Fairer Benchmarking of Optimization Algorithms via Derivative Free Optimization

W. L. Hare* Y. Wang†

October 13, 2010

Abstract

Research in optimization algorithm design is often accompanied by benchmarking a new algorithm. Some benchmarking is done as a proof-of-concept, by demonstrating the new algorithm works on a small number of difficult test problems. Alternately, some benchmarking is done in order to demonstrate that the new algorithm in someway out-performs previous methods. In this circumstance it is important to note that algorithm performance can heavily depend on the selection of a number of user inputed parameters. In this paper we begin by demonstrating that if algorithms are compared using arbitrary parameter selections, results do not just compare the algorithms, but also compare the authors' ability to select good parameters. We further present a novel technique for generating and providing results using each algorithm's "optimal parameter selection". These optimal parameter selections can be computed using modern derivative free optimization methods and generate a framework for fairer benchmarking between optimization algorithms.

Keywords: benchmarking, optimal parameter selection, Derivative Free Optimization

1 Introduction

Algorithm design is one of the pillars of modern optimization research. In order for a new algorithm to be considered successful, authors are generally encouraged to benchmark their algorithm against previously established methods. This consists of running several different algorithms against a collection of test problems and comparing the results. In this work, we examine an issue that arise during benchmarking and discuss a technique that can help researchers to provide fairer benchmarking of algorithms. In particular, we explore the question of how to avoid using arbitrary parameter selections during benchmarking.

In [AO06], Audet and Orban point out that most optimization algorithms rely on a large quantity of input parameters that are independent of the optimization problem. For example, most optimization codes include a parameter how step length decreases as it is deemed too large. In this work we demonstrate that if algorithms are compared using arbitrary parameter selections, results do not just compare the algorithms, but also compare the authors' ability to select good parameters. Thus it is unfair to compare two algorithms using arbitrary parameter selections. Working on the ideas of Audet and Orban, we suggest a novel methodology in which each algorithm's "optimal parameter selection" is determined before benchmarking is performed.

*Dept of Math, Stats, & Physics, UBC O, warren.hare@ubc.ca.
†Dept of App Math, UWO, ywang767@uwo.ca.

In order to illustrate the ideas laid out in this work, we explore two methods of adapting the Nelder-Mead algorithm to include a very basic line search. It should be noted that the goal of this research is not to advance the Nelder-Mead algorithm. We select the Nelder-Mead algorithm for our illustrations for two reasons. First, despite criticisms on its effectiveness (see for example [McK99]), the Nelder-Mead algorithm is still one of the most widely applied optimization algorithms for applied problems [CSV09]. Second, as Nelder-Mead algorithm is not currently a major focus of active research, we are not biased to prove or disprove its effectiveness.

The remainder of this paper is organized as follows. In Section 2 we outline the ideas behind optimal parameter selection, and outline the algorithms we shall use for our numerical examples. In Section 3 we discuss the test problems used in this paper, and review the construction and interpretation of *performance profiles*. In Section 4 we demonstrate that arbitrary parameter selections can result in inaccurate benchmarking of optimization algorithm. We then detail our method for determining the optimal parameter selections (Subsection 4.2) based on the work of Audet and Orban and argue that this parameter selection provides a fairer benchmark of the algorithms. We finish with some concluding remarks.

2 Overview and test algorithms

Considerable researcher into optimization focuses on the construction of computer implemented algorithms to solve minimization problems. Most algorithms include a variety of input parameters, to be chosen by the user, which represent aspects such as step-length reduction. With regards to these parameters, most algorithms can be proven to converge in a generally sense. That is, algorithms have some constraints on the parameter choice, but generally the number of options is continuous in nature. For example, a step-length reduction parameters might be required to be in $\tau \in (0, 1)$, but it is left to the user to determine whether to be aggressive (e.g., $\tau = 0.01$), neutral (e.g., $\tau = 0.5$), or conservative (e.g., $\tau = 0.95$).

From a purely theoretical point of view, this is a very powerful aspect of optimization research. However, from a practical point of view, this generates some difficulty when benchmarking algorithms. The benchmarking process consists of running several different algorithms against a collection of test problems and comparing the results. In benchmarking an algorithm, authors pursue one of two goals.

In some benchmarking research, the goal is to show a proof-of-concept that the algorithm can be implemented and work in practice. For this goal, it suffices to select any parameter selection that results in reasonable performance.

In other benchmarking research, the goal is to show that the new algorithm in someway out-performs previous methods. For this goal, the choice of parameters can be crucial. We shall see, in Section 4 Subsection 4.1, that small changes in algorithm parameters can vastly alter an algorithm's performance. This leads naturally to the question of how to find good parameter selections for benchmarking, and how to find fair parameter selections for all algorithms.

One option lies in recent results by Audet and Orban involving Derivative Free Optimization [AO06]. Derivative Free Optimization (DFO) methods are designed to minimize objective functions for which derivative information is not available. (We refer readers to the book of Conn, Scheinberg, and Vicente for general information on DFO methods [CSV09].) In [AO06], Audet and Orban demonstrate that parameter selection can be phrased as a DFO problem where the inputs are the parameters and the goal is to minimize run time.

Of course, in order to apply this idea it is necessary to solve each optimization problem multiple times. In applied research this appears impractical, however Audet and Orban demonstrate how the use of a collection of surrogate functions (functions that are small versions of the original problem and maintain similar problem structure) can lead to a good parameter selection for the original problem.

In the case of benchmarking, the issue of solving the collection of test problems multiple times is not serious. Indeed, many researchers will attest that they "tinker" with algorithm parameters before benchmarking in order to attain better results. In this paper we propose that instead of "tinkering", researchers apply the techniques of Audet and Orban to formally find the "optimal parameter selection" for each algorithm being benchmarked, and then compare algorithms under their optimal performance.

In this paper we will utilize the classic Nelder-Mead algorithm and two simple variations in order to demonstrate the necessity of proper benchmarking techniques. For the sake of completeness, our implementation of the Nelder-Mead algorithm and details on our two variants are discussed in the remainder of this Section.

All algorithms were implemented in MATLAB 7.6.0 (R2008a).

