Adaptive Extensions of the Nelder and Mead Simplex Method for Optimization of Stochastic Simulation Models

H. Gonda Neddermeijer / *Econometric Institute, Faculty of Economics* and *Department of Public Health, Faculty of Medicine and Health Sciences, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands; EMail: neddermeijer@few.eur.nl*

Gerrit J. van Oortmarssen[1] / *Department of Public Health, Faculty of Medicine and Health Sciences, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands; EMail: vanoortmarssen@mgz.fgg.eur.nl*

Nanda Piersma / *Econometric Institute, Faculty of Economics, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands; EMail: piersma@few.eur.nl*

R. Dekker / *Econometric Institute, Faculty of Economics, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands; EMail: rdekker@few.eur.nl*

J. Dik F. Habbema / *Department of Public Health, Faculty of Medicine and Health Sciences, Erasmus University Rotterdam, P.O.Box 1738, 3000 DR Rotterdam, The Netherlands; EMail: habbema@mgz.fgg.eur.nl*

ECONOMETRIC INSTITUTE REPORT EI2000-22/A

Subject classifications: Simulation, Programming: Nonlinear: Algorithms, Health Care
Other key words: Nelder and Mead Simplex Method

We consider the Nelder and Mead Simplex Method for the optimization of stochastic simulation models. Existing and new adaptive extensions of the Nelder and Mead simplex method designed to improve the accuracy and consistency of the observed best point are studied. We compare the performance of the extensions on a small microsimulation model, as well as on five test functions. We found that gradually decreasing the noise during an optimization run is the most preferred approach for stochastic objective functions. The amount of computation effort needed for successful optimization is very sensitive to the timing of noise reduction and to the rate of decrease of the noise. Restarting the algorithm during the optimization run, in the sense that the algorithm applies a fresh simplex at certain iterations during an optimization run, has adverse effects in our tests for the microsimulation model and for most test functions.

[1]Corresponding author.

This paper investigates the performance of adaptive extensions of the Nelder and Mead simplex method for optimization of stochastic simulation models. Although the Nelder and Mead simplex method was originally designed for optimization of deterministic multidimensional functions (Nelder and Mead, 1965), it is frequently used for the optimization of stochastic objective functions. In particular, this method can be used for the optimization of stochastic simulation models, where one tries to estimate the model parameters that optimize some specific stochastic output of the simulation model. In this optimization procedure, the stochastic simulation model is often considered as a black-box model (Pflug, 1996) where the output of the simulation model can be regarded as a stochastic function of the model parameters.

The goal of this investigation is to find optimization methods that can be used for stochastic simulation models for which the corresponding stochastic objective function has the following characteristics (Wright, 1996):

- Calculation of this function is very expensive or time-consuming.
- Exact first partial derivatives of this function cannot be calculated.
- Numerical approximation of the gradient of this function is impractically expensive or slow.

The Nelder and Mead simplex method is a potential candidate for optimization of a function with the properties listed above, since it is a direct search method, i.e. it only uses function values and does not require a gradient (Wright, 1996).

Our particular interest in the Nelder and Mead Simplex method stems from the need for efficient algorithms for the optimization of microsimulation models for disease control. In microsimulation models individual fictitious life histories are simulated, where each of the simulated individuals can be at risk for developing a certain disease. Microsimulation models are used for the evaluation of specific interventions. For example, the cancer screening microsimulation model MISCAN (Loeve et al., 1999) is used in the evaluation of mass cancer screening programs. Microsimulation models for infectious diseases (Plaisier et al., 1990; Plaisier et al., 1998; van der Ploeg et al., 1998) are used for evaluation of interventions such as control of transmission of vector-borne diseases, and promotion of condom use and mass treatment for sexually transmitted diseases.

For the evaluation of interventions the parameters of the microsimulation model have to be quantified. Only some of these parameters can be quantified directly on basis of existing knowledge. Inferences for other parameters can be obtained by optimizing the goodness-of-fit of the simulation model on empirical data (van Oortmarssen et al., 1990). Evaluation of this stochastic objective function is often very time-consuming.

The Nelder and Mead simplex method is robust to small inaccuracies or stochastic perturbations in function values since it only uses the ranks of the function values to determine the next move, not the function values themselves (Barton and Ivey, 1991). However, considerable noise may change the relative ranks of the function values, leading to inappropriate termination, possibly at a solution that is far from the optimum (Barton and Ivey, 1996). Since the first paper on the Nelder and Mead Simplex method, numerous modifications have been proposed (for an overview, see Betteridge, Wade and Howard (1985)). Part of these modifications aim at improving its performance for stochastic objective functions or in particular for stochastic simulation models.

Some simple modifications that are proposed are restarting the optimization in the current observed best point (Betteridge, Wade and Howard, 1985), and re-evaluation of the function value in the currently observed best point (see e.g. Spendley, Hext and Himsworth (1962)). Barton and Ivey (1991, 1996) compare a number of re-evaluation strategies and some modifications of the shrinking of the simplex during the optimization. They propose a modified algorithm that improves the performance of the original Nelder and Mead simplex method, in terms of a smaller probability of inappropriate termination of the optimization process.

Tomick, Arnold and Barton (1995) report further improvements of this modified algorithm when applied to stochastic test functions. They consider the average of a number of replicated observations to evaluate the stochastic objective function. The number of replications used in an iteration step is determined dynamically by considering a statistical test on the differences of the function values in the current simplex. Humphrey and Wilson (1998) designed a three-phase application of the Nelder and Mead Simplex Method that is based on restart. The size of the initial simplex and the way the simplex is shrunk during an optimization run is changed in each phase of their algorithm. The incentive for starting a new phase is based on the size of the simplex, and the observed best point of the algorithm is defined as the minimum of the observed best points of each of the three phases. They report improved performance over the modified algorithm by Barton and Ivey (1996).

In the optimization of a stochastic simulation model, the key issue to address is to identify whether and when noise is dominating the optimization process. Precise timing of actions like noise reduction or restart is very important, since otherwise the effect of these actions may not be optimal. For example, the algorithm may have been drifting around already for several iterations, or the algorithm is corrected too soon, thus preventing the algorithm to make serious progress towards the optimum.

In this paper we will investigate and compare the performance of adaptive extensions of the Nelder and Mead simplex method that address both the timing and the type of action to be taken to improve the optimization process. In these extensions, we incorporate the existing ideas found in literature as described above. At the start of each iteration of the algorithm, a

criterion is checked to test whether the optimization process is hindered by noise. If the criterion is fulfilled, an action is taken that adapts the evaluation of the function values and / or the size of the simplex before the optimization routine is continued.

We consider a number of criteria that check whether the noise is dominating the optimization process. These criteria identify when an accidentally good function evaluation hinders the optimization process, or when the differences between the function values are dominated by noise. The set of actions that we will consider are rather straightforward and include noise reduction of the simulation model, restarting the optimization run and re-evaluating the currently best point. We apply the criterion-action modifications in an automated Nelder and Mead simplex algorithm. Therefore, the algorithm has an automated means of identifying when noise dominates the optimization process and it is able to correct itself when it encounters this situation.

We test the extensions for accuracy, consistency and computational effort. The accuracy is measured by the error in the returned observed best point compared to the optimum. Consistency is measured by the standard deviation of the errors resulting from repeated application of the algorithm. Small standard deviations show that repeated application of the algorithm gives comparable results.

The algorithms are tested on stochastic versions of well-known deterministic test functions. We also consider a microsimulation version of a model that is frequently used in evaluation of cancer screening policies. Although this model is a simplification of state-of-the-art models available, the model is well known and furthermore its performance can be checked by an analytical version of the model (Day and Walter, 1984). The optima of the test functions and the microsimulation model are known and therefore we can test for the accuracy and consistency of the automated adaptive Nelder and Mead simplex algorithms.

The results of our tests will show that the performance of the Nelder and Mead simplex method in stochastic optimization problems will benefit from the proposed modifications. The extensions prevent inappropriate termination of the optimization, and most of the extensions do not increase the computational effort too much.

In Section 1 the standard Nelder and Mead simplex method will be described. Section 2 contains the criteria and actions that we propose to improve the method when applied to stochastic simulation models. The remainder of the paper describes our test in Sections 3 and 4 and our test results in Section 5. We will end this paper with a conclusion and some points for further research.

1. Simplex algorithm

In this section we describe the Nelder and Mead Simplex Method for the optimization of an n-dimensional stochastic objective function. Without loss of generality, we assume that the

optimization is a minimization problem. Mathematically, this problem can be described by

$$\text{minimize} \quad f : D \to \mathbb{R}, \quad D \subseteq \mathbb{R}^n$$

where $f(\theta) = \mathbb{E}(F(\theta))$, $\theta \in D$. Here, $F(\theta)$ denotes the stochastic output for given input θ, and $\mathbb{E}(F(\theta))$ denotes its expected value. When optimizing a simulation model, the argument θ represents the parameters of the simulation model.

The simplex algorithm uses a *simplex* with $(n+1)$ vertices, and evaluates the objective function in every vertex. Based solely on the ranks of the observed function values in the vertices of the simplex, different steps can be taken, such as reflection, expansion or contracting vertices or shrinking the simplex, in order to find better vertices.

