 9 \times 10^{-5}$, $e_2 = \max(|1.00018 - 1.00011|, |2.50101 - 2.50101|) = 7 \times 10^{-5}$, $e_3 = \max(|1.00018 - 1.00015|, |2.50101 - 2.60101|) = 0.1$. Since e_1 and e_2 are less than ε_x , $\mathbf{v}_1$ and $\mathbf{v}_2$ are in the neighborhood of $\mathbf{x}$, and we select $\mathbf{v}_2$ to represent $\mathbf{x}$ because e_2 has the smallest uniform-norm distance from $\mathbf{x}$.

The proposed NM-based algorithms with memory are as follows:

2.3.1. 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 an already-seen solution, NMSM computes one new simulation output and averages it with previous simulation outputs at that point. This procedure is used to protect the search from an unusually good or bad history.

Let V^j be a collection of visited solutions during the j^{th} iteration where $V^0 = \emptyset$. If $\mathbf{x} \notin V^j$, then $V^j = V^{j-1} \cup \mathbf{x}$. The NM is modified as follows: before generating new simulation outputs $G(\mathbf{x}, \xi_i)$ for $\mathbf{x} \in \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d, \mathbf{x}^R, \mathbf{x}^E, \mathbf{x}^C\}$, we check whether they have been visited. If $\mathbf{x} \in V^j$, we use their past simulation outputs and generate only one new observation. The historical and one additional observation are used to calculate a new $\hat{\mu}(\mathbf{x})$ of a revisited solution. Otherwise, the NMSM generates m new observations for calculating new $\hat{\mu}(\mathbf{x})$ then registers $\mathbf{x}$ into the visited set. The details of the NMSM algorithm are shown in Figure 4.

2.3.2. The adaptive Nelder-Mead selection with memory (ANSMRS9)

The ANSM+RS9 is NMSM+RS9, but when search reaches the step of updating the simplex in Step 4, the $\hat{\mu}^j(\mathbf{x}_1)$, $\hat{\mu}^j(\mathbf{x}_2)$, ..., $\hat{\mu}^j(\mathbf{x}_{d+1})$ are estimated by the NMSM with a modified Step 4 as follows:

For each $\mathbf{x} \in \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d, \mathbf{x}^R, \mathbf{x}^E, \mathbf{x}^C\}$

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

Add one new simulation output

and calculate cumulative sample means as a new $\hat{\mu}(\mathbf{x})$

and let $count = count + 1$.

End if

End for

Figure 4 NMSM algorithm

Step 4: Re-order the simplex points in ascending order then compare all updated expected objective functions of simplex points for determining the m^{j+1} by (6). Let $j = j + 1$.

3. New variants of the Nelder-Mead simplex algorithm

This work is extended from Tippayawannakorn and Pichilamken (2013) which apply existing variants for continuous problems. Moreover, we find that when the search gets stuck in any local optimum, it also consumes more computational resources. In this work, we suggest that a method to avoid getting stuck in the local optimum is to stop the optimization process when we find that the search stands still at the best point for an excessively long time. The algorithm is then restarted from a different initial simplex. We allow the restart to happen no more than 3 times, or a smaller number of times if the computational resources are limited. The aim of using the restart process is to try to find a better optimum than the local optima that have already been found.

The modified version of the NM algorithm that includes a Restart Search algorithm (RSS) is shown in Figure 5.

```

Let  $count = 0$  and  $restart\_index: \eta = 0$ .

While  $\eta < 3$  and  $count < N_{\hat{\mu}}$  are true,

    If  $\max_{k=0,1,2,\dots,9} \|\hat{\mathbf{x}}_0^{j-k} - \hat{\mathbf{x}}_0^j\|_2 \leq 10^{-8}$ 

        let  $\eta = \eta + 1$ ,

        then do NM-algorithm as show in Section 2.1

        until reach the stopping criteria. Then

        return  $\hat{\mathbf{x}}^*$  and  $\hat{\mu}(\hat{\mathbf{x}}^*)$  as  $\hat{\mathbf{x}}^{*\eta}$  and  $\hat{\mu}(\hat{\mathbf{x}}^{*\eta})$ .

    Else

        Return  $\min_{\eta} \hat{\mu}(\hat{\mathbf{x}}^{*\eta})$  and  $\hat{\mathbf{x}}^{*\eta}$ 

    End if

End while

```

Figure 5 RSS algorithm

4. Numerical Experiments

In Section 4.1, we describe a set of test functions and their starting solutions. Section 4.2 explains the main performance measure that we use to evaluate and compare the performance of our modifications of the NM. Section 4.3 discusses the empirical test setup.