2.1 Classic Nelder-Mead (CNM)

One of the oldest methods in DFO literature is the famous Nelder-Mead algorithm [NM65]. Due to its ease in implementation, range of applications, and general effectiveness, the Nelder-Mead algorithm is still one of the most widely applied optimization algorithms [CSV09, Chpt 8]. The Nelder-Mead algorithm begins with a simplex of potential solution points to an optimization problem. After evaluating the objective function at each point, the algorithm operates a reflection, an expansion, a contraction (inside or outside) or a shrink on the simplex. The goal is to replace the current worst point of the simplex with a newly generated better point (if there is any) in order to have a potentially better pool of possible solutions. If no better point can be found, the shrink operation reduces the search area so a more precise local solution can be determined. A pseudo-code version of the Nelder-Mead algorithm appears in Algorithm 2.1.

Algorithm 2.1 (Classic Nelder-Mead Algorithm: CNM)

0. Initialization: *Input an initial simplex* $SP = \{x^0, x^1, x^2, ..x^n\}$*, a reflection parameter* $0 < \alpha$*, an expansion parameter* $1 < \gamma$*, a contraction parameter* $0 < \beta < 1$*, a shrink parameter* $0 < \delta < 1$*, and stopping criteria.*

1. Order: *Determine the simplex points* x^b*,* x^s*, and* x^w *such that their function values* $f^b = f(x^b), f^s = f(x^s)$*, and* $f^w = f(x^w)$ *are a best, a second worst, and a worst.*

2. Stopping: *Check stopping criteria and terminate if desired.*

3. Reflect: *Compute the reflection point and value*

$$x^r = c + \alpha(c - x^w),\ f^r = f(x^r) \quad \textit{where} \quad c = \frac{1}{n} \sum_{x^i \in SP \setminus x^w} x^i.$$

If $f^b \leq f^r < f^s$ *replace* x^w *with* x^r *and return to Order.*
If $f^r < f^b$ *proceed to Expand.*
Otherwise ($f^r \geq f^s$) *proceed to Contract.*

4. Expand: ($f^r < f^b$) *Compute the expansion point and value*

$$x^e = c + \gamma(c - x^w), f^e = f(x^e).$$

If $f^e < f^r$ *replace* x^w *with* x^e *and return to Order.*
Otherwise replace x^w *with* x^r *and return to Order.*

5. Contract: ($f^r \geq f^s$)
If $f^s \leq f^r < f^w$ *then compute the outside contraction point and value*

$$x^c = c + \beta(c - x^w),\ f^c = f(x^c).$$

If $f^c \leq f^r$ replace x^w with x^c and return to Order,
Otherwise continue to Shrink.

If $f^w \leq f^r$ then compute the inside contraction point and value

$$x^c = c + \beta(x^w - c), \ f^c = f(x^c).$$

If $f^c < f^w$ replace x^w with x^c and return to Order,
Otherwise continue to Shrink.

6. Shrink: *Reconstruct the simplex by setting*

$$x^i = x^b + \delta(x^i - x^b), \quad \text{for } i = 0, 1, 2, ...n.$$

Return to Order.

Various implementations of Nelder-Mead have arisen in the past 40 years, each of which differs slightly from the original Nelder-Mead algorithm. Our implementation differs from the original in several manners. Most importantly, [NM65] does not consider the shrink parameter δ as a parameter, instead defining it to be 0.5. We also differ from [NM65] in a variety of small changes. For example, we accept the reflection point when $f^b \leq f^r < f^s$, while [NM65] accepted the reflection point when $f^b \leq f^r \leq f^s$. Other changes of the same magnitude are also present. Most varieties of the Nelder-Mead algorithm differ from the [NM65] to a similar level (compare [NM65], [McK99], and [CSV09] for example).

In Step 0, Initialization, we are required to input a starting simplex. However, most test problems, in particular those in [MGH81], provide a single starting point $x^0 \in \mathbb{R}^n$. One simple way of generating a starting simplex from a single point is to generate

$$x^i = x^0 + e^i \text{ for } i = 1, 2, ...n,$$

where $\{e^i\}_{i=1}^n$ is an orthogonal set of unit direction vectors. We use this method, with the standard set of orthogonal unit vectors (e^i contains a 1 in the i^{th} position and 0 elsewhere).

In Step 1, Order, we use the term "a" worst point (second worst, best) as it is possible that a tie may occur. In our implementation, points just added to the simplex are favoured over points that were previously in the simplex. Further ties (as could occur after a Shrink step) are broken by favouring the point with the lowest index.

In Step 2, Stopping, we employ three stopping criteria. First, we stop as soon as more than 6000 function evaluations are detected (note this may result in slightly over 6000 function evaluations if, for example, a shrink step occurs at the final iteration). This ensures eventual stoppage regardless of the test problem. Second, we check the diameter of the simplex and terminate if the simplex becomes too small ($< 10^{-6}$). Finally, we examine simplex degeneracy by implementing the simplex degeneracy test found in [CSV09] (with tolerance of 10^{-12}).

2.2 Double Expansion Nelder-Mead (DENM)

In Step 4, Expand, the reflection point has demonstrated objective function improvement over the current best point. This effectively demonstrates that $c - x^w$ is a descent direction from the point x^w. The expansion step allows the Nelder-Mead algorithm to explore further in that direction to a limited distance. In our first variant on the Nelder-Mead algorithm, we allow for a second expansion step when the first expansion point is successful.

Algorithm 2.2 (Double Expansion Nelder-Mead: DENM)

Replace Step 4 in Algorithm 2.1 with

4. Expand (DENM): *($f^r < f^b$) Compute the expansion point and value*

$$x^e = c + \gamma(c - x^w), f^e = f(x^e).$$

If $f^e \geq f^r$ then replace x^w with x^r and return to Order.

Otherwise ($f^e < f^r$), compute the second expansion point

$$x^{e2} = c + \gamma(\gamma(c - x^w))$$

and continue to 4b.

4b. Expand II: *If $(SP \setminus x^w) \cup x^{e2}$ is nondegenerate then compute the second expansion value*

$$f^{e2} = f(x^{e2}).$$

Otherwise set $f^{e2} = \infty$.
If $f^{e2} < f^e$ then replace x^w with x^{e2} and return to Order.
Otherwise ($f^{e2} \geq f^e$, $f^e < f^r$) replace x^w with x^e and return to Order.