The original Nelder and Mead simplex algorithm (Nelder and Mead, 1965) can be described as follows. Iteration k of the algorithm starts with the simplex resulting from the previous iteration, consisting of the vertices $\left\{\theta_1^k, ..., \theta_{n+1}^k\right\}$ and corresponding function values $\left\{F\left(\theta_1^k\right), ..., F\left(\theta_{n+1}^k\right)\right\}$. First, the vertex with the lowest value (θ_{low}^k), the vertex with the highest value (θ_{hi}^k) and the vertex with the next-to-highest value (θ_{nexthi}^k) are determined. Next, vertex θ_{hi}^k is reflected through the centroid θ_{cent}^k of the remaining vertices to find a new vertex θ_{refl}^k:

$$\theta_{refl}^k = (1+\alpha)\,\theta_{cent}^k - \alpha\theta_{hi}^k, \quad \alpha > 0$$

and the objective function is evaluated in vertex θ_{refl}^k. A new simplex is then constructed as follows:

1. If $F(\theta_{refl}^k) \geq F(\theta_{hi}^k)$ then the objective function is evaluated in a contracted vertex between θ_{hi}^k and θ_{cent}^k, defined by

   $$\theta_{c1}^k = \beta\theta_{hi}^k + (1-\beta)\,\theta_{cent}^k, \quad 0 < \beta < 1$$

   If $F(\theta_{c1}^k) < F(\theta_{hi}^k)$, then the new simplex is found by replacing vertex θ_{hi}^k with vertex θ_{c1}^k, otherwise the new simplex is found by shrinking the current simplex around vertex θ_{low}^k, replacing vertex θ_i^k with

   $$\delta\theta_i^k + (1-\delta)\,\theta_{low}^k, \quad i = 1, ..., n+1, \quad \theta_i^k \neq \theta_{low}^k, \quad 0 < \delta < 1$$

2. If $F(\theta_{nexthi}^k) < F(\theta_{refl}^k) < F(\theta_{hi}^k)$ then the objective function is evaluated in a contracted vertex between θ_{refl}^k and θ_{cent}^k, defined by

   $$\theta_{c2}^k = \beta\theta_{refl}^k + (1-\beta)\,\theta_{cent}^k, \quad 0 < \beta < 1$$

   If $F(\theta_{c2}^k) < F(\theta_{refl}^k)$, then the new simplex is found by replacing vertex θ_{hi}^k with vertex θ_{c2}^k, otherwise the new simplex is found by first replacing vertex θ_{hi}^k by vertex θ_{refl}^k and subsequently shrinking the resulting simplex around vertex θ_{low}^k, replacing vertex θ_i^k with

   $$\delta\theta_i^k + (1-\delta)\,\theta_{low}^k, \quad i = 1, ..., n+1, \quad \theta_i^k \neq \theta_{low}^k, \quad 0 < \delta < 1$$

3. If $F(\theta^k_{low}) < F(\theta^k_{refl}) < F(\theta^k_{nexthi})$ then the new simplex is found by replacing vertex θ^k_{hi} with vertex θ^k_{refl}.

4. If $F(\theta^k_{refl}) < F(\theta^k_{low})$ then the objective function is evaluated in an expanded vertex $\theta^k_{\exp}$ defined by

$$\theta^k_{\exp} = \gamma\theta^k_{refl} + (1-\gamma)\,\theta^k_{cent}, \quad \gamma > 1$$

If $F(\theta^k_{\exp}) < F(\theta^k_{low})$, then the new simplex is found by replacing vertex θ^k_{hi} with vertex $\theta^k_{\exp}$, otherwise the new simplex is found by replacing vertex θ^k_{hi} with vertex θ^k_{refl}.

The next iteration starts with the new simplex $\left\{\theta^{k+1}_1, ..., \theta^{k+1}_{n+1}\right\}$. Vertex θ^{k+1}_{low} is taken as the estimator for the optimum θ^* at the kth iteration. The parameters $(\alpha, \beta, \gamma, \delta)$ are traditionally set to $(1, 0.5, 2, 0.5)$ (Nelder and Mead, 1965; Barton and Ivey, 1996). If a vertex is defined outside the domain D, either during initialization of the first simplex or during an iteration, then this vertex is projected onto the boundary of this region before evaluating it.

We will compare extensions of the Nelder and Mead simplex method on their performance over a large preset number of evaluations, assuming that the results of the algorithm will not improve much further if more evaluations are used. The issue of finding an appropriate stopping criterion will not be addressed in this paper.

In comparing the extensions, a variant of the original Nelder and Mead simplex algorithm which includes two well-established modifications will serve as a benchmark algorithm (see Figure 1). The extensions will be applied to this benchmark algorithm instead of to the original Nelder and Mead simplex algorithm.

First, it is standard practice today to compare the expanded vertex $\theta^k_{\exp}$with the reflected vertex θ^k_{refl} in the expansion step (Åberg and Gustavsson, 1982; Morgan and Burton, 1990; Wright, 1996; Lagarias, Reeds, Wright and Wright, 1998; McKinnon, 1998) instead of comparing the expanded vertex with vertex θ^k_{low} (Nelder and Mead, 1965; Barton and Ivey, 1996). Secondly, Barton and Ivey (1996) investigated the performance of a modification that re-evaluates the objective function in the best vertex at each shrink step and reduces the simplex by 10% ($\delta = 0.9$) at each shrink step rather than 50% ($\delta = 0.5$). They found that the modified algorithm, when applied to stochastic objective functions, leads to improvements in the expected value of the objective function at termination at the cost of more function evaluations. Other studies (Tomick, Arnold, and Barton 1995; Humphrey and Wilson, 1998; Neddermeijer et al., 1999) also found this modification to perform significantly better for stochastic functions. We include this second modification in the benchmark algorithm, since we are especially interested in a simplex algorithm that further improves the performance on stochastic problems with considerable noise, beyond the improvements proposed by Barton and Ivey. Preliminary tests showed that the algorithm that results from applying the two modifications to the original algorithm can indeed be used as a benchmark algorithm.

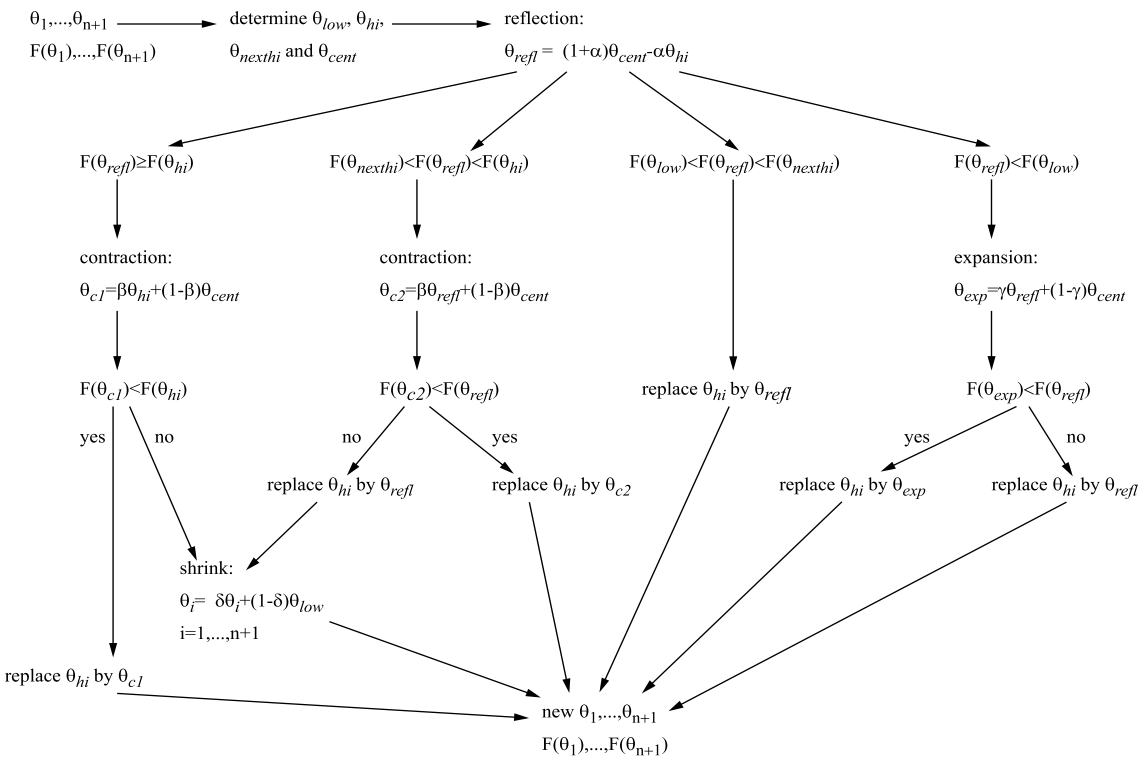

Figure 1. Flow chart for iteration k of the benchmark Nelder and Mead simplex algorithm (the superscript k is suppressed for notational convenience).

Nelder and Mead (1965) mention two types of initial simplices, i.e. regular simplices and axial or cornered simplices. Öberg (1998) found that regular initial simplices perform better than cornered initial simplices. In this paper, we will therefore use a regular initial simplex. This initial simplex $\left\{\theta_1^0, ..., \theta_{n+1}^0\right\}$ is given by :

$$\begin{aligned}
\theta_1^0 &= \left(\tau_1^0, ..., \tau_n^0\right) \\
\theta_2^0 &= \left(\tau_1^0 + \lambda_1, \tau_2^0 + \mu_2, ..., \tau_n^0 + \mu_n\right) \\
&... \\
\theta_{n+1}^0 &= \left(\tau_1^0 + \mu_1, ..., \tau_{n-1}^0 + \mu_{n-1}, \tau_n^0 + \lambda_n\right)
\end{aligned}$$

Here,

$$\lambda_i = \frac{c_i}{n\sqrt{2}}\left\{\sqrt{n+1} + n - 1\right\} \quad \text{and} \quad \mu_i = \frac{c_i}{n\sqrt{2}}\left\{\sqrt{n+1} - 1\right\}$$

where $\theta_1^0 = (\tau_1^0, ..., \tau_n^0)$ denotes the starting point of the optimization and the step sizes $(c_1, ..., c_n)$ determine the size of the initial simplex.

2. Extensions

The adaptive extensions consist of a criterion that will be checked in each iteration of the Nelder and Mead simplex algorithm, and an action that is taken when this criterion is fulfilled. At the beginning of an iteration k, a simplex with vertices $\left\{\theta_1^k, ..., \theta_{n+1}^k\right\}$ and the corresponding observed function values $\left\{F\left(\theta_1^k\right), ..., F\left(\theta_{n+1}^k\right)\right\}$ are given. At this point in the iteration a criterion is checked, and possibly an action is taken. The iteration then proceeds with the first step in each iteration, i.e. the reflection step.