4.1. Test functions

We test the unconstrained optimization algorithm on a set of four deterministic test functions with an added white-noise term; that is,

$$G(\mathbf{x}, \xi_{\mathbf{x}}) = g(\mathbf{x}) + \xi_{\mathbf{x}}, \quad (7)$$

where $g(\mathbf{x})$ is a deterministic test problem and $\xi_{\mathbf{x}}$ is an unbiased random element with $E[\xi_{\mathbf{x}}] = 0$ and $V[\xi_{\mathbf{x}}] = \sigma_{\mathbf{x}}^2$. The test functions can have from 1 to 15 dimensions (d). Our standard deviations,

σ_x , are 0.10, 0.15, 0.20, 0.25 and 0.3 times $g(x^*)$ as defined in (7). In the previous study (Tippayawannakorn and Pichitlamken 2013) used large values for σ_x , with even the smallest one being relatively large, $0.75 g(x^*)$. In this paper, we consider smaller values of σ_x in order to study the algorithm sensitivity. Common random numbers are used for each of our modified algorithms in order to make the comparison sharper. Some of the selected functions that we use have appeared in previous studies. For example, the Test Functions 2, 3 are classical ones created by Moré et al. (1981). They were also used by Humphrey and Wilson (2000) and Barton and Ivey (1996) for tests of their algorithms for optimization of noisy responses. Test Function 4 is adapted from Neddermeijer et al. (2000). Each of these deterministic test functions has a unique optimum.

4.1.1. Test function 1: Paraboloid function

The paraboloid function is defined as

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

The starting point is given by $\mathbf{x} = [d, d, \dots, d]$. The optimal function value $g^* = 1$ is achieved at point $\mathbf{x}^* = [0, \dots, 0]$. Figure 6 depicts the polynomial function for case $d = 2$. This function is a convex function and therefore has only one minimum point. It is easy to optimize if no noise exists. However, when noise is present, optimization can be difficult.

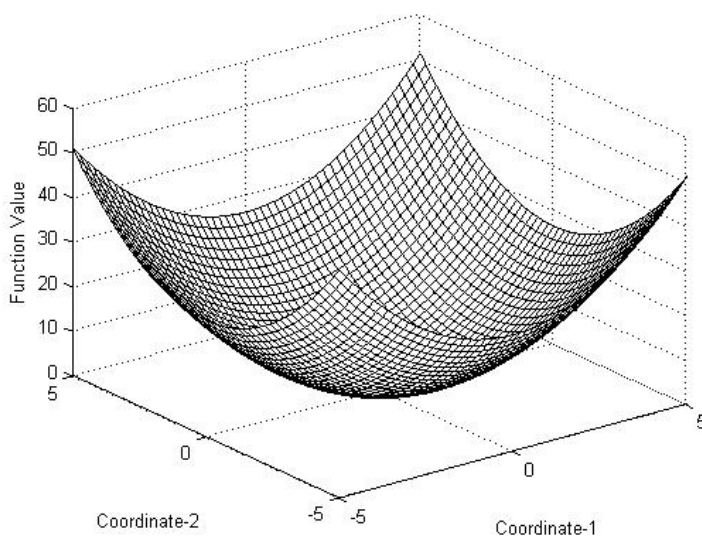


Figure 6 Paraboloid function for $d = 2$

4.1.2. Test function 2: Extended rosenbrock

The extended Rosenbrock function is defined as

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

where for $i = 1, \dots, d/2$, $f_{2i-1}(\mathbf{x}) = 10(x_{2i} - x_{2i-1}^2)$ and $f_{2i}(\mathbf{x}) = (1 - x_{2i-1})$. The starting point is $\mathbf{x} = [-1.2, 1, \dots, -1.2, 1]$. The optimal function value $g^* = 1$ occurs at $\mathbf{x}^* = [1, \dots, 1]$. Figure 7 depicts

the extended Rosenbrock function for the case of $d = 2$. This function is non-convex. The global minimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, but it is difficult for a Nelder-Mead method to converge to the global minimum.