In DENM the second expansion point is only adopted if the resulting geometry is nondegenerate and the function value at this new point is improved. This helps prevent the simplex from becoming degenerate too quickly. We perform a check on the new simplex geometric before evaluating the function at the second expansion point in order to avoid unnecessary function evaluations. As in the full algorithm, in Step 4b we implement the simplex degeneracy test from [CSV09] (with tolerance of 10^{-12}).

2.3 Double Expansion Double Contraction Nelder-Mead (DEDCNM)

This variant is similar to DENM, however, we allow double contraction in additional to double expansion.

Algorithm 2.3 (Double Expansion Double Contraction Nelder-Mead: DEDCNM)

Replace Step 4 in Algorithm 2.1 with Step 4 of DENM (Algorithm 2.2).

Replace Step 5 in Algorithm 2.1 with

5. Contract: *($f^r \geq f^s$)*
If $f^s \leq f^r < f^w$ then compute the outside contraction point and value

$$x^c = c + \beta(c - x^w), \ f^c = f(x^c).$$

If $f^c > f^r$ continue to Shrink
Otherwise ($f^c \leq f^r$), compute the second outside contraction point

$$x^{c2} = c + \beta(\beta(c - x^w))$$

and continue to 5b

If $f^w \leq f^r$ then compute the inside contraction point and value

$$x^c = c + \beta(x^w - c), \ f^c = f(x^c).$$

If $f^c \geq f^w$ continue to Shrink
Otherwise ($f^c < f^w$), compute the second inside contraction point

$$x^{c2} = c + \beta(\beta(x^w - c))$$

and continue to 5b

5b. Contract II: *If* $(SP \setminus x^w) \cup x^{c2}$ *is nondegenerate then compute the second contraction value*

$$f^{c2} = f(x^{c2}).$$

Otherwise set $f^{c2} = \infty$.
If $f^{c2} < f^c$ *then replace* x^w *with* x^{c2} *and return to Order.*
Otherwise replace x^w *with* x^c *and return to Order.*

Similar to DENM, the second expansion or the second contraction point is only adopted if the resulting geometry is nondegenerate and the function value at the new point is improved.

3 Test Problems and Performance Profiles

For our tests we use the problems listed in [MGH81][1]. In [MGH81], test problems are listed with a dimension (number of input variables) and a number of sub-functions; or a range of possible dimensions and number of sub-functions. Sub-functions describe the complexity of the problem within a fixed dimension. In particular, each test problem is given of the form $\sum_{i=1}^{m}(f_i)^2$, where m is the number of sub-functions. The problems examined are listed in Table 1, Appendix A, along with the dimension and number of sub-functions each uses. Problem number is used for referencing, and refers to the problem number as provided in [MGH81]. In some cases, the problem listed in [MGH81] has options on the dimension or number of sub-functions. In such cases we report the numbers that we used. For further details on these test problems we refer readers to [MGH81].

In order to provide a visual comparison of each algorithms performance, we employ *performance profiles* as follows. Let $t_{p,s}$ represent the number of function evaluations required by solver s to solve problem p. If the solver fails to solve the problem then $t_{p,s}$ is set to infinity. From the values $t_{p,s}$ we generate the performance ratio value

$$r_{p,s} = \frac{t_{p,s}}{\min_s\{t_{p,s}\}}$$

for each solver and each problem. Performance profiles now graphs for each solver the function $\rho_s(\tau)$ defined by

$$\rho_s(\tau) = \frac{1}{P}\big|\{p : r_{p,s} \leq \tau\}\big|,$$

where P is the number of test problem and $|\{\cdot\}|$ is the number of elements in the set $\{\cdot\}$. Thus $\rho_s(\tau)$ "is the probability for solver s that a performance ratio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$ of the best possible ratio" [DM02].

At the far left of the graphs ($\tau = 1$) each solver begins at the value equal to the portion of test problems that it solved with the least number of function evaluations. At the far right ($\tau \to \infty$) each solver converges to the portion of test problems that it successfully solved. In general solver s_1 is performing better than s_2 if its graph is higher (i.e. $\rho_{s_1}(\tau) > \rho_{s_2}(\tau)$ for most τ).

4 Optimal parameters selections

The purpose of this paper is to demonstrate that benchmarking optimization algorithms using arbitrary parameter selections can create inconsistent results, and to suggest a simple solution

[1]Some erratums to the [MGH81] problem list can be found at [KK97]. We implement these as appropriate.

to this problem. To demonstrate the flaw with arbitrary parameter selections, we compare CNM to the two variant describe in Subsections 2.2 and 2.3 using several different selections for the parameters α, γ, β, and δ. Stopping criteria, discussed in Subsection 2.1, are consistent across algorithms, and tuned to stop when the algorithm believes it has correctly determined the solution to 6 digits of accuracy. A problem is considered solved if the relative error, defined by

$$RE := \frac{|f^{min} - f^*|}{|f^*| + 1}, \tag{1}$$

where f^{min} is the found minimum and f^* is the true global minimum, is less than 10^{-6}.

4.1 Comparisons Across Arbitrary Parameter Selections

In the next three examples we show that CNM can perform worse than, similar to, or better than DENM, simply by considering different parameter selections.

Example 4.1 (CNM is worse than DENM ($\alpha = 1$, $\gamma = 1.9$, $\beta = 0.6$, and $\delta = 0.6$))

We benchmark CNM and DENM using the parameter selection $\alpha = 1$, $\gamma = 1.9$, $\beta = 0.6$, and $\delta = 0.6$ (for both algorithms). Complete test results can be found in Table 2, Appendix A. In Figure 1 we provide the performance profile for this test.

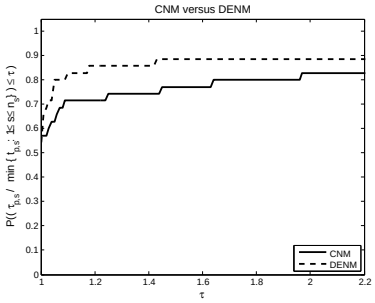

Figure 1: Performance profile of Classic Nelder-Mead versus Double Expansion Nelder-Mead using $\alpha = 1$, $\gamma = 1.9$, $\beta = 0.6$, and $\delta = 0.6$.

Examining Figure 1 we see that the graph of CNM is never above the graph of DENM, so overall the performance profile suggests that CNM is worse than DENM.