It is important to note that when the parameters of a simulation model are being optimized, a function evaluation consists of performing one or more simulation runs. To evaluate a vertex θ we need to specify the simulation size used for one simulation run, S, and the number of simulation runs, N. If multiple simulation runs are used, i.e. $N > 1$, then the observed function value in vertex θ is defined as the average of the N replicated observations $F^{(j)}(\theta), \;\; j = 1, ..., N$.

2.1 Criteria

2.1.1 Simplex size

We noticed that if there is considerable noise, the best function value shows no further improvement when the size of the simplex starts to decrease. This observed function value can still be far from the optimum. If the simplex becomes too small, the differences in the function values in the $(n + 1)$ vertices are probably small as well, and are therefore likely to be dominated by noise. In this case, only minor improvements of the observed best function value are possible. Therefore, we want to detect the moment that the simplex size stops increasing or starts to decrease.

Our first criterion is based on the relative size of the simplex also used in the Dennis and Woods stopping criterion (1987). The relative size of the simplex in iteration k is defined as:

$$\psi_k = \frac{1}{\Delta_k} \max_{\substack{i=1,...,n+1 \\ \theta_i^k \neq \theta_{low}^k}} \left\|\theta_i^k - \theta_{low}^k\right\| \;\; \text{where } \Delta_k = \max\left(1, \left\|\theta_{low}^k\right\|\right)$$

where $\|.\|$denotes the Euclidian norm. We consider the difference in the relative sizes of the simplices in two successive iterations, i.e.

$$\Psi_k = \psi_k - \psi_{k-1}$$

If the difference is smaller than some preset tolerance level, i.e. $\Psi_k < \varepsilon_{ss}$, then the criterion is fulfilled.

2.1.2 Lack of change in function value

This criterion tests on sufficient change in the observed function values. Using regression analysis we examine if the observed best function value has changed significantly during the preceding iterations. If this is not the case, then we conclude that the differences in the observed optima can only be attributed to noise.

At the start of iteration k, we apply regression analysis to a number (say q) of observed best function values found in the previous q successive iterations, by determining the linear model

$$y = \beta_1 x + \beta_0 + \varepsilon, \;\; \varepsilon \sim N(0, \sigma^2)$$

where the explanatory variable x is the iteration number and the dependent variable y relates to the observed function value. If the slope β_1 is not significantly different from zero, then we conclude that possible differences in the function values can only be attributed to noise. Although a linear model may not be the best fit for this improvement, we are solely interested in detecting a lack of change in the function values. For this criterion we thus consider a two-sided t-test for the null hypothesis $H_0 : \beta_1 = 0$ with significance level α_{lc}.

2.1.3 Dominant noise

This criterion, which originates from Tomick, Arnold, and Barton (1995), is only applicable to algorithms that use multiple simulation runs to evaluate a vertex. The simplex algorithm depends on the differences in the observed function values in the $(n+1)$ vertices. The criterion checks whether the function values are not significantly different from each other. If not, we conclude that the differences in the function values in the vertices can only be attributed to noise.

At the beginning of iteration k, we test the null hypothesis $H_0 : f\left(\theta_1^k\right) = f\left(\theta_2^k\right) = ... = f\left(\theta_{n+1}^k\right)$ using the observed function values $\left\{F\left(\theta_1^k\right), ..., F\left(\theta_{n+1}^k\right)\right\}$ generated from N_{k-1} replicated observations, by considering an F-test with significance level α_{dn} applied to the following data:

$$F^{(j)}\left(\theta_i^k\right), \quad j = 1, ..., N_{k-1}, i = 1, ..., n+1$$

2.1.4 Retaining the best function value and re-evaluation of the best vertex

This criterion uniquely combines with an action, so they will be described simultaneously. If the observed function value of the best vertex is accidentally low due to noise, then the best point could be retained for too long. Accidental low results can possibly be corrected by re-evaluating the best vertex. In the benchmark algorithm the best vertex is therefore re-evaluated at each shrink step. In addition, we evaluate the often used "n+1" rule for

re-evaluation (Spendley, Hext and Himsworth, 1962; Morgan and Deming, 1974; Shavers, Parsons and Deming, 1979). Here, the best vertex is re-evaluated if the best vertex is retained in $(n+1)$ successive iterations, where n is the number of parameters of the objective function. Since we already apply re-evaluation at each shrink step, we use an adjusted version of the "n+1" rule: if the observed function value in the best vertex is retained in $(n+1)$ successive iterations then the best vertex is re-evaluated before reflection is applied.

2.2 Actions

Besides re-evaluation of the best vertex we consider the actions restart and noise reduction.

2.2.1 Restart

If at the start of iteration k the chosen criterion is fulfilled, then a fresh simplex is constructed with the same dimensions as the initial regular simplex, using the current best vertex θ^k_{low} as starting point. Various studies mention restart as a possibility for dealing with noisy objective functions (Akitt, 1976; Gill, Murray and Wright, 1981; Betteridge, Wade and Howard, 1985II; Walters and Deming, 1985; Wright, 1996).

2.2.2 Noise reduction

As will be obvious, decreasing the amount of noise would largely solve the difficulties caused by this noise. We will apply this action in two ways (Azadivar and Lee, 1988): in case a single simulation run is used to evaluate a vertex, the simulation size is increased, whereas in case multiple simulation runs are used, the number of replicated observations is increased.

- Increasing the simulation size
  If at the start of iteration k the chosen criterion is fulfilled, then the simulation size S_k is increased by setting $S_k = \lfloor b_s \cdot S_{k-1} \rfloor$. All vertices in the current simplex are re-evaluated with the new simulation size, and the algorithm continues using the new simulation size S_k. To prevent explosion of the simulation size we set $S_k = \min(\lfloor b_s \cdot S_{k-1} \rfloor, S_{\max})$. Moreover, in order to prevent that the increase in simulation size becomes zero due to rounding, the inequality $(b_s - 1)S_0 \geq 1$ should hold.

- Increasing the number of replicated observations
  If at the start of iteration k the chosen criterion is fulfilled, then the number of replicated observations N_k is increased by setting $N_k = \lfloor b_r \cdot N_{k-1} \rfloor$. All vertices in the current simplex are re-evaluated with the new number of replicated observations, by adding $N_k - N_{k-1}$ replications, and the algorithm continues using the new number of replications N_k. Again, to prevent explosion of the number of replications we set $N_k = \min(\lfloor b_r \cdot N_{k-1} \rfloor, N_{\max})$, whereas the inequality $(b_r - 1)N_0 \geq 1$ should hold.

2.3 The tested extensions

The combinations of a criterion and an action that will be tested are shown in Table I. Increasing the simulation size is only applied to the algorithms using one simulation run per function evaluation, whereas increasing the number of replications is only applied to the algorithms using multiple simulation runs. The extended algorithms are compared with the benchmark algorithm denoted by BM. An initial simulation size S_0 and an initial number of replications N_0 are given and for the algorithms that increase the simulation size or the number of replications, a maximum simulation size $S_{\max}$ or a maximum number of replications $N_{\max}$ are given.

3. The test problems

We test the optimization algorithms on a set of five test functions with known optima, which consist of a deterministic term and an stochastic error term. We also consider a microsimulation version of an existing cancer screening model. This model has three parameters that need to be estimated from an observed data set by constrained minimization of a goodness-of-fit test statistic. For this particular model the optimal parameters can also be determined analytically.

3.1 One stage - one test breast cancer screening model

The microsimulation model is a simulation implementation of the breast cancer screening model developed by Day and Walter (1984). In this model only one disease stage, the detectable preclinical phase ($DPCP$), is modeled. At the end of the $DPCP$ a cancer is clinically diagnosed, usually on basis of symptoms, whereas during the $DPCP$ a cancer can be detected by a screening test for early detection of breast cancer. The $DPCP$ has incidence rate J and we assume that the duration of the $DPCP$ is exponentially distributed with parameter λ. A cancer that is clinically detected in the period following a negative screening test is called an interval cancer.

A screening program consisting of four annual screening rounds is modeled. The sensitivity of the screening test, i.e. the probability that the screening test correctly identifies an individual as being in the DPCP (Day and Walter, 1984), is denoted with φ. In each microsimulation run a large number of individual life histories, including disease processes and the impact of screening, is simulated. The microsimulation model results in estimates for the detection rates at each of the screening rounds and estimates for the incidence rates of interval cancers, for different time intervals following each of the screening rounds.

The model will be applied to data from the first randomized trial for breast cancer screening, viz. the HIP study (Day and Walter, 1984; Shapiro et al., 1974; van Oortmarssen et

al., 1990). In the HIP study approximately 62,000 women, who were aged between 40 and 64 at entry, were randomly allocated to either a study group or a control group. Only the study group was offered annual breast cancer screening for four years. About 65 percent of the study group (20,166 women) agreed to take part and were screened at least once (these women all attended the first screening). We will use follow-up data until 5 years after the last screening. The results from the HIP screening trial that will be used are described by Day and Walter (1984), and consists of 4 detection rates and 14 incidence rates of interval cancers occurring after a previous negative test result.

The parameters J, λ and φ will be estimated from the observed data set through minimization of a chi-square goodness-of-fit test statistic. This stochastic objective function is given by

$$F\left(J,\lambda,\varphi\right)=\sum_{i=1}^{18}\frac{\left(O_i-E_i\left(J,\lambda,\varphi\right)\right)^2}{E_i\left(J,\lambda,\varphi\right)}$$

where O_i is the observed number of cancers during screening round or interval i and E_i is the number of simulated cancers during screening round or interval i, $i=1,...,18$. The optimal parameters of the model for the HIP data are determined using the objective function

$$f\left(J,\lambda,\varphi\right)=\sum_{i=1}^{18}\frac{\left(O_i-A_i\left(J,\lambda,\varphi\right)\right)^2}{A_i\left(J,\lambda,\varphi\right)}$$

where A_i is the number of cancers during screening round or interval i, $i=1,...,18$, as predicted by the analytical implementation of the breast cancer screening model (Day and Walter, 1984). We determined the optimal parameters $(J^*,\lambda^*,\varphi^*)$ of the model applied to the HIP data set by extensive enumeration (using the step sizes 10^{-5} for J, 10^{-3} for λ and 10^{-3} for φ) of $f\left(J,\lambda,\varphi\right)$: $f\left(J^*,\lambda^*,\varphi^*\right)=f\left(0.0021,0.61,0.87\right)\approx 13.34$.

