A NUMERICAL COMPARISON BETWEEN THE LANCELOT AND MINOS PACKAGES FOR LARGE-SCALE NONLINEAR OPTIMIZATION

by I. Bongartz[0], A.R. Conn[1], N. I. M. Gould[2],
M. Saunders[3] and Ph.L. Toint[4]

Report 97/13 October 9, 1997

[0] ProMIRA Software Inc, 2650 Queensview Drive, Suite 100,
Ottawa, ON K2B 8H6 CANADA
Email: ingridb@promira.com

[1] IBM T.J. Watson Research Center, P.O.Box 218, Yorktown Heights, NY 10598, USA
Email: arconn@watson.ibm.com

[2] Rutherford Appleton Laboratory,
Chilton, Oxfordshire, England
Email: N.Gould@rl.ac.uk
Current reports available by anonymous ftp from the directory
"pub/reports" on joyous-gard.cc.rl.ac.uk (internet 130.246.9.91)

[3] Department of Operations Research, Stanford University
Stanford, CA 94305, USA
Email: mike@sol-michael.stanford.edu

[4] Department of Mathematics, Facultés Universitaires ND de la Paix, 61, rue de Bruxelles,
B-5000 Namur, Belgium, EU
Email: pht@math.fundp.ac.be
Current reports available by anonymous ftp from the directory
"pub/reports" on thales.math.fundp.ac.be (internet 138.48.4.14)

A numerical comparison between the LANCELOT and MINOS packages for large-scale constrained optimization

I. Bongartz* A. R. Conn† N. I. M. Gould‡ M. A. Saunders§
Ph. L. Toint¶

October 9, 1997

Abstract

We present the results of a numerical comparison of two nonlinear optimization packages capable of handling large problems, MINOS 5.5 and LANCELOT (Release A). The comparison was performed using over 900 constrained and unconstrained problems from the CUTE collection.

With the default options, LANCELOT makes use of first and second derivatives, while MINOS requires gradients but cannot use higher derivatives.

We conclude that LANCELOT is usually more efficient in terms of the number of function and derivative evaluations. If the latter are inexpensive, MINOS may require less CPU time unless there are many degrees of freedom. LANCELOT proves to be less reliable than MINOS on linear programming problems, but somewhat more reliable on problems involving nonlinear constraints.

Keywords: nonlinear optimization, constrained optimization, software, numerical testing

1 Introduction

MINOS (Murtagh and Saunders, 1993) and LANCELOT (Conn, Gould and Toint, 1992*a*) are two popular packages for solving large and small, constrained and unconstrained, nonlinear optimization problems of the form

$$\begin{array}{ll} \underset{x\in\mathbf{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & l \leq \begin{pmatrix} x \\ Ax \\ c(x) \end{pmatrix} \leq u, \end{array} \tag{1.1}$$

*ProMIRA Software Inc, 2650 Queensview Drive, Suite 100 Ottawa, ON K2B 8H6 CANADA. (ingridb@promira.com).

†IBM T. J. Watson Research Center, Yorktown Heights, NY 10598, USA (arconn@watson.ibm.com). The research of this author was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Air Force Office of Scientific Research under Contract No. F49620-91-C-0079. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

‡Department for Computation and Information, Rutherford Appleton Laboratory, Chilton, Oxfordshire OX11 0QX, England (n.gould@rl.ac.uk). Technical reports from ftp://joyous-gard.cc.rl.ac.uk/pub/reports.

§Department of EESOR, Stanford University, Stanford, CA 94305-4023, USA (mike@SOL-michael.stanford.edu). Partially supported by National Science Foundation grant DMI-9500668 and Office of Naval Research grant N00014-96-1-0274.

¶Dept of Mathematics, Facultés Universitaires Notre-Dame de la Paix, 61, rue de Bruxelles, B-5000 Namur, Belgium (pht@math.fundp.ac.be). Technical reports from ftp://thales.math.fundp.ac.be/pub/reports.

where A is a $p \times n$ matrix and where $c(x) \in \mathbf{R}^m$. The objective function $f(x)$ and the constraint functions $c_i(x)$ are assumed to be smooth.