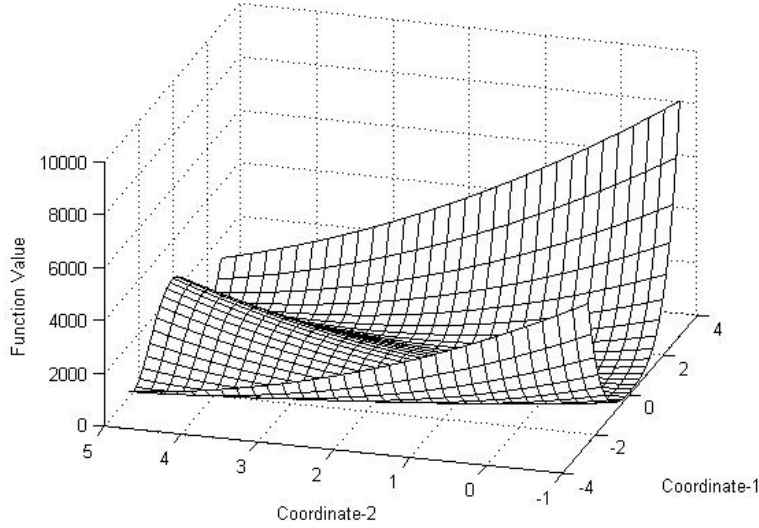


Figure 7 Extended Rosenbrock function for $d = 2$

4.1.3. Test Function 3: Variably Dimensioned Function

The variably dimensioned function is given by

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

where $f_i(\mathbf{x}) = x_i - 1$ for $i = 1, \dots, d$, $f_{d+1}(\mathbf{x}) = \sum_{j=1}^d j(x_j - 1)$, and $f_{d+2}(\mathbf{x}) = \left[\sum_{j=1}^d j(x_j - 1) \right]^2$. The starting point is given by $\mathbf{x} = [x_1, x_2, \dots, x_d]$, where $x_j = 1 - (j/d)$, $j = 1, 2, \dots, d$. The optimal function value $g^* = 1$ is achieved at point $\mathbf{x}^* = [1, \dots, 1]$. Figure 8 depicts the variably

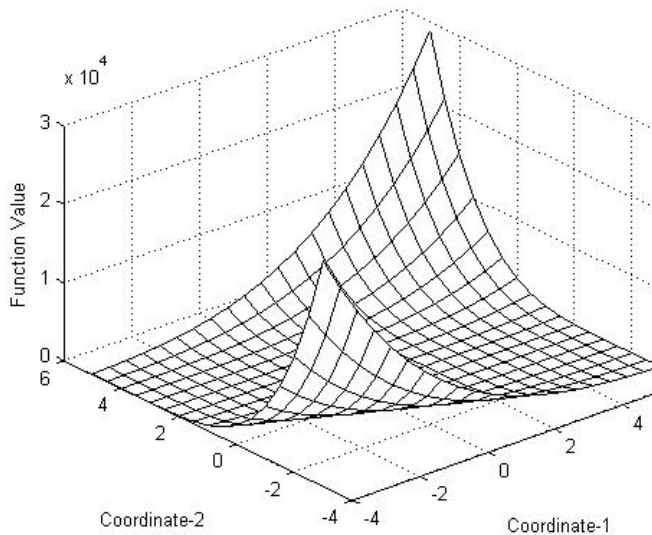


Figure 8 Variably dimensioned function for $d = 2$

dimensioned function for $d = 2$. The search area is a U-shaped curve, which is crossed by a flat area. There are numerous local minima in the region of a flat area but only one unique global minimum exists.

4.1.4. Test function 4: Symmetrical Gaussian function

The symmetrical Gaussian function is defined as

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

where $f_i(\mathbf{x}) = 100 - x_i$ for $i = 1, \dots, d$. The starting point is $\mathbf{x} = [70, \dots, 70]$. The optimal of the function value $g^* = 1$ is achieved at the point $\mathbf{x}^* = [100, \dots, 100]$. Figure 9 depicts the symmetrical Gaussian function for $d = 2$. If any starting point is in the area of the valley where the curve is changing, a Nelder-Mead method converges to the unique global minimum point. On the other hand, if any starting point is in the flat area, it is difficult for the Nelder-Mead method to reach the minimum point.

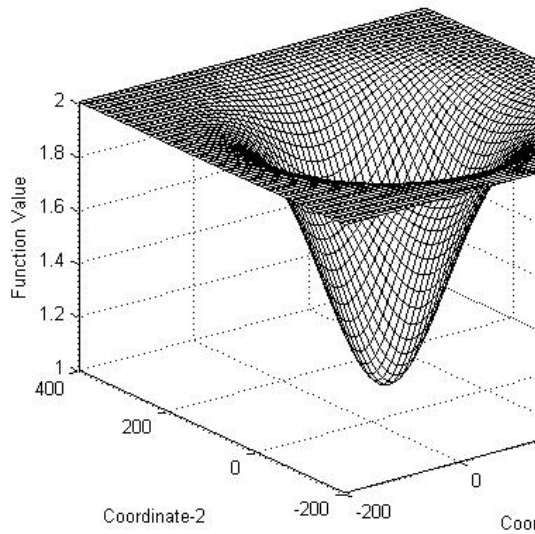


Figure 9 Symmetrical Gaussian function for $d = 2$

4.2. Search performance measures

When the search terminates, optimal solutions can be tested in at least 3 different ways: the solution on-hand, the most frequently visited solution, or the solution with the best cumulative averages (Banks 1998; Andradóttir 2006). 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.