3.2 The test functions

We test randomized versions of five deterministic nonlinear objective functions. Each of these deterministic test functions has a unique optimum. Two of the test functions are classical, whereas the other test functions show a characteristic behavior that in our opinion may occur in stochastic objective functions resulting from microsimulation models. We randomize the deterministic test functions by adding a normal distributed error term with zero mean and standard deviation σ^2 to each test function. Independent random number streams are used for each optimization run. The test functions are:

1. Rosenbrock's function

$$f(X_1,X_2)=100\left(X_2-X_1^2\right)^2+\left(1-X_1\right)^2$$

This is a classical test function (Åberg and Gustavsson, 1982; Betteridge, Wade and Howard, 1985I; Brumby, 1989; Hedlund and Gustavsson, 1992; Nelder and Mead, 1965; Parker, Cave and Barnes, 1985; Phillips, 1972). The minimum, given by $f(1,1) = 0$, lies at the base of a banana-shaped valley (Gill, Murray and Wright, 1981).

2. Powell's singular function

$$f(X_1, ..., X_4) = (X_1 + 10X_2)^2 + 5(X_3 - X_4)^2 + (X_2 - 2X_3)^4 + 10(X_1 - X_4)^4$$

This often used classical test function has four parameters (Nelder and Mead, 1965; Phillips, 1972; Brumby, 1989). The minimum is given by $f(0,0,0,0) = 0$.

3. Symmetrical Gaussian function

$$f(X_1, X_2) = -10 \exp\left\{-\left[(100 - X_1)^2 + (100 - X_2)^2\right]/15000\right\}$$

This test function is adapted from Van der Wiel (1980). Apart from a small region around the optimum this symmetrical function is very flat. The minimum is given by $f(100,100) = -10$.

4. An asymmetrical function

$$f(X_1, ..., X_8) = \sum_{i=1}^{8} \left(2^{X_i - 4} + (6 - X_i)\right)$$

This function is highly asymmetrical. The minimum is given by $f(4.529, ..., 4.529) = 23.311$.

5. A 5-variable paraboloid

$$f(X_1, ..., X_5) = \sum_{i=1}^{5} X_i^2$$

This function is symmetrical and rather easy to optimize if no noise is included. However, we observed that in case noise is imposed on the function, optimization with the benchmark algorithm is still difficult. The minimum is given by $f(0,0,0,0,0) = 0$.

4. Experiments and statistical analysis

The tested extended algorithms and their control variables such as significance and tolerance levels are described in Tables I and II, respectively. In order to compare the algorithms in the same way for all six objective functions, we treat an evaluation of each of the the five test functions as if it were a simulation run.

For the algorithms that use one simulation run to evaluate a vertex, we set the initial simulation size to $S_0 = 50000$. For the algorithms that use multiple simulation runs, we set the initial simulation size to $S_0 = 10000$ and the number of replications to $N_0 = 5$. The maximum number of replications and maximum simulation size are set to $N_{\max} = 50$ and $S_{\max} = 500000$, respectively. Furthermore, the variance σ^2 of the additive error term of the test functions are set by $\sigma^2 = 50000/S_0$. In this way, an initial simulation size $S_0 = 50000$ corresponds to a standard normal distributed error term. If during an iteration the simulation size is multiplied with a factor b_s, then this is modeled by dividing the variance of the error by b_s.

The algorithms BM, RV-EV, SS-RS and LC-RS do not increase the simulation size or the number of replications. In case of optimization of the test functions, the results for these algorithms will not depend on the usage of either single or replicated observations. Therefore, when optimizing a test function these algorithms are only run using replicated observations. However, for the microsimulation model the results may very well depend on the use of either a single simulation run or multiple simulation runs to evaluate a vertex. Therefore, these algorithms are run for both settings, i.e. $N_0 = 1$, $S_0 = 50000$ and $N_0 = 5$, $S_0 = 10000$. The algorithms are denoted with superscript 1 or 5, depending on the number of simulation runs.

The relevant information for each of the six objective functions is given in Table III. From preliminary studies we find that when the settings described in Table III are applied, the benchmark algorithm often terminates far from the optimum and the results of repeated optimization runs can vary considerably, for all of the objective functions. For both the microsimulation model and each of the test functions, we perform twenty optimization runs with each optimization algorithm. The optimization runs are terminated after 250 function evaluations. For each of the algorithms we denote the number of iterations performed in optimization run j, $j = 1, ..., 20$ with M_j. For the simplex resulting from the kth iteration of optimization run j, the best vertex is only determined in the first step of the next iteration and is denoted by $\theta_{low}^{j,k+1}$. This vertex is used as as the estimator for the optimum θ^* at the kth iteration of this optimization run. We define the error in iteration k of optimization run j as the difference between the expected function value in the best vertex in this iteration, $\theta_{low}^{j,k+1}$, and the expected function value in the optimum θ^*:

$$\epsilon^{j,k} = E\left(F\left(\theta_{low}^{j,k+1}\right)\right) - E\left(F\left(\theta^*\right)\right) = f\left(\theta_{low}^{j,k+1}\right) - f\left(\theta^*\right), \; j = 1, ..., 20$$

For each algorithm we consider the smallest error for each of the 20 optimization runs, denoted by $\epsilon_{small}^{j} = \min_{k=1,...,M_j} \epsilon^{j,k}$, $j = 1, ..., 20$, and the final error of the optimization runs, i.e. the error at the end of the optimization runs, denoted by $\epsilon_{end}^{j} = \epsilon^{j,M_j}$, $j = 1, ..., 20$.

For each objective function, we want to compare the accuracy, consistency and computational effort of the algorithms. For each algorithm and for each of the two types of errors that we consider, the errors resulting from the 20 optimization runs are mutually

independent and identically distributed. The errors of the different algorithms are also mutually independent. However, as can be seen in the next section, the error distributions for the algorithms can be quite different, both in mean and variance. Therefore, to compare the accuracy of the algorithms, we test if there is any stochastic difference between the algorithms (Hollander and Wolfe, 1999), which means that we test if the probability that an error resulting from one algorithm is smaller than an error resulting from another algorithm is significantly different from $\frac{1}{2}$. First, we check if there is any significant overall stochastic difference between the algorithms, by applying the nonparametric Kruskal-Wallis test at a 5% significance level. If this is the case, then we test which extended algorithms perform stochastically different than the benchmark algorithm, using distribution-free multiple comparisons based on the Kruskal-Wallis test at a 5% significance level. We measure the consistency of the algorithms using the standard deviation of the errors.

For the optimization runs of the six test problems, the cumulative number of individuals that have been simulated at the end of each iteration of an optimization run is determined. We study the computational effort employed by the optimization algorithms by considering only the optimization runs for which the smallest error or the final error is smaller than some preset tolerance level Δ. For each algorithm, the number of runs for which the smallest or final error fulfills this property is denoted by $r_{small,\Delta}$ and $r_{end,\Delta}$, respectively. We compare the computational effort of the algorithms by reporting the number of individuals that are simulated until the error $\epsilon^{j,k}$ is smaller than the tolerance level Δ for the first time in these optimization runs, $j \in \{1, ..., 20\}$, which is denoted by $E^j_{end,\Delta}$ when the final error is considered and $E^j_{small,\Delta}$ when the smallest error is considered. We consider three tolerance levels: $\Delta = 0.5$, $\Delta = 1.0$, and $\Delta = 2.0$.

5. Results

We compare the extended Nelder and Mead simplex algorithms and the benchmark algorithm with respect to accuracy, consistency and computational effort. For all test functions and for the microsimulation model, we find that there is an overall significant difference between the optimization algorithms with respect to accuracy for both the smallest errors and the errors at the end of the optimization runs.

For the microsimulation model, algorithms BM, RV-EV, SS-RS and LC-RS were tested using both one large simulation run and five smaller simulation runs to evaluate a point. We do not find statistically significant differences in accuracy or consistency for these two settings, and we compare the extended algorithms with the benchmark algorithm that uses one simulation run to evaluate a point, i.e. algorithm BM[1].

The results for the test functions with respect to the accuracy, consistency and computational effort of the algorithms when considering the errors at the end of the

optimization runs are shown in Tables IVa, IVb and IVc. For each test function, the accuracy is represented by the average errors $\bar{\epsilon}_{end}$ and the consistency is represented by standard deviations of these errors, see Column 2. Column 3 indicates whether the accuracy of an algorithm differs significantly from the benchmark algorithm, as indicated by the results T_{end} of the distribution-free multiple comparisons based on the Kruskal-Wallis test. $T_{end} = + \ / \ 0 \ / \ -$ means that the algorithm concerned performed stochastically better / equal / worse than the benchmark algorithm when considering the errors at the end of the optimization runs. The last column shows $r_{end,0.5}$, the number of runs for which the error at the end of the run is below the tolerance level $\Delta = 0.5$, and $\bar{E}_{end,0.5}$, the average number of individuals that are simulated until the error is smaller than the tolerance level Δ for the first time in these optimization runs, which is used as a measure for the computational effort employed by the algorithms. In comparing the algorithms, the results for the smallest errors and tolerance levels $\Delta = 1.0$ and $\Delta = 2.0$ have been inspected as well. In Table V the results for the microsimulation model are shown, both for the smallest errors and the errors at the end at the optimization runs[2].