Algorithmically, the packages are quite different. MINOS uses a reduced-space approach to handle linear constraints and linearizations of any nonlinear constraints. LANCELOT uses an augmented Lagrangian technique in which only the bound constraints are enforced directly. The numerical performance of the various algorithmic options within LANCELOT has been analyzed by Conn, Gould and Toint (1996), but little has been published concerning the numerical performance of MINOS, and even less is known about how these packages compare in reliability and efficiency. Our aim is to provide such a comparison. (Recently Gill, Murray and Saunders, 1997, have compared MINOS and SNOPT on many test problems, including the CUTE collection used here.)

A *fair* comparison between two pieces of software such as LANCELOT and MINOS seems extremely difficult for several reasons:

- A specific set of test problems must be chosen to which both packages can be applied without biasing the conclusions. This requires careful thought and a sufficient number of test problems; otherwise, the statistics presented can be of only limited value.

- Various measures of algorithmic efficiency are possible—for example, function evaluation counts or CPU time. Thus, even on a common set of problems, the conclusions can be sensitive to a particular choice.

- Certain algorithmic options must be selected for each package because comparing all possible combinations is not practical. In addition, some options may be particularly suitable for certain special situations; for example, each algorithm has an associated folklore about options that are best under certain circumstances. Moreover, options are not always independent.

- Decisions about stopping criteria and memory allocation are important in practice but it may be very difficult to determine the best choice in any given circumstance.

- Finally, there always remains the difficulty of presenting the comparison data itself, where a compromise has to be sought between exhaustive and concise presentations. The main tendencies need to be captured, yet the detail must be such that the conclusions are clearly supported by the *presented* evidence.

The paper is organized as follows. Section 2 briefly recalls the algorithmic features of both packages. Section 3 describes the comparison methodology and indicates how the above-mentioned difficulties have been tackled. Section 4 presents the principal numerical comparison using default options for both packages. Finally, some conclusions are drawn in Section 5.

2 A brief outline of LANCELOT and MINOS

2.1 LANCELOT

The kernel algorithm for LANCELOT is an adaptation of trust-region methods to optimization problems with simple bounds. The method is extended to general constraints by using an augmented Lagrangian function, and the bounds are handled directly via projections that are easy to compute. We use group partial separability (a generalization of sparsity, introduced by Griewank and Toint, 1982) to allow efficient storage and updating of matrices in matrix-vector product form. This approach has the further advantage that accurate approximations to the second derivatives of the element functions, which are normally of low rank, are easier to obtain than for the assembled matrices. An introduction to group partial separability, a remarkably general structure, is given by Conn, Gould and Toint (1990). There is also a book (Conn et al., 1992*a*) to accompany the LANCELOT software.

The basic idea of the trust-region scheme is to model the objective function about the current point x^k. One 'trusts' this model in a neighbourhood (called the trust region) of x^k. The model is approximately optimized within the trust region, producing a new point $x^{k+1} = x^k + s^k$. One then determines how well the model actually predicted the change in the true objective function. If good agreement is obtained, the trust region is expanded, while if the agreement is moderate, the trust region remains unchanged. If the agreement is poor, x^{k+1} is reset to x^k and the trust region is contracted. The beauty of such an approach is that, when the trust region is small and the problem is smooth, the approximation will be reliable so long as the model and true objective function gradients are close. Moreover, provided that one finds an approximate minimizer of the model within the trust region whose value is at least as small as the minimum along the steepest-descent direction (this determines the so-called Cauchy point), one can ensure convergence to a stationary point. In addition, the trust region will eventually be sufficiently large that it does not interfere with the subsequent iterates. Thus, providing that the underlying model minimization is sufficientl y sophisticated in this situation, one can ensure fast asymptotic convergence. For further details, see Moré (1983).