4.2.1. 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 a rough indication of the total computational work required by a simulation procedure.

4.2.2. Deviation of the best estimated optimal function value from the true optimal value

For a measure of quality delivered by a simulation optimization procedure, we consider

$$D = \left| \frac{\mu(\hat{\mathbf{x}}^*) - \mu(\mathbf{x}^*)}{\mu(\mathbf{x}^*)} \right|, \quad (8)$$

where $\mu(\mathbf{x}^*)$ is the true minimum and $\mu(\hat{\mathbf{x}}^*)$ is the estimated minimum.

4.2.3. The analysis of mean (ANOM)

The analysis of mean (ANOM) is a graphical design tool which was introduced by Ott (1983) for determining which of k groups has significant difference with respect to their overall average.

The ANOM constructs an interval estimation by computing the lower- and upper-decision lines (LDL and UDL, respectively) as follows:

$$\begin{aligned}
 LDL: \quad & \bar{\bar{X}} - h_{k,n_j} \sqrt{\frac{S_p^2(k-1)}{n}}, \\
 UDL: \quad & \bar{\bar{X}} + h_{k,n_j} \sqrt{\frac{S_p^2(k-1)}{n}},
 \end{aligned} \tag{9}$$

where $\bar{\bar{X}}$ is the overall average or grand mean; $\bar{\bar{X}} = \sum_{j=1}^k \bar{X}_j / k$. Also, the sample size of group j is n_j , the pooled variance is $S_p^2 = (S_1^2 + S_2^2 + \dots + S_k^2) / k$, and the Nelson's h statistics with k groups and n_j equal sample sizes are given in the table of h statistics in Nelson (1993).

Any individual group mean not contained in the interval (LDL, UDL) is considered to be significantly lower than the overall group average if it falls below LDL and significantly above the overall group average if it lies above UDL. The test setup is explained in the next section.

4.3. Empirical test setup

Our implementations were run on MATLAB by modifying the *fminsearch* function. The NM coefficients were as follows: $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$ and $\tau = 0.5$. The initial step size, s_i in (3), was 10^{-4} . Minimum deviation ε_x and $\varepsilon_{\hat{\mu}}$ were 10^{-4} . Maximum budget consumption $N_{\hat{\mu}}$ and N_{search} were 10^5 . To estimate the objective function, the sample size m was 6. The standard deviations of random noise ξ_x were 0.10, 0.15, 0.20, 0.25 and 0.30 times $g(\mathbf{x}^*)$. These standard deviations were small compared with the lowest of degree of randomness 0.75 times of $g(\mathbf{x}^*)$ used by Tippayawannakorn and Pichitlamken (2013). We used these smaller values to test the sensitivity of each search algorithm when the variance is changed for a small amount.

The factor of increasing simulation size b used by Tomick (1995) was 1.25. We performed 15 macoreplications (i.e., experiments) for each test problem on the following 10 search algorithms:

1. NM – The original Nelder-Mead Simplex,
2. RS9 – The Barton and Ivey stochastic modification,
3. RSS – The RS with Restart search
4. NMSM – The Nelder-Mead selection with memory,
5. NMSMRS9 – The NMSM with RS9,
6. NMSMRSS – The NMSM with RSS,
7. ANRS9 – Adaptive Nelder-Mead modification,
8. ANRSS – Adaptive Nelder-Mead modification with RSS,
9. ANSMRS9 – Adaptive Nelder-Mead modification with memory
10. ANSMRSS – ANSM with RSS.

5. Empirical Test Results

Figure 10 shows the ANOM of the logarithm of the budget consumption of each algorithm until either the computation budget was exhausted or until the search was unable to get any improvements. Figure 11 contains the ANOM of average deviance in estimating the objective function values as defined in (8). As expected, as the dimensionality is increased, a given test problem becomes more difficult. Most algorithms perform worse for all Test Functions when the dimensionality goes up, and their estimated optimal solutions become further away from the true optimum, i.e., $\bar{D}$ and the computational budget consumed increases as the dimension increases. As shown in Figures 10 and 11

as the degree of randomness increases the computational resources and the quality of solutions change uniformly for all test problems. This shows that all search algorithms have low sensitivity to randomness. As shown in Figure 10, the algorithms with memory utilization have approximately 50% less resource consumption than the corresponding algorithms with non-memory utilization. However, as shown in Figure 11, the deviances for the memory and non-memory algorithms are approximately the same. Similarly, the results show that the adaptive algorithms have approximately 30% less resource consumption than the non-adaptive algorithm, but the deviances are approximately the same for the adaptive and non-adaptive algorithms. The results also show that including both memory utilization and adaptivity in an algorithm reduces resource consumption by approximately 50% but does not appreciably improve the quality of solutions. A comparison of the restart (RSS) and non-restart methods shows that RSS gives a 50% increase in quality of solution. However, RSS uses 67% more computational resources than non-RSS. In terms of $\bar{D}$, the algorithm with RSS gives an improvement over the non-RSS algorithm. However, in terms of $\bar{D}$, the algorithm with memory-utilization and the adaptive algorithm do not appreciably improve the estimated optimal solution.