Example 4.2 (CNM and DENM are similar ($\alpha = 1$, $\gamma = 1.9$, $\beta = 0.5$, and $\delta = 0.6$))

We benchmark CNM and DENM using the parameter selection $\alpha = 1$, $\gamma = 1.9$, $\beta = 0.5$, and $\delta = 0.6$. Complete test results can be found in Table 3, Appendix A. In Figure 2 we provide the performance profile for this test.

Examining Figure 2 we see that CNM begins above DENM, but ends below. The two graphs cross at about $\tau = 1.4$. The performance profile suggests that DENM is slightly more robust than CNM, but slightly worse in speed. Overall the two algorithms are quite comparable.

Example 4.3 (CNM is better than DENM ($\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$))

We benchmark CNM and DENM using the parameter selection $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$. Complete test results can be found in Table 4, Appendix A. In Figure 3 we provide the performance profile for this test.

Since the graph of CNM is above the graph of DENM at all times, the performance profile suggests that CNM is better than DENM.

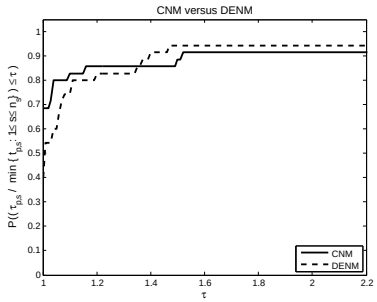

Figure 2: Performance profile of Classic Nelder-Mead versus Double Expansion Nelder-Mead using $\alpha = 1$, $\gamma = 1.9$, $\beta = 0.5$, and $\delta = 0.6$.

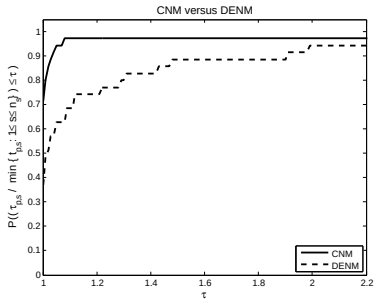

Figure 3: Performance profile of Classic Nelder-Mead versus Double Expansion Nelder-Mead using $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$.

Our next example quickly shows that the same inconsistency of results can be achieved for DEDCNM.

Example 4.4 (comparing CNM and DEDCNM) *We benchmark CNM and DEDCNM using three parameter selections:*

i) $\alpha = 1.1, \gamma = 2, \beta = 0.8,$ *and* $\delta = 0.5,$

i) $\alpha = 1, \gamma = 2, \beta = 0.7,$ *and* $\delta = 0.5,$ *and*

i) $\alpha = 1, \gamma = 2, \beta = 0.5,$ *and* $\delta = 0.5.$

Complete test results can be found in Tables 5, 6, and 7 respectively. In Figure 4 we provide the three performance profiles resulting from these parameter selections.

Examining the performance profiles we see that DEDCNM can appear better than, similar to, or worse than CNM depending on the parameter selection chosen.

4.2 Comparisons Across Optimal Parameter Selections

The examples above demonstrate that comparing two codes using arbitrary parameter selections can easily lead to inconsistent conclusions. In particular, such an approach results in benchmarking not just the algorithms, but also the authors' ability to provide good parameters. (The examples above can be made to be even more pronounced by using difference parameter selections for each algorithm.) To avoid this problem we propose seeking out the optimal parameter

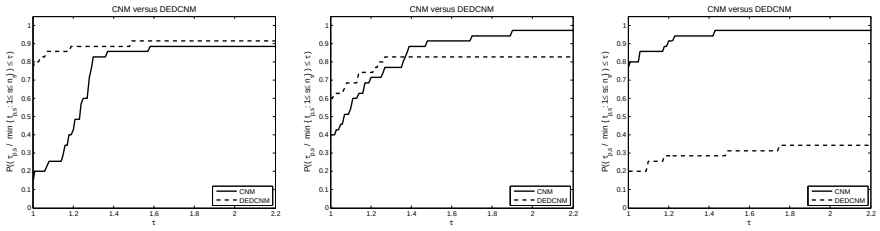

Figure 4: Performance profile of Classic Nelder-Mead versus Double Expansion Double Contraction Nelder-Mead using: $\alpha = 1.1$, $\gamma = 2$, $\beta = 0.8$, and $\delta = 0.5$ (left); $\alpha = 1$, $\gamma = 2$, $\beta = 0.7$, and $\delta = 0.5$ (middle); and $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$ (right).

selection for each of the algorithms and then re-benchmark using this selection. We do this using techniques from [AO06].

In our case, we generate an objective function whose input is the four parameters used in each algorithm and whose output is the total number of function evaluations required with a penalty for each unsolved test problem. The penalty is applied by resetting the number of function evaluations to 7500 for any test problem that is unsolved. (Recall that algorithms were terminated after 6000 function evaluates were detected.) A problem is considered solved if the relative error is less than 10^{-6}. A second penalty function is employed to enforce parameters to stay within the bounds laid out in Algorithm 2.1, Step 0, Initialization. Thus we have the following function,

$$
\begin{array}{rcccl}
F & : & \mathbb{R}^4 & \rightarrow & \mathbb{R} \\
 & : & (\alpha,\beta,\gamma,\delta) & \rightarrow & \left\{ \begin{array}{rl} \text{function evaluations used to} & \text{if } 0<\alpha, 0<\beta<1, \\ \text{solve all problems, plus penalty} & 1<\gamma, 0<\delta<1 \\ \text{for unsolved problems} & \\ & \\ 10^8 & \text{otherwise} \end{array} \right.
\end{array}
$$

Note that the objective function F is based on finding parameters that provide a globally good solution rate and time. In particular, we do not seek to find optimal parameters for each algorithm-problem pair. If we took that approach, then we would very likely find different parameters for each pairing, and solve many problems in a highly unrealistic manner.