To extend these ideas to the case of simple bounds, we generalize the Cauchy point to the minimum along the *projected* gradient path within the trust region, where the projection is with respect to the simple bounds. As in the unconstrained case, global convergence can be guaranteed, provided one finds a model value which is at least as small as at the generalized Cauchy point. A satisfactory asymptotic convergence rate can be obtained by suitable further reductions of the model function. In the context of LANCELOT, this is achieved by fixing the variables which are active at the generalized Cauchy point, and further reducing the model within the feasible region and trust region using just the remaining free variables. Updating of the trust-region size is performed exactly as in the unconstrained case.

Provided the quadratic model is reasonable, we are able to prove convergence to a Kuhn-Tucker point. Moreover, we identify the correct active bound constraints after a finite number of iterations, provided that strict complementarity is satisfied and the free variables determined by the generalized Cauchy point remain free when the model is further reduced. Details are given in Conn, Gould and Toint (1988).

The extension to general constraints is carried out by means of an augmented Lagrangian function. Linear constraints are treated no differently from nonlinear constraints, and we may thus assume that $p = 0$ in (1.1). LANCELOT first introduces slack or surplus variables, if necessary, to change general inequality constraints to equations. Subsequently a sequence of augmented Lagrangian functions of the form

$$\Phi(x, \lambda, S, \mu) = f(x) + \lambda^T c(x) + \frac{1}{2\mu} c(x)^T c(x) \tag{2.1}$$

are (approximately) minimized subject to explicit bounds, using the earlier algorithm. The Lagrange multiplier estimates λ and penalty parameter μ may be suitably adjusted so that convergence to a first-order stationary point for the original nonlinear program is ensured under suitable conditions. Furthermore, if there is a single limit point, the penalty parameter μ is bounded away from zero. Under somewhat stronger conditions, we can show that only a single iteration of the simple bounds algorithm is required to satisfy the convergence conditions for the inner problem (the optimization for the given λ and μ). Details of these important properties are given by Conn, Gould and Toint (1991, 1992*b*).

A significant, often dominant, cost in optimization is solving a linear system to obtain a search direction. Typically the system is symmetric and arises from the need to determine an approximate stationary point for a quadratic function. If the system is large there are two common approaches. The first is to apply a direct method based upon multifrontal techniques, involving partial assembly and dense-matrix technology on sparse matrices. General details are given in Duff, Erisman and Reid (1986), and an application in the context of LANCELOT is given in Conn, Gould, Lescrenier and Toint (1994). Our experience to date, however, has been that an iterative approach is more robust. The most popular such method is preconditioned conjugate gradients, and LANCELOT provides a variety of suitable preconditioners.

2.2 MINOS

For practitioners familiar with linear programming methods, the MINOS package of Murtagh and Saunders (1978, 1982) serves as a very useful bridge to nonlinear programming. So let us first assume that all the constraints are linear and there is no $c(x)$.

Linearly constrained (LC) problems are dealt with most easily, using sparse basis-handling techniques as in the simplex method. Since nonlinearities generally move optimal points away from a vertex of the feasible region (where fewer than n constraints are active), the set of active constraints is represented in the form

$$\hat{A}x = (\hat{A}P)(P^T x) = \begin{pmatrix} & & I \\ B & S & N \end{pmatrix} \begin{pmatrix} x_B \\ x_S \\ x_N \end{pmatrix} = \hat{b},$$

where $\hat{A}x$ is a subset of the rows of (x, Ax) in (1.1), P is a permutation, $\hat{b}$ are the active bounds from l and u, and B is square and nonsingular. The variables associated with B, S and N are called basic, superbasic and nonbasic respectively.