5.1 The re-evaluation algorithm

For the five test functions and for the microsimulation model we find that re-evaluation algorithm RV-EV does not perform significantly different from the benchmark algorithm with respect to accuracy, nor does this extended algorithm seem to improve the consistency or decrease the amount of computational effort needed. Re-evaluation is already part of the shrink step of the benchmark algorithm, and it appears that the additional re-evaluation measures that we tested are not able to further improve the optimization algorithm.

5.2 The restart algorithms

Except for the Symmetrical Gaussian function, we find that the restart algorithms SS-RS, LC-RS and DN-RS perform considerably worse than the benchmark algorithm. For all restart algorithms, we find that the errors frequently do no get below any of the tolerance levels Δ. Apparently, restart will disrupt the optimization process, resulting in a very low accuracy. If restart is applied, the algorithm has to start all over again, which annuls most of the progress that was already made during the optimization run. Apparently, this is insufficiently compensated by the possibility of finding a better starting point. This explanation is supported by the increase in accuracy when tolerance or significance levels are set in such a way that restart is delayed as long as possible.

[2] Additional tables for the smallest errors resulting from the test functions and for tolerance levels $\Delta = 1.0$ and $\Delta = 2.0$ for both the test functions and the microsimulation model are available on request from the corresponding author.

For the Symmetrical Gaussian function, the three restart algorithms yield considerable improvement in the accuracy and consistency when compared to the benchmark algorithm. The different behavior for this test function is due to the different nature of this problem. Any optimization algorithm has to find the small region with the optimum in a virtually flat surface. Through restart, it becomes more likely that one of the vertices is placed close to this small region.

5.3 The noise reduction algorithms

Algorithms SS-IS and LC-IS reduce the noise by increasing the simulation size, and algorithms SS-IR, LC-IR and DN-IR reduce the noise by increasing the number of replications. Most noise reduction algorithms perform stochastically better than the benchmark algorithm. The control variables of the algorithms can even be chosen in such a way that the algorithms perform substantially better than the benchmark algorithm with respect to accuracy and consistency, but note that in practice one does not know the best values for the control variables.

The improvement found for the noise reduction techniques depends on the nature of the objective function. For the easy five-variable paraboloid all but one of the noise reduction algorithms performs better than the benchmark, whereas for the rather complex eight-variable asymmetrical function and Powell's singular function the improvements depend on the timing of the action and on the values of the control variables. For the Symmetrical Gaussian function, only the algorithms SS-IS, LC-IS and DN-IR perform better than the benchmark algorithm for some values of the control variables. However, for this test function noise is not the main difficulty that one needs to overcome. The flat surface of the domain of this test function is the main cause of failure for the algorithms.

Each of the criteria DN, SS and LC is able to detect whether and when noise is dominating the optimization process. By continuing the optimization process with reduced noise, inappropriate termination of the optimization process is prevented. For all noise reduction algorithms, the improvement in accuracy and consistency is achieved at the cost of an increase in the computational effort. For most objective functions we find that algorithm DN-IR needs less additional computational effort than the other noise reduction algorithms. Moreover, DN-IR is less dependent on the values of its control variables.

The dominant noise criterion detects whether determination of the true ranks of the function values, which is crucial in each step of the simplex method, is impeded by the presence of noise. When the DN criterion is satisfied it is very likely that the optimization process is hindered by the presence of noise. The other criteria are less selective, in the sense that they can also be satisfied if the process is still successfully iterating towards a (local) optimum. For example, when the simplex approaches the neighborhood of the optimum, then the simplex size will decrease due to contraction and shrinking. Hence, depending on the

values of the control variables, the SS and LC criteria can lead to unnecessary and expensive noise reduction measures. The DN criterion also has a built-in correction mechanism: if the size is increased accidentally and thus too early, it will on average take longer before the criterion is fulfilled again and a further increase in size occurs. We conclude that the noise reduction extensions are effective modifications of the benchmark algorithm. The DN-IR algorithm is to be preferred in view of its accuracy and its computational effort.

5.4 Increasing simulation size vs. increasing number of replications

Compared to the benchmark algorithm, we find that in general the algorithms that increase the simulation size need more computational effort than the algorithms that use the same criterion but increase the number of simulation runs used to evaluate a vertex. In the latter case, only a number of replications are added in the current iteration, which obviously needs less computational effort than re-evaluating the vertex for the increased simulation size. On the other hand, accidentally good function values in vertices are not completely corrected, and indeed the results for the microsimulation model show some additional improvements in accuracy and consistency when the simulation size is increased instead of the number of replications. However, these improvements are not worth the considerable extra computational effort. For the test functions there is no clear difference in accuracy and consistency between the two actions when the same criterion is applied. Thus, the use of more replications is preferred over the use of a larger simulation size.

5.5 Applying a large simulation size

We find that applying an extended algorithm can lead to substantially more accurate and consistent results, at the cost of extra computational effort. However, if optimizing an objective function using the benchmark algorithm with a larger simulation size would lead to the same results as optimizing the objective function with a more complex extended algorithm, then there would be no point in using the extended algorithms. To investigate this, we performed twenty optimization runs with the benchmark algorithm using replicated observations ($N_0 = 5$) and three different large initial simulation sizes, i.e. $S_0 = 20000$, $S_0 = 50000$ and $S_0 = 100000$, for both the microsimulation model and each of the test functions. Again the optimization runs are terminated after 250 function evaluations.

In Table VI the results for the 5 test functions and for the microsimulation model are shown. Column 2 displays the initial simulation sizes for the benchmark algorithm. Column 3 displays the averages $\bar{\varepsilon}_{end}$ and the standard deviations of the errors at the end of the optimization runs. Column 4 shows $r_{end,0.5}$, the number of runs that return a final error smaller than 0.5, and $\bar{E}_{end,0.5}$, the average number of individuals that are simulated until the error is smaller than the tolerance level $\Delta = 0.5$ for the first time in these optimization runs.

From the results we find that increasing the simulation size in the benchmark algorithm leads to higher accuracy at the cost of substantial extra computational effort. Many of the noise reduction algorithms can result in a smaller average error and standard deviation than the benchmark algorithm with a large simulation size. However, as was mentioned above, the amount of computational effort needed by the noise reduction algorithms largely depends on the chosen values of the control variables, especially for algorithms SS-IS, SS-IR, LC-IS and LC-IR. It is likely that when one of these four noise reduction algorithms is applied without thoughtfully choosing values for the control variables, the amount of computational effort needed for an improvement in accuracy and consistency is comparable to the effort needed by the benchmark algorithm with a large initial simulation size. Therefore, when applying such a noise reduction algorithm, extra information regarding the stochastic objective function is needed to choose the values for the control variables. On the other hand, since the computational effort needed by algorithm DN-IR is considerably less dependent on the values of the control variables, we believe that this algorithm is probably more efficient than the benchmark algorithm in combination with a large initial simulation size, even if the values of the control variables are chosen without any knowledge of the stochastic objective function.

6. Conclusion and Further Research

The Nelder and Mead simplex method is useful for the optimization of stochastic simulation models. However, the amount of noise in the simulation model largely determines the success of the optimization procedure. The study described in this paper is designed to improve the performance of the Nelder and Mead simplex method when applied to the optimization of stochastic simulation models. Several adaptive extensions of the Nelder and Mead simplex algorithm are tested using five test functions, and a representative (albeit simple) microsimulation model. These adaptive extensions include algorithms that apply restart and re-evaluation and algorithms that gradually apply noise reduction during an optimization run. We compared the extended algorithms to a benchmark algorithm based on the original method by Nelder and Mead.

We find that relatively simple extensions of the Nelder and Mead simplex method are able to detect whether and when noise obstructs the optimization process, and consequently, indicate the moments when a reduction of the noise is needed to successfully continue the optimization process. For the test microsimulation model and for most of the test functions, gradually increasing the simulation size or the number of simulation runs used for a function evaluation, can lead to considerable improvements in accuracy and consistency of the observed estimator of the optimum. For one of the objective functions however we find that restarting the algorithm will improve the optimization process more than decreasing the noise. This is presumably related to the almost flat surface for the entire domain for this function except for

a small region around the optimum.

The criterion that determines the timing of noise reduction considerably influences the efficiency of the algorithm, in terms of the computational effort needed for an improvement in accuracy and consistency. In particular, we find that the amount of computational effort needed by the noise reduction algorithms using the simplex size criterion or the lack of change criterion is very sensitive to the values of the control variables of the algorithms. Therefore, extra information on the stochastic objective function is needed to properly choose these values. Due to the black-box nature of a stochastic simulation model, it is likely that such information is not available. The amount of computational effort needed by algorithm DN-IR, which applies the dominant noise criterion, is less dependent on the values of the control variables. The evaluation of a stochastic simulation model is often very time-consuming, and if no extra information on the corresponding stochastic objective function is available, algorithm DN-IR is to be preferred over the use of the benchmark algorithm with a expensively large simulation size.

In the tests described in this paper we use a maximum number of evaluations to end the optimization process. However, in applications, the use of more sophisticated stopping criteria, such as the Dennis and Woods stopping criterion (1987), should be considered. Furthermore, some consideration should be given to the choice of the initial simplex size. We want to emphasize the fact that the Nelder and Mead simplex method is a local search method, no guarantee is given for finding the global optimum. Therefore, multistart using multiple starting points and / or multiple searches from the same starting point should always be considered. We did not study the effect of multistart, since we were primarily interested in the performance of single applications of the extended algorithms.

In this paper we focus on noise reduction algorithms that apply their adaptive measures to all vertices used during an iteration. This type of algorithms can also be compared to modified algorithms that determine the simulation size for each separate function evaluation during the optimization run. For example, Azadivar and Lee (1988) describe an algorithm that chooses the simulation size for each vertex in such a way that whenever two function evaluations have to be compared, the difference between the two function evaluations is statistically significant.

In addition to considering alternative improvements of the Nelder and Mead simplex algorithm, the question how this algorithm compares to other algorithms such as Stochastic Approximation, Response Surface Methodology and Simultaneous Perturbation Stochastic Approximation (see Kleijnen, 1998; Jacobson and Schruben, 1989; Spall, 1992; Fu, 1994) in the optimization of stochastic objective functions also remains to be addressed.