We employed the CNM algorithm to minimize F over all valid parameter selections. This process was repeated for each of the algorithms.
The resulting parameters for the Classic Nelder-Mead algorithm are

$$
\begin{array}{rl}
Reflection: & \alpha = 0.987481 \\
Expansion: & \gamma = 2.115042 \\
Contraction: & \beta = 0.519800 \\
Shrink: & \delta = 0.465897.
\end{array}
$$

The resulting parameters for Double Expansion Nelder-Mead are

$$
\begin{array}{rl}
Reflection: & \alpha = 1.009165 \\
Expansion: & \gamma = 1.941588 \\
Contraction: & \beta = 0.508081 \\
Shrink: & \delta = 0.489064.
\end{array}
$$

The resulting parameters for Double Expansion Double Contraction Nelder-Mead are

$$\begin{aligned} Reflection: &\quad \alpha = 1.129736 \\ Expansion: &\quad \gamma = 2.790789 \\ Contraction: &\quad \beta = 0.782744 \\ Shrink: &\quad \delta = 0.201809. \end{aligned}$$

After determining the optimal parameters for each variant of the Nelder-Mead algorithm, we test each algorithm against the 35 test problems listed in Table 1. As before, a problem is considered solved if the relative error is less than 10^{-6}. In order to demonstrate the power of optimal parameter selections (over default parameter selections) we also benchmark all variants using the parameter selection $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$.

Detailed results can be found in Tables 8 and 9 in Appendix A. In Figure 5 we present the performance profiles for CNM and the two variants of Nelder-Mead tested using default and optimal parameter selections.

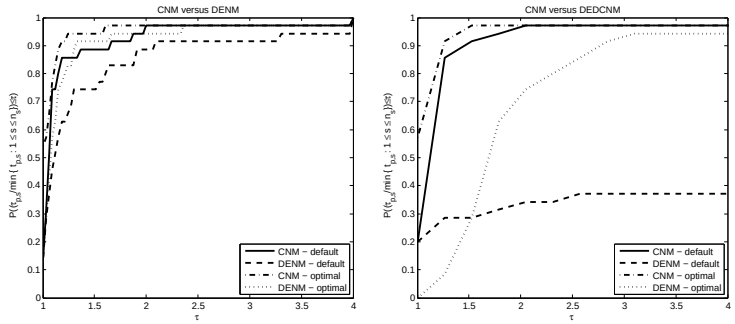

Figure 5: Performance profiles using default and optimal parameter selections.
Left: Classic Nelder-Mead versus Double Expansion Nelder-Mead.
Right: Classic Nelder-Mead versus Double Expansion Double Contraction Nelder-Mead.

In all algorithms the optimal parameter selection shows improvement over the default selection. This reinforces the point that good parameter selection can change the apparent quality of an algorithm. In both profiles we see that CNM using optimal parameter selection is a clear winner over all other algorithms.

In the left profile, we see that DENM using the optimal parameter selection is fairly comparable to CNM using the default parameter selection. This again shows the importance of comparing algorithms using optimal parameter selections. In particular, since DENM is a new algorithm, the authors could have tweaked its parameters while keeping CNM at default parameters and then suggested the two algorithms were comparable. This selective tweaking may even seem reasonable, as often perviously designed algorithms are not programmed by the benchmarkers, so tweaking their parameters can be an onerous task.

Examining the right profiles, it is clear that DEDCNM is notably worse than CNM. In particular, DEDCNM has lower profiles using optimal parameter selections than CNM using default parameters.

(On a side note, the performance profiles in Figure 5 clearly show that Nelder-Mead algorithm does not benefit from either of the line search additions tested in this research.)

5 Concluding remark

The method presented in this paper provides a novel technique for generating fairer benchmarking of optimization algorithms. However, it should be noted that the optimal parameter selection methods for benchmarking algorithms described in this paper will only give optimal parameter selections for the particular test set used in parameter generation. As such, the optimal parameters should not necessarily be used to solve any optimization problem. Methods for determining optimal parameter selections for hard problems, by use of simple surrogate problems, is discussed in [AO06].

Before closing, we should note that there is no reason to expect the optimal parameter selection problem to be convex. Therefore it is likely that the initial point for your parameter selection problem will impact the resulting "optimal parameter selection". Future research should examine this issue, try to determine when the optimal parameter selection problem is convex, and research methods of working with a nonconvex optimal parameter selection problem.

Finally, we remark that in building our optimal parameter objective function, we placed a heavy penalty on unsolved problems. This naturally results in seeking parameters that favour robustness over speed. Changing this is simple, and should be done with care. The important part being simply to ensure each algorithm's optimal parameters are computed using the same optimal parameter objective function.

Acknowledgements

The authors would like to thank the IRMACS Centre at Simon Fraser University for providing technical infrastructure and support for this research. The authors would like to thank Dr. M. Macklem for useful feedback during the final stages of this research. Hare research supported in part by NSERC DG, and the MoCSSy CTEF SFU. Wang research supported in part by NSERC DG.

References

[AO06] C. Audet and D. Orban. Finding optimal algorithmic parameters using derivative-free optimization. *SIAM J. Optim.*, 17(3):642–664 (electronic), 2006.

[CSV09] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*, volume 8 of *MPS/SIAM Book Series on Optimization*. SIAM, 2009.

[DM02] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2, Ser. A):201–213, 2002.

[KK97] A. Kuntsevich and F. Kappel. Moré set of test functions. `http://www.uni-graz.at/imawww/kuntsevich/solvopt/results/moreset.html`, 1997.

[McK99] K. I. M. McKinnon. Convergence of the Nelder-Mead simplex method to a nonstationary point. *SIAM J. Optim.*, 9(1):148–158 (electronic), 1999.

[MGH81] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Software*, 7(1):17–41, 1981.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Comp. J.*, 7(4):308–313, 1965.

[Tor97] V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7(1):1–25, 1997.