We define n_S, the *number of degrees of freedom* at the current point, to be the number of superbasic variables (the column dimension of S). It is the number of variables minus the number of active constraints, and is often quite small for constrained nonlinear problems regardless of the number of nonlinearities. Optimization algorithms differ in their ability to deal with many degrees of freedom, and this is one of the trends we shall examine in our numerical experiments. In MINOS, a triangular matrix R of dimension n_S is needed to perform unconstrained optimization in an appropriate subspace. Quasi-Newton updates are used to approximate the *reduced Hessian* in the form

$$R^T R \approx Z^T H Z, \qquad Z = P \begin{pmatrix} -B^{-1}S \\ I \\ 0 \end{pmatrix}, \tag{2.2}$$

where $H = \nabla^2 f(x)$ and Z spans the null space of $\hat{A}$. Normally, R is stored as a *dense* upper-triangular matrix, but when there are many degrees of freedom, MINOS works with a "partial Hessian" of the form

$$R = \begin{pmatrix} R_r & 0 \\ & D \end{pmatrix}, \tag{2.3}$$

where R_r is a dense triangle and D is diagonal. In our numerical tests, we set `Hessian dimension` to `500` to define the maximum size of R_r. MINOS treats Z as an operator without forming its columns explicitly, using sparse LU factors of B to compute the products Zd and $Z^T g$. The main workspace required by MINOS is for L, U and the dense triangle R_r.

For nonlinearly constrained (NC) problems, MINOS applies the above algorithm to a *sequence of LC subproblems*, in which the constraints are linearized at the current point x_k and the objective is a modified augmented Lagrangian. Thus, each major iteration involves an LC subproblem of the form

$$\begin{array}{ll} \underset{x \in \mathbf{R}^n}{\text{minimize}} & f(x) - \lambda_k^T d_L(x, x_k) + \frac{1}{2}\rho_k \|d_L(x, x_k)\|^2 \\ \text{subject to} & l \leq \begin{pmatrix} x \\ Ax \\ c_L(x, x_k) \end{pmatrix} \leq u, \end{array} \tag{2.4}$$

where $c_L(x, x_k) = c(x_k) + \nabla c(x_k)(x - x_k)$ is the constraint linearization and $d_L(x, x_k) = c(x) - c_L(x, x_k)$ is the departure from linearity. The vector λ_k is a set of Lagrange multiplier estimates for the nonlinear constraints, and ρ_k is a penalty parameter.

Ideally, the final primal and dual variables from the subproblem become the next approximation to the solution, (x_{k+1}, λ_{k+1}). A major theoretical difficulty lies in choosing a steplength from (x_k, λ_k) towards that point. MINOS usually takes a full step, but heuristically may shorten the step to prevent large changes to $\|x_k\|$ and $\|\lambda_k\|$. This strategy has not, as yet, been shown to ensure convergence to a suitable first-order constrained stationary point. This may be the main reason why LANCELOT proves to be more reliable on problems with nonlinear constraints in our numerical experiments.

3 The comparison methodology

Having recalled the main algorithmic features of LANCELOT and MINOS, we now describe the procedure that has been used to compare these packages. As mentioned in the introduction, there are a number of intrinsic difficulties that arise in such a comparison. The purpose of this section is to discuss the options taken, many of which were influenced by the choices made in a similar context by Conn et al. (1996).

Unfortunately, there is little consensus in the community as to the most appropriate criteria for such comparisons, no doubt because such judgements are both difficult and inherently subjective. Inevitably our comparison procedure could be improved. However, we believe that we have taken adequate precautions to ensure that the resulting methodology makes the numerical comparison meaningful and of interest.

3.1 The test problems

The first decision taken was to test and report on a large number and variety of test problems. In our experience, this is essential for a valid assessment of software reliability and performance. Smaller test sets are more likely to introduce unwanted bias and make the statistical results less useful.

We have chosen 112 linear programs, 225 quadratic programs, 300 unconstrained or bound-constrained, 58 linearly constrained and 218 nonlinearly constrained problems. Most of the linear programs are from the NETLIB collection (Gay, 1985). The remaining problems are from the CUTE collection (Bongartz, Conn, Gould and Toint, 1995), the largest single set of optimization problems currently available. They include almost all of those tested by Conn et al. (1996), augmented by new problems that were not available at the time. Only two instances of each of the variable-dimension problem were used, one large and one small. We felt that the inclusion of further instances would provide little additional information and perhaps bias the overall conclusions.