The One-way Normal ANOM for L

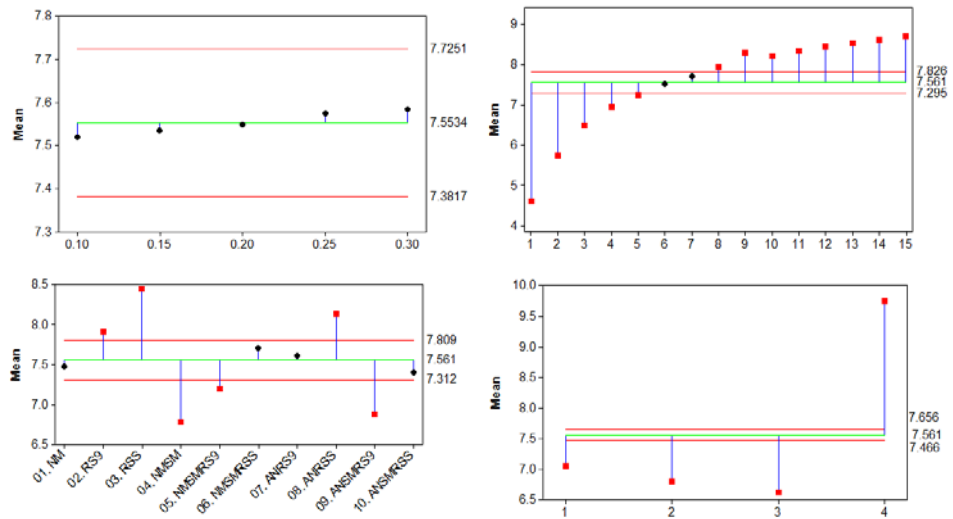


Figure 10 ANOM of logarithm of computational effort (L)

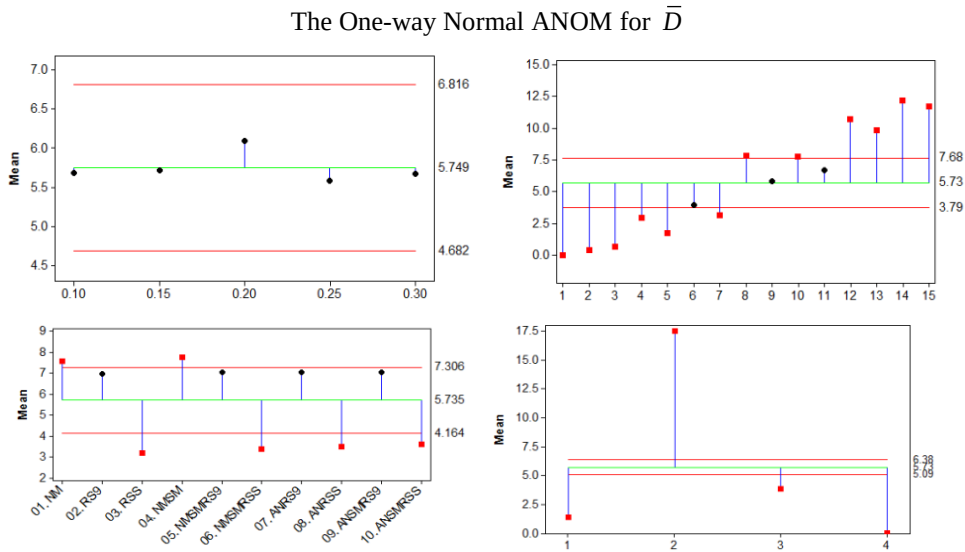


Figure 11 ANOM of average deviation of function value of algorithm from true function value ($\bar{D}$)

Figure 12 shows the logarithm of computation effort, L , of each algorithm for each test function, and Figure 13 shows the deviance, $\bar{D}$, of each algorithm for each test function. The figures show that Test Functions 1, 2 and 3 use similar small amounts of computation effort and that Test Functions 1, 3 and 4 have small $\bar{D}$.