A Tables

#	Problem Name	Dimension	Sub-functions	min ≈
(1)	Rosenbrock function	2	2	0
(2)	Freudenstein and Roth function	2	2	0
(3)	Powell badly scaled function	2	2	0
(4)	Brown badly scaled function	2	3	0
(5)	Beale function	2	3	0
(6)	Jennrich and Sampson function	2	2	1.24363E+02
(7)	Helical valley function	3	3	0
(8)	Bard function	3	15	8.21488E-03
(9)	Gaussian function	3	15	1.12794E-08
(10)	Meyer function	3	16	8.79459E+01
(11)	Gulf research and development	3	3	0
(12)	Box three-dimensional function	3	4	0
(13)	Powell singular function	4	4	0
(14)	Wood function	4	6	0
(15)	Kowalik and Osborne function	4	11	3.07506E-04
(16)	Brown and Dennis function	4	20	8.58223E+04
(17)	Osborne 1 function	5	33	5.46490E-05
(18)	Biggs EXP6 function	6	13	0
(19)	Osborne 2 function	11	65	4.01378E-02
(20)	Watson function	6	31	2.28768E-03
(21)	Extended Rosenbrock function	6	6	0
(22)	Extended Powell singular function	12	12	0
(23)	Penalty function I	4	5	2.24998E-05
(24)	Penalty function II	4	8	9.37630E-06
(25)	Variably dimensioned function	7	9	0
(26)	Trigonometrix function	6	6	0
(27)	Brown almost-linear function	4	4	0
(28)	Discrete boundary value function	7	7	0
(29)	Discrete integral equation function	7	7	0
(30)	Broyden tridiagonal function	7	7	0
(31)	Broyden banded function	5	5	0
(32)	Linear function - full rank	4	8	4
(33)	Linear - rank 1	4	8	1.64706E+00
(34)	Linear - rank 1 with 0 columns & rows	4	8	3.15385E+00
(35)	Chebyquad function	2	2	0

Table 1: Test problems.

#	Classic NM	Classic NM	DENM	DENM
	fevals	min	fevals	min
1	426	9.98402E-31	394	1.97215E-31
2	N.A.	4.89843E+01	N.A.	4.89843E+01
3	973	0	943	0
4	562	6.41934E-21	549	7.45796E-22
5	317	4.93038E-32	317	4.93038E-32
6	6000	1.24362E+02	386	1.24362E+02
7	507	1.53917E-31	507	1.53917E-31
8	528	8.21488E-03	552	8.21488E-03
9	6003	1.12793E-08	6003	1.12793E-08
10	3336	8.79459E+01	3637	8.79459E+01
11	873	3.44571E-07	831	3.44571E-07
12	804	1.23260E-32	647	5.42743E-37
13	713	1.52652E-24	744	3.69336E-25
14	1243	4.50390E-30	858	5.30016E-30
15	741	3.07506E-04	775	3.07506E-04
16	681	8.58222E+04	6004	8.58222E+04
17	1921	5.46489E-05	1974	5.46489E-05
18	2606	2.74301E-31	2451	8.40187E-32
19	4206	4.01377E-02	6000	4.01377E-02
20	1389	2.28767E-03	1635	2.28767E-03
21	6003	4.14152E-30	3674	1.72563E-30
22	5579	1.73539E-20	5657	5.34018E-20
23	6005	2.24998E-05	2098	2.24998E-05
24	4867	9.37629E-06	2477	9.37629E-06
25	6007	5.30016E-31	1499	1.23260E-31
26	N.A.	2.74129E-04	N.A.	2.74129E-04
27	794	0	794	0
28	1432	2.72339E-33	1433	2.72339E-33
29	1342	1.97408E-33	1342	1.97408E-33
30	1285	1.13399E-30	1286	1.13399E-30
31	909	3.73630E-31	910	3.73630E-31
32	614	4.00000E+00	6002	4.00000E+00
33	6001	1.64706E+00	583	1.64706E+00
34	592	3.15385E+00	593	3.15385E+00
35	6001	0	6001	0

Table 2: Results using CNM and DENM using $\alpha = 1$, $\gamma = 1.9$, $\beta = 0.6$, and $\delta = 0.6$. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DENM	DENM
	fevals	min	fevals	min
1	323	7.88861E-31	311	0
2	N.A.	4.89843E+01	N.A.	4.89843E+01
3	854	0	781	0
4	352	2.32008E-22	345	3.59387E-22
5	237	0	237	0
6	310	1.24362E+02	331	1.24362E+02
7	384	2.03786E-32	386	3.36183E-32
8	447	8.21488E-03	431	8.21488E-03
9	400	1.12793E-08	400	1.12793E-08
10	3175	8.79459E+01	3357	8.79459E+01
11	757	3.44571E-07	658	3.44571E-07
12	725	0	477	7.87379E-35
13	670	6.71089E-24	645	2.42195E-24
14	1048	2.76718E-30	702	0
15	696	3.07506E-04	767	3.07506E-04
16	652	8.58222E+04	698	8.58222E+04
17	1636	5.46489E-05	2194	5.46489E-05
18	2362	3.26334E-30	3459	2.36552E-31
19	3982	4.01377E-02	5535	4.01377E-02
20	1275	2.28767E-03	1315	2.28767E-03
21	6006	5.26220E-28	2587	2.91385E-29
22	5440	1.03232E-13	5755	4.06871E-10
23	1412	2.24998E-05	6000	2.24998E-05
24	4318	9.37629E-06	5915	9.37629E-06
25	1442	1.12166E-30	1717	3.56220E-30
26	1032	1.40015E-31	1143	2.90238E-31
27	590	0	590	0
28	1314	1.91780E-32	1323	1.91780E-32
29	1155	4.42964E-33	1158	4.42964E-33
30	1208	2.50587E-29	1213	2.50587E-29
31	6004	7.39557E-31	810	2.01537E-29
32	529	4.00000E+00	562	4.00000E+00
33	488	1.64706E+00	506	1.64706E+00
34	517	3.15385E+00	518	3.15385E+00
35	245	0	245	0

Table 3: Results using CNM and DENM using $\alpha = 1$, $\gamma = 1.9$, $\beta = 0.5$, and $\delta = 0.6$. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DENM	DENM
	fevals	min	fevals	min
1	333	0	311	1.97215E-31
2	N.A.	4.89843E+01	N.A.	4.89843E+01
3	828	0	792	0
4	369	8.78073E-22	364	8.94573E-24
5	243	0	243	0
6	298	1.24362E+02	322	1.24362E+02
7	382	6.27580E-32	383	6.27580E-32
8	398	8.21488E-03	407	8.21488E-03
9	378	1.12793E-08	378	1.12793E-08
10	2985	8.79459E+01	3130	8.79459E+01
11	624	3.44571E-07	623	3.44571E-07
12	449	2.14104E-38	441	7.10712E-36
13	597	1.00450E-23	646	4.11967E-25
14	1225	3.80995E-30	1364	1.39776E-30
15	639	3.07506E-04	653	3.07506E-04
16	518	8.58222E+04	665	8.58222E+04
17	1696	5.46489E-05	2205	5.46489E-05
18	2715	1.41797E-31	3303	6.01372E-32
19	3853	4.01377E-02	5666	4.01377E-02
20	1174	2.28767E-03	1226	2.28767E-03
21	2209	4.14152E-30	4204	3.51290E-30
22	5455	8.54983E-10	N.A.	2.32264E-03
23	1711	2.24998E-05	3401	2.24998E-05
24	3850	9.37629E-06	5472	9.37629E-06
25	1560	3.93198E-30	1550	4.63456E-30
26	1208	8.01187E-32	1163	8.43557E-32
27	583	0	583	0
28	1399	1.10278E-31	1562	3.02160E-32
29	1387	4.62705E-32	1402	4.62705E-32
30	1342	2.85962E-30	1347	2.85962E-30
31	771	1.44214E-30	776	1.44214E-30
32	487	4.00000E+00	476	4.00000E+00
33	440	1.64706E+00	439	1.64706E+00
34	430	3.15385E+00	431	3.15385E+00
35	238	3.08149E-33	238	3.08149E-33