Care was taken to ensure that both packages could be started on each problem. Thus it was necessary to eliminate some large problems whose memory requirements exceeded the storage provided. Of course, the sparse factorizations in MINOS and some of the algorithmic variants of LANCELOT might subsequently request additional memory that is not available. This infrequent occurrence is treated as early termination, and was not used to eliminate test problems.

It was also decided that smaller problems should *not* be excluded from the comparison, despite the fact that both packages are known particularly for their ability to handle larger problems. Indeed, experience shows that both LANCELOT and MINOS are useful for solving problems of moderate (and even small) size. Ultimately, 913 test problems were included, derived from 704 *different* problems, the additional examples being determined by varying the dimension.

Of course we cannot include here all details of the test problems. Suffice it to say, our test set includes the following collections:

- The NETLIB linear programming examples.
- The "Argonne test set" of Moré, Garbow and Hillstrom (1981), the Testpack suite of Buckley (1989), the collection of Hock and Schittkowski (1981), the quadratic problems of Moré and Toraldo (1991), and the network problems of Dembo (1984) and Toint and Tuyttens (1990).
- Most problems from the PSPMIN collection of Toint (1983). Some trivial problems were skipped, as were problems for which different local minima were known.
- Problems inspired by the orthogonal regression report by Gulliksson (1990).
- Those problems from the MINPACK-2 collection of Averick, Carter and Moré (1991) and Averick and Moré (1991) that we could reconstruct from the data given in the reports.
- Some of the second Schittkowski (1987) collection.
- A number of original problems from various application areas.

Following Conn et al. (1996), we present some of the problem characteristics in Figures 1 and 2 and Table 1.

- Figure 1 shows the distribution of the problem dimensions.

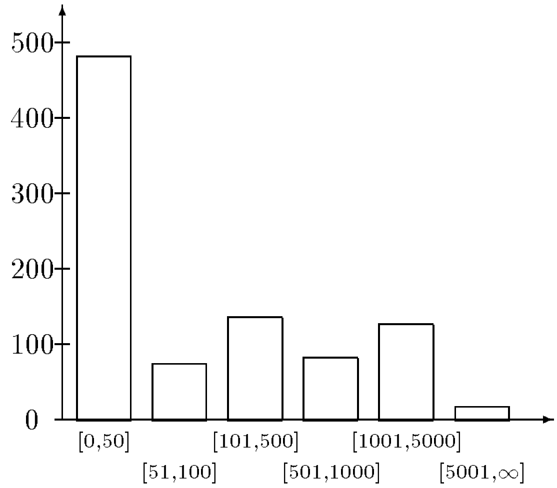

Figure 1: Distribution of problem dimensions n

- Figure 2 illustrates the distribution of the ratio m/n, where m is the total number of general equality and inequality constraints. The higher this ratio, the more constrained the problem. Only constrained problems ($m > 0$) are considered in this statistic.