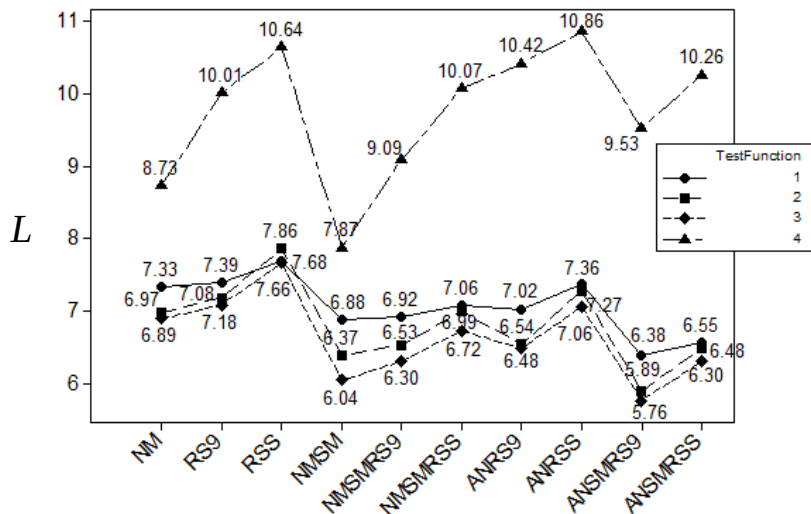


Figure 12 Logarithm of computational effort (L) plot for each test function

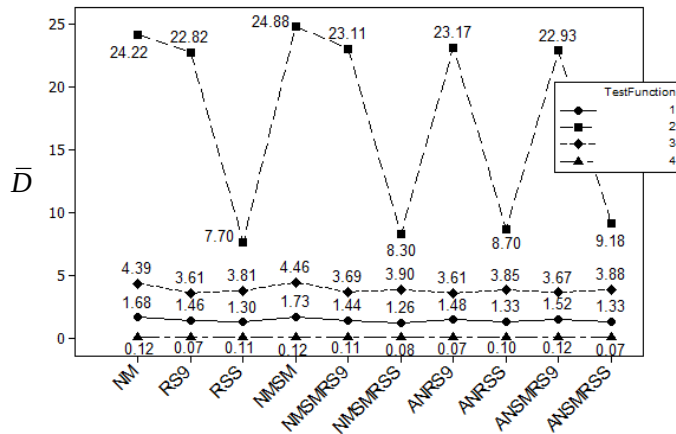


Figure 13 Average deviation of function value of algorithm from true function value ($\bar{D}$) plot for each test function

This may be because it is easier to find the unique global optima for Test Functions 1, 3 and 4. However, Test Function 4 requires large computation resources even though it has small deviance from the true optimal solution. The Test Function 3 has a unique global minimum although there are many local minima in the flat area around it. Test Function 4 requires a small amount of computation resource L but gives large $\bar{D}$. For this function, if the start point is in the flat region, then the improvement in function value is initially very slow, but as soon as the search reaches the valley region, a method rapidly converges to the global optimum. For Test Function 2, all algorithms give a high $\bar{D}$, but use a low amount of computational resource. The reason might be that Test Function 2 has a global minimum along a long, narrow valley. Although, it is possible to converge, an optimization algorithm obtains a high accuracy value for the function, but a very low accuracy for the optimal point $\mathbf{x}$.

To clarify the comparison, Table 1 contains relative ratios of computational effort between pairs (i, j) of algorithms, e^L_i / e^L_j and Table 2 contains the relative ratios $\bar{D}_i / \bar{D}_j$ of $\bar{D}$. The data are divided into 3 sets. Firstly, we compare adaptive and non-adaptive methods. The data shows that an adaptive method reduces resource consumption by approximately 25% on average compared to a non-adaptive method. However, the $\bar{D}$ values are similar for the two methods. Secondly, a comparison of the memory-utilization and non-memory utilization methods shows that the memory methods reduce resource utilization by approximately 50%. However, they have slightly more deviance. Lastly, a comparison of the Restart and non-Restart methods shows that the Restart methods give a better result. However, they use up to 68% more resources.

5. Conclusions

We show that the Nelder-Mead algorithm which is designed for deterministic optimization can be modified to accommodate discrete stochastic outputs. Using past information generally decreases computational effort but not reduce the performance significantly. Although all adaptive-with-memory algorithms, ANSMRS9 and ANSMRSS, do not improve the quality of the estimated solutions, they use less computational resources compared with their corresponding non-adaptive ones. The most economical algorithms on discrete small variation problems is the ANSMRSS algorithm.