Acknowledgements

Alex J. Koning is acknowledged for his suggestions regarding the statistical aspects in this paper.

References

Åberg, E.R., A.G.T. Gustavsson. 1982. Design and Evaluation of Modified Simplex Methods. *Analytica Chimica Acta* **144** 39-53.

Akitt, J.W. 1977. Function Minimization Using the Nelder and Mead Simplex Method with Limited Arithmic Precision: the Self Regenerative Simplex. *The Computer Journal* **20** 84-85.

Azadivar, F., Y.-H. Lee. 1988. Optimization of Discrete Variable Stochastic Systems by Computer Simulation. *Mathematics and Computers in Simulation* **30** 331-345.

Barton, R.R., J.S. Ivey. 1996. Nelder-Mead Simplex Modifications for Simulation Optimization. *Management Science* **42(7)** 954-973.

Barton, R.R., J.S. Ivey. 1991. Modifications of the Nelder-Mead Simplex Method for Stochastic Simulation Response Optimization. B.L. Nelson, W.D. Kelton, G.M. Clark, eds. *Proceedings of the 1991 Winter Simulation Conference.* IEEE Press, Piscataway, NJ. 945-953.

Betteridge, D., A.P. Wade, A.G. Howard. 1985I. Reflections on the Modified Simplex-I. *Talanta* **32(8B)** 709-722.

Betteridge, D., A.P. Wade, A.G. Howard. 1985II. Reflections on the Modified Simplex-II. *Talanta* **32(8B)** 723-734.

Brumby, S. 1989. Exchange of Comments on the Simplex Algorithm Culminating in Quadratic Convergence and Error Estimation. *Analytical Chemistry* **61** 1783-1786.

Day, N.E., S.D. Walter. 1984. Simplified Models of Screening for Chronic Disease: Estimation Procedures from Mass Screening Programmes. *Biometrics* **40** 1-14.

Dennis, J.E., D.J. Woods. 1987. Optimization on Microcomputers: The Nelder-Mead Simplex Algorithm. A. Wouk, ed. *New Computing Environments: Microcomputers in Large Scale Computing.* SIAM, Philadelphia, PA. 116-122.

Fu, M.C. 1994. Optimization via Simulation: A Review. *Annals of Operations Research* **53** 199-247.

Gill, P.E., W. Murray, M.H. Wright. 1981. *Practical Optimization.* Academic Press, London.

Hedlund, P., A. Gustavsson. 1992. Design and Evaluation of Modified Simplex Methods having Enhanced Convergence Ability. *Analytica Chimica Acta* **259** 243-256.

Hollander, M., D.A. Wolfe. 1999. *Nonparametric Statistical Methods.* John Wiley & Sons, New York.

Humphrey, D.G., J.R. Wilson. 1998. A Revised Simplex Search procedure for Stochastic Simulation Response-Surface Optimization. D.J. Medeiros, E.F. Watson, J.S. Carson, M.S. Manivannan, eds. *Proceedings of the 1998 Winter Simulation Conference.* IEEE Press, Piscataway, NJ. 751-759.

Jacobson, S.H., L.W. Schruben. 1989. Techniques for Simulation Response Optimization. *Operation Research Letters* **8** 1-9.

Kleijnen, J.P.C., 1998. Experimental Design for Sensitivity Analysis, Optimization, and Validation of Simulation models. J. Banks, ed. *Handbook of Simulation: Principles, Methodology, Advances, Applications and Practice.* John Wiley & Sons, New York. 173-223.

Lagarias, J.C., J.A. Reeds, M.H. Wright, P.E. Wright. 1998. Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization* **9(1)** 112-147.

Loeve, F., R. Boer, G.J. van Oortmarssen, M. van Ballegooijen, J.D.F. Habbema. 1999. The MISCAN-COLON Simulation Model for the Evaluation of Colorectal Cancer Screening. *Computers and Biomedical Research* **32** 13-33.

McKinnon, K.I.M. 1998. Convergence of the Nelder-Mead Simplex Method to a Nonstationary Point. *SIAM Journal on Optimization* **9(1)** 148-158.

Morgan, E., K.W. Burton. 1990. Optimization Using the Modified Simplex Method. *Chemometrics and Intelligent Laboratory Systems* **7** 209-222.

Morgan, E., S.N. Deming. 1974. Simplex Optimization of Analytical Chemical Methods. *Analytical Chemistry* **46(9)** 1170-1181.

Neddermeijer, H.G., N. Piersma, G.J. van Oortmarssen, J.D.F. Habbema, R. Dekker. 1999. Comparison of Response Surface Methodology and the Nelder and Mead Simplex Method for Optimization in Microsimulation Models. Econometric Institute Report EI-9924/A, Erasmus University Rotterdam, The Netherlands.

Nelder, J.A., R. Mead. 1965. A Simplex Method for Function Minimization. *The Computer Journal* **7** 308-313.

Öberg, T. 1998. Importance of the First Design Matrix in Experimental Simplex Optimization. *Chemometrics and Intelligent Laboratory Systems* **44** 147-151.

Oortmarssen, G.J. van, J.D.F. Habbema, J.Th.N. Lubbe, P.J. van der Maas. 1990. A Model-based Analysis of the HIP Project for Breast Cancer Screening. *International Journal of Cancer* **46** 207-213.

Parker, L.R., M.R. Cave, R.M. Barnes. 1985. Comparison of Simplex Algorithms. *Analytica Chimica Acta* **175** 231-237.

Pflug, G.Ch. 1996. *Optimization of Stochastic Models: The Interface Between Simulation and Optimization.* Kluwer Academic Publishers, Boston.

Phillips, D.A. 1972. A Preliminary Investigation of Function Optimization by a Combination of Methods. *The Computer Journal* **17(1)** 75-79.

Plaisier, A.P., G.J. van Oortmarssen, J.D.F. Habbema, J. Remme, E.S. Alley. 1990. ONCHOSIM: a Model and Computer Simulation Program for the Transmission and Control of Onchocerciasis. *Computer Methods and Programs in Biomedicine* **31(1)** 43-56.

Plaisier, A.P., S. Subramanian, P.K. Das, W. Souza, T. Lapa, A.F. Furtado, C.P.B. van der Ploeg, J.D.F. Habbema, G.J. van Oortmarssen. 1998. The LYMFASIM Simulation Program for Modelling Lymphatic Filariasis and its Control. *Methods of Information in Medicine* **37** 97-108.

Ploeg, C.P.B van der, C. van Vliet, SJ de Vlas, J.O. Ndinya-Achola, L. Fransen, G.J. van Oortmarssen, J.D.F. Habbema. 1998. STDSIM: a Microsimulation Model for Decision Support in STD Control. *Interfaces* **28(3)** 84-100.

Shapiro, S., J.D. Goldberg, G.B. Hutchison. 1974. Lead Time in Breast Cancer Detection and Implications for Periodicity of Screening. *American Journal of Epidemiology* **100** 357-366.

Shavers, C.L., M.L. Parsons, S.N. Deming. 1979. Simplex Optimization of Chemical Systems. *Journal of Chemical Education* **56(5)** 307-309.

Spall, J.C. 1992. Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation. *IEEE Transactions on Automatic Control* **37(3)** 332-341.

Spendley, W., G.R. Hext, F.R. Himsworth. 1962. Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. *Technometrics* **4(4)** 441-461.

Tomick, J.J., S.F. Arnold, R.R. Barton. 1995. Sample Size Selection for Improved Nelder-Mead Performance. C. Alexopoulos, K. Kang, W.R. Lilegdon, D. Goldsman, eds. *Proceedings of the 1995 Winter Simulation Conference.* ACM, Baltimore. 341-345.

Walters, F.H., S.N. Deming 1985. Window Diagrams Versus the Sequential Simplex Method: Which is Correct? *Analytica Chimica Acta* **167** 361-363.

Wiel, P.F.A. van der. 1980. Improvement of the Super-Modified Simplex Optimization Procedure. *Analytica Chimica Acta* **122** 421-433.

Wright, M.H. 1996. Direct Search Methods: Once Scorned, Now Respectable. D. F. Griffiths, G. A. Watson, eds. *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis).* Addison Wesley Longman, Harlow, UK. 191-208.

Table I. Tested extended Nelder and Mead Simplex algorithms*

Criterion	Action			
	Re-evaluate best vertex (EV)	Restart (RS)	Increase simulation size (IS)	Increase number of replications (IR)
Retaining best function value (RV)	RV-EV	–	–	–
Simplex size (SS)	–	SS-RS(ε_{ss})	SS-IS(ε_{ss}, b_s)	SS-IR(ε_{ss}, b_r)
Lack of change (LC)	–	LC-RS(α_{lc}, q)	LC-IS(α_{lc}, q, b_s)	LC-IR(α_{lc}, q, b_r)
Dominant noise (DN)	–	DN-RS(α_{dn})	–	DN-IR(α_{dn}, b_r)

* This table shows the tested extended algorithms, as determined by the criterion used and the subsequent action.