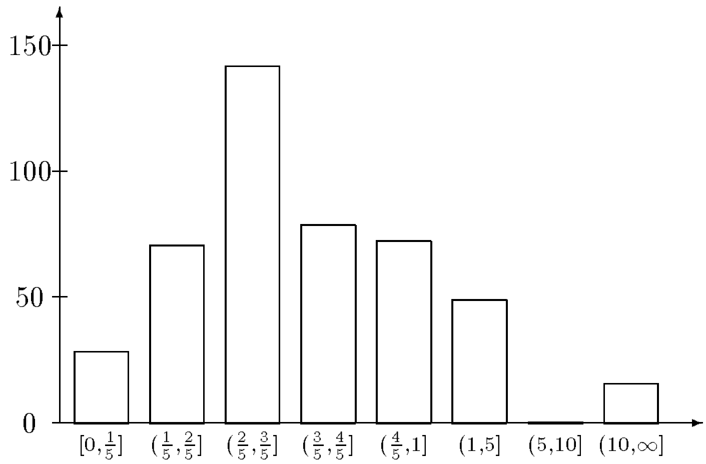

Figure 2: Distribution of the relative number of constraints m/n

- Figure 3 shows the distribution of the "number of degrees of freedom" in the problems; that is, the difference between the number of variables n and the number of constraints (including bounds) that are active at the solution[1]. We obtained these as the number of superbasic variables reported by MINOS (see Section 2.2). The set of problems considered here therefore excludes 37 problems for which MINOS failed before giving this information; see Table 2 below.

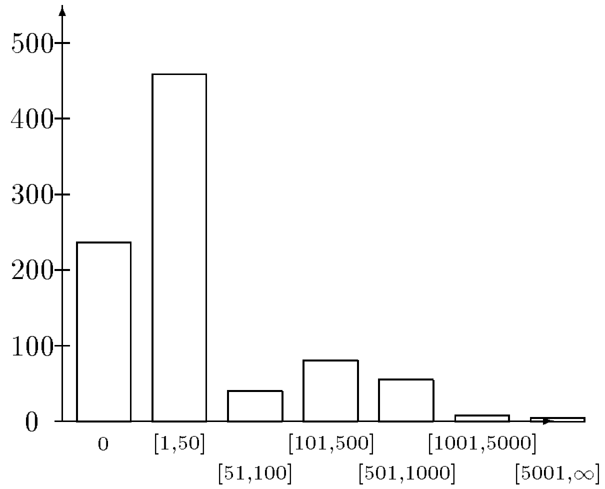

Figure 3: Distribution of the number of degrees of freedom per problem

- Table 1 reports the number of problems for which a given characteristic lies in one of five possible intervals $[0, 0.2]$, $(0.2, 0.4]$, $(0.4, 0.6]$, $(0.6, 0.8]$ and $(0.8, 0.1]$. Four characteristics are examined:

  1. The *relative nonlinearity of the objective function*:

  $$\nu_{\rm obj} \stackrel{\rm def}{=} \frac{\text{number of nonlinear groups in the objective}}{\text{number of groups in the objective}}. \tag{3.1}$$

  See Conn et al. (1990) or Conn et al. (1992*a*) for the precise definition of groups.

  2. The *relative nonlinearity of the constraints*:

  $$\nu_{\rm cons} \stackrel{\rm def}{=} \frac{\text{number of nonlinear constraints}}{m}, \tag{3.2}$$

  where the denominator is the total number of constraints excluding bounds.

  3. The proportion of variables subject to *bound constraints*: n_b/n

  4. The proportion of *equality constraints*:

  $$\gamma \stackrel{\rm def}{=} \frac{\text{number of equality constraints}}{m}. \tag{3.3}$$

[1]Note that, for all unconstrained and bound constrained problems, this number was reset to the actual number of variables in the problem, because the number given by MINOS sometimes underestimates the true number of degrees of freedom in the case where some variables have the same value at the starting point and the solution.

	$[0,\frac{1}{5}]$	$(\frac{1}{5},\frac{2}{5}]$	$(\frac{2}{5},\frac{3}{5}]$	$(\frac{3}{5},\frac{4}{5}]$	$(\frac{4}{5},1]$
$\nu_{\rm obj}$	163	1	13	12	725
$\nu_{\rm cons}$	307	8	29	7	153
n_b/n	448	22	37	39	368
γ	155	8	28	11	302

Table 1: Further problem characteristics

Note that most problems are not very large. However, we recall that testing LANCELOT and MINOS on small problems is useful because the packages are widely applied to such problems. Furthermore, the classes of larger problems are far from empty. We also observe that most large problems tend to have a somewhat regular structure. This is noticeable in the distribution of the relative nonlinearity of the constraints, where either most or very few, if any, are nonlinear. The same phenomenon is observed for the proportion of bounded variables, which tends to be either very low or close to one. Finally, a relatively important class of problems have the same number of variables and general constraints—they are systems of nonlinear equations. A fair proportion of the other constrained problems have approximately half as many constraints as variables, and very few involve significantly more general constraints than variables.