Table 4: Results using CNM and DENM using $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DEDCNM	DEDCNM
	fevals	min	fevals	min
1	738	4.93038E-32	628	4.93038E-32
2	N.A.	4.89843E+01	N.A.	4.89843E+01
3	1454	0	1270	0
4	951	9.03167E-27	762	2.40286E-22
5	673	0	527	4.93038E-32
6	547	1.24362E+02	473	1.24362E+02
7	1232	6.68312E-33	904	7.60884E-31
8	886	8.21488E-03	693	8.21488E-03
9	858	1.12793E-08	694	1.12793E-08
10	4172	8.79459E+01	4197	8.79459E+01
11	N.A.	1.40000E-03	N.A.	1.40000E-03
12	1227	1.09039E-30	966	1.65170E-35
13	1356	8.32547E-24	1124	8.75408E-25
14	1529	1.77494E-31	1622	1.77494E-31
15	1224	3.07506E-04	1020	3.07506E-04
16	1020	8.58222E+04	949	8.58222E+04
17	4132	5.46489E-05	2619	5.46489E-05
18	N.A.	5.65565E-03	5803	1.34093E-31
19	4839	4.01377E-02	5751	4.01377E-02
20	1987	2.28767E-03	1870	2.28767E-03
21	3885	7.44487E-30	5779	5.05364E-31
22	6001	1.70055E-16	6000	1.08989E-15
23	2764	2.24998E-05	2869	2.24998E-05
24	N.A.	9.42020E-06	N.A.	9.45158E-06
25	2783	8.62817E-31	2306	2.85962E-30
26	2403	4.99779E-31	2382	1.26919E-31
27	1404	0	1090	0
28	2719	9.99377E-33	2202	1.75530E-32
29	2772	4.33334E-34	2140	2.93704E-33
30	2623	2.36658E-30	2127	2.85962E-30
31	1872	1.08468E-30	1449	2.98904E-31
32	939	4.00000E+00	800	4.00000E+00
33	955	1.64706E+00	744	1.64706E+00
34	864	3.15385E+00	747	3.15385E+00
35	673	0	528	0

Table 5: Results using CNM and DEDCNM using $\alpha = 1.1$, $\gamma = 2$, $\beta = 0.8$, and $\delta = 0.5$. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DEDCNM	DEDCNM
	fevals	min	fevals	min
1	504	0	458	0
2	N.A.	4.89843E+01	N.A.	4.89843E+01
3	1113	0	1068	0
4	652	1.31897E-23	474	2.54013E-23
5	445	0	325	0
6	367	1.24362E+02	367	1.24362E+02
7	737	2.36036E-31	534	2.16804E-31
8	608	8.21488E-03	510	8.21488E-03
9	594	1.12793E-08	509	1.12793E-08
10	6002	8.79459E+01	3538	8.79459E+01
11	1041	3.44571E-07	N.A.	8.43383E-04
12	774	1.15789E-35	612	4.45413E-36
13	865	1.88957E-23	929	1.02610E-24
14	1374	5.72417E-30	1701	7.76535E-31
15	829	3.07506E-04	747	3.07506E-04
16	779	8.58222E+04	731	8.58222E+04
17	1865	5.46489E-05	2260	5.46489E-05
18	4733	1.61056E-31	6001	2.58867E-26
19	3918	4.01377E-02	N.A.	4.63546E-02
20	1323	2.28767E-03	1507	2.28767E-03
21	2304	7.54348E-30	N.A.	3.69759E+00
22	5534	7.61374E-17	N.A.	2.90655E-06
23	4156	2.24998E-05	2190	2.24998E-05
24	3578	9.37629E-06	N.A.	9.57015E-06
25	1899	2.46519E-32	2031	3.18010E-30
26	1505	2.37515E-31	1702	5.96056E-30
27	1150	0	781	0
28	1877	3.01227E-33	1902	5.14283E-33
29	1809	3.27408E-33	1694	1.78149E-33
30	1730	1.08468E-30	1687	6.31089E-30
31	1251	5.42342E-31	1100	4.00593E-30
32	665	4.00000E+00	610	4.00000E+00
33	673	1.64706E+00	537	1.64706E+00
34	650	3.15385E+00	559	3.15385E+00
35	455	0	337	7.70372E-34