Table II. Tested settings of the optimization parameters

Action / criterion	Settings
Simplex size (SS)	$\varepsilon_{ss} \in \{0, 0.01\}$
Lack of change (LC)	$q = 5; \alpha_{lc} \in \{0.01, 0.05, 0.20\}$
Dominant noise (DN)	$\alpha_{dn} \in \{0.01, 0.05, 0.20\}$
Increase simulation size (IS)	$b_s \in \{1.25, 1.50\}$
Increase number of replications (IR)	$b_r \in \{1.25, 1.50\}$

Table III. Initial settings for the tested functions

Objective function	n	Starting point	Domain	Step sizes
Microsimulation model	3	$(0.4, 0.6, 0.0015)$	$0.01 \leq \lambda \leq 1$ $0 \leq \phi \leq 1$ $0.001 \leq J \leq 0.003$	$(0.1, 0.1, 0002)$
Rosenbrock's function	2	$(-1.2, 1)$	$-25 \leq X_i \leq 25,\ i = 1, 2$	$(5, 5)$
Powell's singular function	4	$(3, -1, 0, 1)$	$-25 \leq X_i \leq 25,\ i = 1.., 4$	$(5, ..., 5)$
Symmetrical Gaussian function	2	$(-100, -100)$	$-250 \leq X_i \leq 250,\ i = 1, 2$	$(50, 50)$
Asymmetrical function	8	$(-5, ..., -5)$	$-10 \leq X_i \leq 10,\ i = 1, ..., 8$	$(2, ..., 2)$
Five-variable paraboloid	5	$(3, -3, 3, -3, 3)$	$-5 \leq X_i \leq 5,\ i = 1, .., 5$	$(1, ..., 1)$

Table IVa. Results for Rosenbrock's function and Powell's singular function

Algorithm	Rosenbrock's function			Powell's singular function		
	$\bar{\epsilon}_{end}$ ($st.dev.$)	T_{end}	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)	$\bar{\epsilon}_{end}$ ($st.dev.$)	T_{end}	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)
BM	0.80 (0.27)	n/a	37 (2)	0.25 (0.18)	n/a	32 (17)
RV-EV	0.76 (0.24)	0	39 (2)	0.29 (0.23)	0	34 (17)
SS-RS(0)	6.79 (2.67)	-	- (0)	40.28 (4.22)	-	- (0)
SS-RS(0.01)	8.31 (7.38)	0	37 (9)	63.70 (0.00)	-	- (0)
LC-RS(0.01,5)	1.07 (0.00)	-	- (0)	166.93 (0.68)	-	- (0)
LC-RS(0.05,5)	1.09 (0.33)	-	87 (1)	97.14 (34.82)	-	- (0)
LC-RS(0.20,5)	0.91 (0.77)	0	74 (6)	35.68 (14.76)	-	- (0)
DN-RS(0.01)	0.94 (0.66)	0	106 (5)	1.72 (1.03)	-	- (0)
DN-RS(0.05)	1.23 (0.76)	0	75 (2)	1.32 (0.85)	-	45 (4)
DN-RS(0.20)	0.70 (0.35)	0	66 (4)	1.25 (0.81)	-	35 (4)
SS-IS(0,1.25)	0.53 (0.20)	+	271 (7)	0.09 (0.07)	+	240 (20)
SS-IS(0,1.5)	0.61 (0.19)	+	271 (4)	0.08 (0.07)	+	253 (20)
SS-IS(0.01,1.25)	0.57 (0.18)	+	295 (6)	0.13 (0.10)	+	344 (20)
SS-IS(0.01,1.5)	0.61 (0.13)	+	318 (2)	0.08 (0.05)	+	318 (20)
LC-IS(0.01,5,1.25)	0.68 (0.19)	0	210 (3)	0.11 (0.10)	+	322 (20)
LC-IS(0.01,5,1.50)	0.63 (0.16)	+	219 (3)	0.07 (0.04)	+	283 (20)
LC-IS(0.05,5,1.25)	0.60 (0.18)	+	326 (4)	0.07 (0.06)	+	286 (20)
LC-IS(0.05,5,1.50)	0.63 (0.14)	0	246 (4)	0.11 (0.10)	+	287 (20)
LC-IS(0.20,5,1.25)	0.60 (0.20)	+	128 (7)	0.13 (0.18)	+	66 (19)
LC-IS(0.20,5,1.50)	0.56 (0.17)	+	217 (5)	0.08 (0.07)	+	69 (20)
SS-IR(0,1.25)	0.62 (0.15)	+	198 (4)	0.13 (0.07)	0	129 (20)
SS-IR(0,1.5)	0.63 (0.24)	+	277 (5)	0.10 (0.10)	+	191 (20)
SS-IR(0.01,1.25)	0.59 (0.19)	+	172 (7)	0.12 (0.10)	+	215 (20)
SS-IR(0.01,1.5)	0.60 (0.17)	+	294 (6)	0.13 (0.10)	+	279 (20)
LC-IR(0.01,5,1.25)	0.54 (0.16)	+	206 (8)	0.11 (0.11)	+	202 (20)
LC-IR(0.01,5,1.50)	0.54 (0.20)	+	224 (8)	0.13 (0.12)	+	236 (20)
LC-IR(0.05,5,1.25)	0.61 (0.12)	+	123 (2)	0.11 (0.08)	+	184 (20)
LC-IR(0.05,5,1.50)	0.62 (0.16)	+	254 (5)	0.14 (0.13)	+	227 (20)
LC-IR(0.20,5,1.25)	0.54 (0.23)	+	97 (8)	0.15 (0.13)	0	46 (19)
LC-IR(0.20,5,1.50)	0.57 (0.13)	+	181 (5)	0.14 (0.14)	+	58 (19)
DN-IR(0.01,1.25)	0.65 (0.25)	0	64 (5)	0.10 (0.09)	+	50 (20)
DN-IR(0.01,1.50)	0.48 (0.24)	+	62 (8)	0.13 (0.09)	+	51 (20)
DN-IR(0.05,1.25)	0.51 (0.25)	+	57 (8)	0.18 (0.14)	0	53 (19)
DN-IR(0.05,1.50)	0.54 (0.15)	+	107 (6)	0.10 (0.11)	+	56 (20)
DN-IR(0.20,1.25)	0.60 (0.25)	+	36 (5)	0.11 (0.10)	+	40 (20)
DN-IR(0.20,1.50)	0.52 (0.23)	+	40 (8)	0.10 (0.08)	+	39 (20)

Table IVb. Results for the Symmetrical Gaussian function and the Asymmetrical function

Algorithm	Symmetrical Gaussian function			Asymmetrical function		
	$\bar{\epsilon}_{end}$ ($st.dev.$)	T_{end}	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)	$\bar{\epsilon}_{end}$ ($st.dev.$)	T_{end}	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)
BM	7.08 (4.23)	n/a	12 (5)	3.24 (1.96)	n/a	- (0)
RV-EV	4.04 (4.47)	0	16 (9)	4.13 (1.81)	0	- (0)
SS-RS(0)	0.89 (2.19)	+	38 (15)	5.78 (4.04)	-	- (0)
SS-RS(0.01)	1.95 (3.49)	0	41 (10)	6.28 (5.11)	-	- (0)
LC-RS(0.01,5)	1.42 (2.16)	+	35 (6)	4.04 (2.99)	0	- (0)
LC-RS(0.05,5)	0.91 (0.82)	+	32 (7)	4.06 (3.11)	0	- (0)
LC-RS(0.20,5)	0.92 (1.92)	+	37 (14)	1.04 (0.62)	+	110 (4)
DN-RS(0.01)	0.86 (2.18)	+	30 (14)	6.76 (5.75)	-	- (0)
DN-RS(0.05)	0.89 (1.04)	+	23 (11)	3.92 (2.07)	0	- (0)
DN-RS(0.20)	1.07 (2.14)	+	28 (12)	3.02 (1.21)	0	- (0)
SS-IS(0,1.25)	4.50 (4.95)	+	80 (10)	2.61 (1.98)	0	650 (3)
SS-IS(0,1.5)	4.67 (4.40)	0	146 (8)	2.20 (1.90)	+	670 (4)
SS-IS(0.01,1.25)	5.07 (4.91)	+	64 (9)	2.67 (2.30)	0	655 (2)
SS-IS(0.01,1.5)	4.20 (4.80)	+	121 (11)	1.34 (0.86)	+	810 (1)
LC-IS(0.01,5,1.25)	5.31 (4.88)	0	135 (8)	2.83 (2.41)	0	- (0)
LC-IS(0.01,5,1.50)	3.57 (4.74)	+	132 (12)	1.37 (1.59)	+	625 (4)
LC-IS(0.05,5,1.25)	6.96 (4.60)	0	149 (6)	4.36 (3.42)	0	- (0)
LC-IS(0.05,5,1.50)	4.54 (4.60)	0	74 (8)	1.17 (1.46)	+	707 (6)
LC-IS(0.20,5,1.25)	4.69 (4.81)	0	70 (7)	5.17 (5.47)	0	- (0)
LC-IS(0.20,5,1.50)	8.02 (3.62)	0	11 (2)	3.31 (2.69)	0	443 (2)
SS-IR(0,1.25)	5.27 (4.60)	0	54 (7)	3.77 (3.36)	0	486 (2)
SS-IR(0,1.5)	5.35 (4.87)	0	84 (9)	1.89 (1.89)	+	468 (4)
SS-IR(0.01,1.25)	4.98 (4.93)	0	27 (9)	2.42 (2.86)	+	488 (3)
SS-IR(0.01,1.5)	6.16 (4.57)	0	76 (6)	1.67 (1.82)	+	543 (3)
LC-IR(0.01,5,1.25)	6.06 (4.62)	0	73 (7)	2.96 (2.63)	0	- (0)
LC-IR(0.01,5,1.50)	4.98 (4.98)	0	152 (10)	1.84 (2.81)	+	595 (1)
LC-IR(0.05,5,1.25)	5.01 (4.89)	0	61 (9)	2.91 (2.49)	0	- (0)
LC-IR(0.05,5,1.50)	4.80 (4.84)	0	42 (9)	1.05 (1.14)	+	575 (6)
LC-IR(0.20,5,1.25)	6.41 (4.77)	0	70 (7)	3.46 (2.23)	0	86 (1)
LC-IR(0.20,5,1.50)	6.33 (4.60)	0	62 (6)	2.51 (1.63)	0	457 (1)
DN-IR(0.01,1.25)	5.51 (4.99)	0	79 (9)	3.56 (2.33)	0	- (0)
DN-IR(0.01,1.50)	3.46 (4.74)	+	67 (13)	2.59 (2.23)	0	411 (1)
DN-IR(0.05,1.25)	3.49 (4.66)	+	88 (12)	1.99 (2.57)	+	235 (7)
DN-IR(0.05,1.50)	2.98 (4.25)	+	72 (12)	2.28 (1.83)	+	226 (1)
DN-IR(0.20,1.25)	5.74 (4.82)	0	74 (7)	2.94 (2.32)	0	149 (2)
DN-IR(0.20,1.50)	3.88 (4.69)	+	96 (12)	2.87 (2.08)	0	- (0)