Table 1 Comparison of relative computational effort of each pair of algorithm

	Algorithm	Test Function				Average ^c	Average ^b	Overall Average
		1	2	3	4			
Adaptive vs Non-Adaptive	ANRS9/RS9	0.690	0.527	0.548	1.500	0.816	0.913	
	ANSMRS9/NMSMRS9	0.583	0.527	0.584	1.553	0.812	0.869	0.693 ^a
	ANRSS/RSS	0.324	0.253	0.256	0.683	0.379	0.393	0.765 ^b
	ANSMRSS/NMSMRSS	0.600	0.603	0.652	1.203	0.765	0.806	
Memory vs Non-Memory	NMSM/NM	0.636	0.549	0.427	0.425	0.509	0.478	
	NMSMRS9/RS9	0.623	0.522	0.459	0.395	0.500	0.469	0.494 ^a
	NMSMRSS/RSS	0.540	0.420	0.393	0.568	0.480	0.465	0.491 ^b
	ANSMRS9/ANRS9	0.526	0.521	0.489	0.409	0.487	0.477	
	ANSMRSS/ANRSS	0.445	0.453	0.467	0.550	0.479	0.487	
Restart vs Non-Restart	RSS/RS9	1.338	1.967	1.786	1.871	1.061	1.073	
	NMSMSS/NMSMS9	1.160	1.582	1.529	2.686	1.070	1.070	1.591 ^a
	ANRSS/ANRS9	1.414	2.085	1.787	1.549	1.069	1.057	1.675 ^b
	ANSMRSS/ANSMRS9	1.194	1.810	1.709	2.080	1.073	1.067	

^a All Test Function average ^b 1,3,4 Test Function Average**Table 2** Comparison of relative $\bar{D}$ of each pair of algorithm

	Algorithm	Test Function				Average ^a	Average ^b	Overall Average
		1	2	3	4			
Adaptive vs Non-Adap-	ANRS9/RS9	1.013	1.015	0.999	0.931	0.989	0.981	
	ANSMRS9/NMSMRS9	1.058	0.992	0.996	1.096	1.036	1.030	0.992 ^a
	ANRSS/RSS	1.024	1.193	1.019	0.584	0.955	0.938	0.963 ^b
	ANSMRSS/NMSMRSS	1.054	1.107	0.996	0.789	0.986	0.969	
Memory vs Non-Memory	NMSM/NM	1.030	1.027	1.017	1.028	1.026	1.025	
	NMSMRS9/RS9	0.985	1.012	1.022	1.465	1.121	1.155	1.073 ^a
	NMSMRSS/RSS	0.971	1.078	1.023	0.741	0.953	0.949	1.088 ^b
	ANSMRS9/ANRS9	1.030	0.989	1.019	1.726	1.191	1.231	
	ANSMRSS/ANRSS	0.997	1.055	1.009	0.638	0.925	0.907	
Restart vs Non-Restart	RSS/RS9	0.850	0.000	1.227	1.041	0.908	0.828	
	NMSMRSS/NMSMRS9	0.876	0.359	1.058	0.782	0.478	1.002	0.767 ^a
	ANRSS/ANRS9	0.902	0.376	1.068	1.521	0.494	1.027	0.985 ^b
	ANSMRSS/ANSMRS9	0.873	0.400	1.058	0.562	0.514	0.994	

^a All Test Function average ^b 1,3,4 Test Function Average

Acknowledgements

This work 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. Krisda Lekdee, Mathematics and Statistics department, Rajamangala University of Technology Phra Nakhon, and Prof. Dr. Elvin James Moore, Mathematics department, King Mongkut's University of Technology North Bangkok, for their editorial comments.