3.2 The performances measures

We have chosen to compare LANCELOT amd MINOS on reliability and numerical efficiency only. Other issues such as ease of use are not unimportant, but they are less amenable to a quantitative analysis.

Numerical efficiency itself has several possible aspects. We have chosen to focus on the number of function and derivative evaluations required by the packages for solving a given problem, as well as on the CPU time needed. Comparing iteration numbers is not meaningful because the iterations in MINOS and LANCELOT are defined quite differently, and involve very disparate amounts of computation.

3.3 The algorithmic options

Comparing all possible algorithmic combinations of LANCELOT with all possible algorithmic combinations of MINOS is, of course, not practical. Instead, we have chosen to compare the "default" version of the package, for each problem and each package.

The default version of LANCELOT is specified by running it with the algorithmic specification file (SPEC.SPC) given in Figure 4. These options specify the use of first and second derivatives (which are known for all problems in the CUTE collection).[2] They also specify the use of preconditioned conjugate gradients for solving the linear system defining the step between two successive iterates, where the preconditioner is a banded restriction of the relevant system matrix, with semi-bandwidth five.

[2]It should be noted that one of the advantages of the group partial separable structure that underlies all CUTE test problems is that this structure facilitates the specification of derivatives.

The default version of MINOS 5.5 is not explicitly defined by Murtagh and Saunders (1993), but a number of tolerances and thresholds are provided by default. We have chosen to use those and alter the algorithmic specifications only to ensure that the memory allocation and the maximum number of iterations are roughly comparable to the “large” version of LANCELOT used in these tests. (See Chapter 6 of Conn et al. (1992*a*) for a detailed description of the sizing options in LANCELOT.) Thus, the “default version of MINOS” means MINOS 5.5 (Dec 1995) using the SPECS file given in Figure 5. The perhaps more widely-used MINOS 5.4 is functionally very similar.

```
BEGIN
   check-derivatives
   ignore-derivative-bugs
   exact-second-derivatives-used
   bandsolver-preconditioned-cg-solver-used 5
   exact-cauchy-point-required
   trust-region-radius             1.0D+0
   maximum-number-of-iterations   10000000
   print-level                          -1
   start-printing-at-iteration           0
   stop-printing-at-iteration     10000000
END
```

Figure 4: The LANCELOT default specification file

```
Begin MINOS Problem
   Minimize
   Jacobian                 Sparse
   Derivative Level              3
   Hessian Dimension           500
   Superbasics Limit          5000
   Iterations             99999999
   Major Iterations       99999999
   Minor Iterations             40
   Print Level                   0
End MINOS Problem
```

Figure 5: The MINOS default specification file

As is clear from these specification files, we have chosen not to limit the number of iterations (the maxima indicated are infinite for all practical purpose), because what is meant by an iteration is very different for each package. Otherwise a comparison would be very difficult to interpret.

4 Comparison of the default versions

Given the number of test problems, it is clearly undesirable to report here the detailed results for both packages on each instance. We therefore discuss only statistics and summaries below. The complete results are given in a separate report (Bongartz, Conn, Gould, Saunders and Toint, 1997), which is available by anonymous ftp from the authors' technical report sites.

All numerical tests were performed on a 333MHz DEC Alphastation 500 under Digital Unix 4.0A using the default options of the f77 Fortran compiler.