Table IVc. Results for the Five-variable paraboloid

Algorithm	Five-variable paraboloid		
	$\bar{\epsilon}_{end}$ ($st.dev.$)	T_{end}	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)
BM	0.59 (0.52)	n/a	27 (11)
RV-EV	0.57 (0.36)	0	26 (8)
SS-RS(0)	1.58 (2.06)	-	62 (5)
SS-RS(0.01)	8.85 (6.85)	-	- (0)
LC-RS(0.01,5)	20.30 (4.53)	-	- (0)
LC-RS(0.05,5)	5.69 (6.12)	-	93 (2)
LC-RS(0.20,5)	1.07 (0.68)	0	51 (3)
DN-RS(0.01)	1.90 (1.24)	-	- (0)
DN-RS(0.05)	1.65 (1.12)	-	- (0)
DN-RS(0.20)	0.88 (0.52)	0	62 (6)
SS-IS(0,1.25)	0.19 (0.12)	+	159 (20)
SS-IS(0,1.5)	0.18 (0.12)	+	205 (19)
SS-IS(0.01,1.25)	0.22 (0.19)	+	254 (19)
SS-IS(0.01,1.5)	0.12 (0.05)	+	237 (20)
LC-IS(0.01,5,1.25)	0.14 (0.11)	+	285 (20)
LC-IS(0.01,5,1.50)	0.17 (0.12)	+	265 (20)
LC-IS(0.05,5,1.25)	0.20 (0.13)	+	242 (20)
LC-IS(0.05,5,1.50)	0.17 (0.12)	+	221 (19)
LC-IS(0.20,5,1.25)	0.51 (0.41)	0	115 (12)
LC-IS(0.20,5,1.50)	0.27 (0.24)	+	156 (18)
SS-IR(0,1.25)	0.19 (0.15)	+	73 (19)
SS-IR(0,1.5)	0.15 (0.11)	+	134 (20)
SS-IR(0.01,1.25)	0.15 (0.07)	+	139 (20)
SS-IR(0.01,1.5)	0.14 (0.07)	+	174 (20)
LC-IR(0.01,5,1.25)	0.19 (0.14)	+	180 (19)
LC-IR(0.01,5,1.50)	0.14 (0.12)	+	170 (20)
LC-IR(0.05,5,1.25)	0.22 (0.24)	+	104 (19)
LC-IR(0.05,5,1.50)	0.15 (0.09)	+	164 (20)
LC-IR(0.20,5,1.25)	0.29 (0.17)	+	42 (17)
LC-IR(0.20,5,1.50)	0.32 (0.39)	+	81 (18)
DN-IR(0.01,1.25)	0.18 (0.12)	+	76 (20)
DN-IR(0.01,1.50)	0.18 (0.18)	+	99 (19)
DN-IR(0.05,1.25)	0.24 (0.13)	+	98 (20)
DN-IR(0.05,1.50)	0.19 (0.16)	+	89 (19)
DN-IR(0.20,1.25)	0.26 (0.23)	+	51 (16)
DN-IR(0.20,1.50)	0.35 (0.33)	+	92 (15)

Table V. Results for the Microsimulation Model

Algorithm	$\bar{\epsilon}_{small}$ ($st.dev.$)	T_{small}	$\bar{E}_{small,0.5}(\times 10^5)$ ($r_{small,o.5}$)	$\bar{\epsilon}_{end}$ ($st.dev.$)	T_{end}	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)
BM[1]	0.62 (0.69)	0	17 (12)	0.70 (0.68)	0	15 (10)
BM[5]	0.58 (0.73)	0	21 (13)	0.67 (0.75)	0	21 (12)
RV-EV[5]	0.63 (0.89)	0	21 (13)	0.74 (0.89)	0	21 (11)
SS-RS(0)[5]	1.08 (0.56)	-	110 (3)	1.19 (0.57)	-	122 (2)
SS-RS(0.01)[5]	1.74 (0.79)	-	0 (0)	2.07 (0.89)	-	0 (0)
LC-RS(0.01,5)[5]	2.02 (1.27)	-	0 (0)	2.06 (1.40)	-	0 (0)
LC-RS(0.05,5)[5]	1.91 (0.82)	-	0 (0)	2.21 (1.12)	-	0 (0)
LC-RS(0.20,5)[5]	0.39 (0.30)	0	38 (14)	0.68 (0.59)	0	48 (9)
DN-RS(0.01)	1.16 (0.87)	-	90 (5)	1.40 (0.93)	-	90 (5)
DN-RS(0.05)	1.18 (1.11)	-	50 (9)	1.29 (1.09)	-	51 (7)
DN-RS(0.20)	0.85 (0.69)	0	66 (9)	1.08 (0.68)	0	75 (5)
SS-IS(0,1.25)	0.02 (0.02)	+	60 (20)	0.04 (0.04)	+	60 (20)
SS-IS(0,1.5)	0.03 (0.03)	+	102 (20)	0.06 (0.05)	+	102 (20)
SS-IS(0.01,1.25)	0.04 (0.05)	+	129 (20)	0.07 (0.05)	+	129 (20)
SS-IS(0.01,1.5)	0.06 (0.08)	+	156 (20)	0.09 (0.08)	+	156 (20)
LC-IS(0.01,5,1.25)	0.03 (0.03)	+	82 (20)	0.06 (0.05)	+	82 (20)
LC-IS(0.01,5,1.50)	0.05 (0.04)	+	129 (20)	0.07 (0.05)	+	129 (20)
LC-IS(0.05,5,1.25)	0.05 (0.08)	+	81 (20)	0.08 (0.08)	+	81 (20)
LC-IS(0.05,5,1.50)	0.03 (0.03)	+	70 (20)	0.05 (0.05)	+	70 (20)
LC-IS(0.20,5,1.25)	0.24 (0.50)	+	16 (18)	0.28 (0.49)	+	16 (18)
LC-IS(0.20,5,1.50)	0.38 (0.77)	+	33 (16)	0.41 (0.79)	+	16 (15)
SS-IR(0,1.25)	0.28 (0.51)	0	34 (18)	0.31 (0.52)	+	32 (17)
SS-IR(0,1.5)	0.06 (0.07)	+	54 (20)	0.10 (0.08)	+	54 (20)
SS-IR(0.01,1.25)	0.03 (0.03)	+	70 (20)	0.06 (0.05)	+	70 (20)
SS-IR(0.01,1.5)	0.02 (0.02)	+	86 (20)	0.04 (0.04)	+	86 (20)
LC-IR(0.01,5,1.25)	0.08 (0.13)	+	46 (20)	0.10 (0.13)	+	46 (20)
LC-IR(0.01,5,1.50)	0.04 (0.04)	+	86 (20)	0.05 (0.05)	+	86 (20)
LC-IR(0.05,5,1.25)	0.18 (0.45)	+	26 (19)	0.21 (0.46)	+	26 (18)
LC-IR(0.05,5,1.50)	0.07 (0.08)	+	62 (20)	0.08 (0.09)	+	62 (20)
LC-IR(0.20,5,1.25)	0.36 (0.64)	0	20 (16)	0.39 (0.64)	+	20 (16)
LC-IR(0.20,5,1.50)	0.49 (1.04)	0	33 (17)	0.55 (1.06)	0	33 (17)
DN-IR(0.01,1.25)	0.17 (0.26)	+	52 (17)	0.20 (0.25)	+	52 (17)
DN-IR(0.01,1.50)	0.04 (0.06)	+	40 (20)	0.06 (0.06)	+	40 (20)
DN-IR(0.05,1.25)	0.19 (0.36)	+	65 (19)	0.22 (0.37)	+	65 (19)
DN-IR(0.05,1.50)	0.20 (0.48)	+	46 (18)	0.23 (0.49)	+	46 (18)
DN-IR(0.20,1.25)	0.36 (0.73)	+	23 (17)	0.39 (0.74)	+	23 (17)
DN-IR(0.20,1.50)	0.23 (0.29)	+	41 (17)	0.26 (0.30)	+	41 (17)

Table VI. Results benchmark algorithm for different initial simulation sizes.*

Algorithm	Initial simulation size S_0	$\bar{\epsilon}_{end}$ (st.dev.)	$\bar{E}_{end,0.5}(\times 10^5)$ ($r_{end,0.5}$)
Rosenbrock's function	10000	0.80 (0.27)	37 (2)
	20000	0.65 (0.20)	75 (5)
	50000	0.61 (0.13)	200 (7)
	100000	0.52 (0.20)	370 (9)
Powell's singular function	10000	0.25 (0.18)	32 (17)
	20000	0.18 (0.19)	60 (19)
	50000	0.15 (0.11)	165 (20)
	100000	0.10 (0.08)	307 (20)
Symmetrical Gaussian function	10000	7.08 (4.23)	12 (5)
	20000	5.88 (4.70)	25 (6)
	50000	3.74 (4.65)	64 (12)
	100000	2.56 (4.37)	102 (15)
Asymmetrical function	10000	3.24 (1.96)	- (0)
	20000	3.02 (2.44)	93 (1)
	50000	2.36 (2.61)	303 (4)
	100000	1.11 (1.35)	673 (6)
Five-variable paraboloid	10000	0.59 (0.52)	27 (11)
	20000	0.39 (0.43)	52 (17)
	50000	0.21 (0.14)	115 (19)
	100000	0.14 (0.11)	226 (20)
Microsimulation model	10000	0.67 (0.75)	21 (12)
	20000	0.39 (0.63)	42 (18)
	50000	0.17 (0.25)	75 (19)
	100000	0.10 (0.12)	140 (20)

* Initial number of replications $N_0 = 5$.
The results for $S_0 = 10000$ and $N_0 = 5$ are taken from Tables IV and V.