References

- Alon G, Kroese D, Raviv T, Rubinstein R. Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Ann Oper Res*. 2005; 134: 137–151. doi: 10.1007/s10479-005-5728-8.
- Andradóttir S. Chapter 20 an overview of simulation optimization via random search. In: Shane GH, Barry LN. editors. *Handbooks in operations research and management science*. Elsevier; 2006. p. 617-631. Available from: <http://www.sciencedirect.com/science/article/pii/S0927050706130200>
- Banks J. Chapter 9 simulation optimization. In: *Handbook of simulation: Principles, methodology, advances, applications, and practice*. Wiley-Interscience; 1998. p. 307–334. Available from: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike0920&path=ASIN/0471134031>.
- Barton RR, Ivey JS. Nelder–Mead simplex modifications for simulation optimization. *Manage Sci*. 1996; 42: 954–973. doi: 10.1287/mnsc.42.7.954.
- Bonello AF, Mosca R, Cotto SP. Multi-objective optimisation of food refluent treatment plant design parameters through designed simulation with significant experimental error. In: *Proceeding of the 14th European Simulation Symposium Dresden; 2002; Germany*. p. 44-51. Available from: <http://www.scseurope.net/services/ess2002/PDF/meth-2.pdf>
- Branco DM. Nonlinear optimization of a stochastic function in a cell migration model. The degree of Master Science [dissertation], Worcester Polytechnic Institute; 2006. Available from: <http://www.wpi.edu/Pubs/ETD/Available/etd-050506-164020/>
- Carson Y, Maria A. Simulation optimization: methods and applications. In: *Proceedings of the Winter Simulation Conference; 1997. USA: IEEE Computer Society*. p. 118-126. doi: 10.1145/268437.268460.
- Dennis JE, Woods DJ. Optimization on microcomputers: The Nelder-Mead simplex algorithm. In: Wouk, A. (ed.) *New computing environments: microcomputers in large-scale computing*, SIAM, Philadelphia; 1987. p. 116-122.
- Gosavi A, Phatakwal S. A finite-differences derivative-descent approach for estimating form error in precision-manufactured parts. *J Manuf Sci E-T Asme*. 2006; 128: 355–359. doi: 10.1115/1.2124989
- Holland J. Genetic algorithms. *Sci AM*. 1992; 267: 66-72. doi: 10.1038/scientificamerican0792-66.
- Hooke R, Jeeves TA. Direct search solution of numerical and statistical problems. *J ACM*. 1961; 8: 212–229. doi:10.1145/321062.321069.
- Humphrey DG, Wilson JR. A revised simplex search procedure for stochastic simulation response surface optimization. *Inform J Comput*. 2000; 12: 272–283. doi: 10.1287/ijoc.12.4.272.11879.
- Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. *Science*. 1983; 220: 671–680. <http://home.gwu.edu/~stroud/classics/KirkpatrickGelattVecchi83.pdf>
- Kumral M. Solution of ore blending problem by stochastic approach. In: *Proceeding of the Mining and Exhibition of Turkey; 2003; Antalya, Turkey*. p. 485-490. Available from: http://www.maden.org.tr/resiler/ekler/2c8938e4cf57777_ek.pdf
- Moré JJ, Garbow BS, Hillstom KE. Testing unconstrained optimization software. *Acm T Math Software*. 1981; 7: 17-41. doi: 10.1145/355934.355936.
- Neddermeijer HG, Oortmarssen GJ, Van PN, Dekker R, Habbema JDF. Adaptive extensions of the Nelder and Mead simplex method for optimization of stochastic simulation models; 2000. *Econometric Institute Report, No. EI 2000-22/A*. Erasmus University Rotterdam, Econometric Institute. Available from: <http://econpapers.repub.ec.org/paper/dgreuire/1765001655.htm>.
- Nelder JA, Mead R. A simplex method for function minimization. *Comput J*. 1965; 7: 308-313. doi:

10.1093/comjnl/7.4.308.

- Nelson PR. Additional uses for the analysis of means and extended tables of critical values. *Technometrics*. 1993; 35: 61-71.
- Ólafsson S, Kim J. Simulation optimization: simulation optimization. In: *Proceedings of the Winter Simulation Conference: Exploring New Frontiers*; 2002. p. 79-84. Available from: <http://dl.acm.org/citation.cfm?id=1030453.1030467>.
- Ott ER. Analysis of means—a graphical procedure. *Industrial Quality Control*. 1967; 24: 101–109. Reprinted in *J Qual Technol*. 1983; 15: 10–18.
- Schwefel HPP. *Evolution and optimum seeking: The sixth generation*. New York, USA: John Wiley&Sons, Inc.; 1993.
- Shi L, Ólafsson S. An integrated framework for deterministic and stochastic optimization. In: *Proceeding of the Winter Simulation Conference*; 1997. USA: IEEE Computer Society. p. 358-365. doi: <http://doi.ieeecomputersociety.org/10.1109/WSC.1997.640421>.
- Spendley W, Hext GR, Himsworth FR. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*. 1962; 4: 441-461. doi: 10.2307/1266283.
- Swisher JR, Hyden PD, Jacobson SH, Schruben LW. A survey of recent advances in discrete input parameter discrete-event simulation optimization. *IIE Trans*. 2004. p. 441-461. doi: 10.2307/1266283.
- Tippayawannakorn N, Pichitlamken J. Nelder–Mead method with local selection using neighborhood and memory for stochastic optimization. *JCS*. 2013; 9: 463-476. doi: 10.3844/jcssp.2013.463.476.
- Tomick JJ. On convergence of the nelder–mead simplex algorithm for unconstrained stochastic optimization. PhD [Dissertation], Pennsylvania State University; 1995. Available from: <http://dl.acm.org/citation.cfm?id=921967>.
- Tomick JJ, Arnold SF, Barton RR. Sample size selection for improved Nelder–Mead performance. In: *Proceeding of the Winter Simulation Conference*; 1995. IEEE. p. 341-345. doi: 10.1109/WSC.1995.478754.
- Yan D, Mukai H. Stochastic discrete optimization. *Siam J Control Optim*. 1992; 30: 594-612. doi: 10.1137/0330034.