Table 6: Results using Classic NM and DEDCNM using $\alpha = 1$, $\gamma = 2$, $\beta = 0.7$, and $\delta = 0.5$. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DEDCNM	DEDCNM
	fevals	min	fevals	min
1	333	0	364	1.97215E-31
2	N.A.	4.89843E+01	N.A.	4.89843E+01
3	828	0	973	0
4	369	8.78073E-22	367	2.55023E-24
5	243	0	198	0
6	298	1.24362E+02	6001	1.24362E+02
7	382	6.27580E-32	944	5.69854E-31
8	398	8.21488E-03	695	8.21488E-03
9	378	1.12793E-08	563	1.12793E-08
10	2985	8.79459E+01	N.A.	2.45515E+02
11	624	3.44571E-07	684	3.44571E-07
12	449	2.14104E-38	383	1.55712E-35
13	597	1.00450E-23	N.A.	9.17965E-02
14	1225	3.80995E-30	N.A.	5.46428E+00
15	639	3.07506E-04	N.A.	3.09442E-04
16	518	8.58222E+04	N.A.	1.43333E+06
17	1696	5.46489E-05	N.A.	3.74676E-03
18	2715	1.41797E-31	N.A.	6.02672E-03
19	3853	4.01377E-02	N.A.	1.61310E-01
20	1174	2.28767E-03	N.A.	6.82074E-03
21	2209	4.14152E-30	N.A.	2.61018E+00
22	5455	8.54983E-10	N.A.	5.93507E+00
23	1711	2.24998E-05	N.A.	3.11092E-05
24	3850	9.37629E-06	N.A.	9.81221E-06
25	1560	3.93198E-30	N.A.	1.00263E-03
26	1208	8.01187E-32	N.A.	1.03420E-03
27	583	0	410	0
28	1399	1.10278E-31	N.A.	1.90491E-06
29	1387	4.62705E-32	N.A.	1.34520E-05
30	1342	2.85962E-30	N.A.	3.41845E-01
31	771	1.44214E-30	N.A.	2.79549E-05
32	487	4.00000E+00	N.A.	4.00009E+00
33	440	1.64706E+00	368	1.64706E+00
34	430	3.15385E+00	408	3.15385E+00
35	238	3.08149E-33	225	3.08149E-32

Table 7: Results using Classic NM and DEDCNM using $\alpha = 1$, $\gamma = 2$, $\beta = 0.5$, and $\delta = 0.5$. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DENM	DENM	DEDCNM	DEDCNM
	fevals	min	fevals	min	fevals	min
1	333	0	311	0	364	0
2	N.A.	4.89843E+01	N.A.	4.89843E+01	N.A.	4.89843E+01
3	828	0	792	0	973	0
4	369	0	364	0	367	0
5	243	0	243	0	198	0
6	298	1.24362E+02	322	1.24362E+02	6001	1.24362E+02
7	382	0	383	0	944	0
8	398	8.21488E-03	407	8.21488E-03	695	8.21488E-03
9	378	1.12793E-08	378	1.12793E-08	563	1.12793E-08
10	2985	8.79459E+01	3130	8.79459E+01	N.A.	2.45515E+02
11	624	3.44571E-07	623	3.44571E-07	684	3.44571E-07
12	449	0	441	0	383	0
13	597	0	646	0	N.A.	9.17965E-02
14	1225	0	1364	0	N.A.	5.46428E+00
15	639	3.07506E-04	653	3.07506E-04	N.A.	3.09442E-04
16	518	8.58222E+04	665	8.58222E+04	N.A.	1.43333E+06
17	1696	5.46489E-05	2205	5.46489E-05	N.A.	3.74676E-03
18	2715	0	3303	0	N.A.	6.02672E-03
19	3853	4.01377E-02	5666	4.01377E-02	N.A.	1.61310E-01
20	1174	2.28767E-03	1226	2.28767E-03	N.A.	6.82074E-03
21	2209	0	4204	0	N.A.	2.61018E+00
22	5455	0	N.A.	2.32264E-03	N.A.	5.93507E+00
23	1711	2.24998E-05	3401	2.24998E-05	N.A.	3.11092E-05
24	3850	9.37629E-06	5472	9.37629E-06	N.A.	9.81221E-06
25	1560	0	1550	0	N.A.	1.00263E-03
26	1208	0	1163	0	N.A.	1.03420E-03
27	583	0	583	0	410	0
28	1399	0	1562	0	N.A.	1.90491E-06
29	1387	0	1402	0	N.A.	1.34520E-05
30	1342	0	1347	0	N.A.	3.41845E-01
31	771	0	776	0	N.A.	2.79549E-05
32	487	4	476	4	N.A.	4
33	440	1.64706E+00	439	1.64706E+00	368	1.64706E+00
34	430	3.15385E+00	431	3.15385E+00	408	3.15385E+00
35	238	0	238	0	225	0

Table 8: Test results using Classic NM, DENM, and DEDCNM - using default parameters. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.

#	Classic NM	Classic NM	DENM	DENM	DEDCNM	DEDCNM
	fevals	min	fevals	min	fevals	min
1	330	0	345	0	582	0
2	N.A.	4.89843E+01	N.A.	4.89843E+01	N.A.	4.89843E+01
3	884	0	857	0	1249	0
4	425	0	409	0	686	0
5	240	0	244	0	472	0
6	321	1.24362E+02	285	1.24362E+02	353	1.24362E+02
7	417	0	438	0	841	0
8	370	8.21488E-03	400	8.21488E-03	570	8.21488E-03
9	368	1.12793E-08	396	1.12793E-08	551	1.12793E-08
10	2880	8.79459E+01	3153	8.79459E+01	4073	8.79459E+01
11	599	3.44571E-07	633	3.44571E-07	N.A.	1.40000E-03
12	499	0	459	0	837	0
13	551	0	648	0	1092	0
14	657	0	843	0	1958	0
15	619	3.07506E-04	753	3.07506E-04	805	3.07506E-04
16	552	8.58222E+04	587	8.58222E+04	818	8.58222E+04
17	1773	5.46489E-05	2233	5.46489E-05	2633	5.46489E-05
18	2025	0	3335	0	5504	0
19	4326	4.01377E-02	3554	4.01377E-02	6005	4.01377E-02
20	996	2.28767E-03	1287	2.28767E-03	1573	2.28767E-03
21	2297	0	2305	0	6001	0
22	5407	1.45842E-08	5729	0	6001	0
23	1037	2.24998E-05	2442	2.24998E-05	1406	2.24998E-05
24	3608	9.37629E-06	4350	9.37629E-06	4091	9.37629E-06
25	1372	0	1430	0	2239	0
26	971	0	610	0	1698	0
27	562	0	564	0	1021	0
28	1228	0	1304	0	2072	0
29	1373	0	1285	0	2118	0
30	1149	0	1224	0	1926	0
31	716	0	806	0	1344	0
32	473	4	468	4	643	4
33	430	1.64706E+00	432	1.64706E+00	593	1.64706E+00
34	416	3.15385E+00	401	3.15385E+00	559	3.15385E+00
35	258	0	244	0	456	0

Table 9: Test results using Classic NM, DENM, and DEDCNM - using optimal parameters. Number of function evaluations used (fevals) and minimum objective function value obtained (fmin). N.A. represents that the function was not solved to a relative error less than 10^{-6}.