4.1 Reliability

We first compare the reliability of the default versions of both packages, and report in Table 2 the results of our runs. In this table, the column heading "BC" indicates the results for the 300 unconstrained or bound-constrained problems, and "LC" and "NC" refer to the 58 linearly constrained and 218 nonlinearly constrained cases of our test set, respectively. These results immediately prompt some comments on the stopping criteria used by both packages. Note that LANCELOT reported failure for 44 problems[3], although the solution was actually found to several significant digits. These reported failures are probably due to too stringent stopping criteria. This is especially noticeable for linear programs (22 cases). The same phenomenon happened 17 times[4] for MINOS. All of these reported LANCELOT and MINOS "failures" were counted as having been successfully solved. On the other hand, MINOS reported solving 22 of the problems[5] when the final point was significantly different from the known (and unique) solution of the problem. These cases were counted as failures. Note that this situation did not occur with LANCELOT. These erroneous reports seem to suggest that LANCELOT's stopping criteria are slightly more stringent than those of MINOS, and therefore cause it to report failure for solved problems more than MINOS, whose somewhat looser criteria resulted in some premature terminations.

The corresponding relative reliabilities are shown in Table 3. One immediately notices the disappointing reliability of LANCELOT on linear programs, especially given the perfect reliability of MINOS on such problems. By contrast both packages seem reasonably reliable on other problem types, with a noticeable advantage for LANCELOT on general nonlinear unconstrained and constrained problems.

We also report the various causes of failures for each class of problems and for both packages in Table 4. Here, "No progress" indicates that the package was stopped because no further significant progress could be made, "Maximum CPU" that the maximum CPU

[3]AGG, AGG2, AGG3, BOEING2, CVXQP1 (1000), CVXQP3 (1000), CZPROB, FINNIS, FIT2P, GANGES, GFRE-PNC, HS69, HS75, HS84, HS100MOD, HS101, HS102, HS103, HS109, HS166, HUESTIS (1000), METHANB8, MODEL NCVXQP7 (1000), POROUS1 (5184), POROUS2 (5184), POWELL20 (1000), PRODPL1, PRODPL2, QPCBOEI2, QPCSTAIR, SCAGR7, SCAGR25, SCORPION, SHELL, SHIP04S, SHIP08L, SHIP08S, SHIP12L, SIERRA, SIPOW4, STANDATA, STANDMPS and WOOD1P.

[4]ARGLINB (100), ARGLINC (100), CATENARY (15), EXTROSNB (1000), HS65, HS95, HS96, HS97, HS114, HS117, LEAKNET, LISWET5 (2002), MEYER3, PRODPL0, PRODPL1, SSEBNLN and SPANHYD.

[5]BEALE, DENSCHNC, HS41, LIARWHD (500), LISWET1 (103), LISWET1 (2002), LISWET7 (2002), LISWET8 (2002), LISWET9 (103), LISWET9 (2002), LISWET10 (2002), LISWET11 (103), LISWET11 (2002), LISWET12 (103), LISWET12 (2002), NONDIA (10), NONDIA (10000), OSBORNEB, SPMSRTLS (499), TOINTGSS (10), WEEDS and YAO(2002).

	LP	QP	BC	LC	NC
Both succeed	73	197	261	55	159
Only LANCELOT succeeds	0	13	29	1	31
Only MINOS succeeds	39	6	4	1	13
Both fail	0	9	6	1	15

Table 2: Reliability on 112 linear, 225 quadratic, 300 unconstrained, 58 linearly constrained and 218 nonlinearly constrained problems using the default versions of LANCELOT and MINOS

	LP	QP	BC	LC	NC
LANCELOT	65%	93%	97%	97%	87%
MINOS	100%	90%	88%	97%	79%

Table 3: Relative reliability on 112 linear, 225 quadratic, 300 unconstrained, 58 linearly constrained and 218 nonlinearly constrained problems using the default versions of LANCELOT and MINOS.

time allowed (namely 12000 seconds) was exceeded, and "Arith. error" that an arithmetic error (overflow or division by zero) occurred, causing the package to fail. Finally, "False conv." denotes cases where convergence was erroneously reported.

Failure type	package	LP	QP	BC	LC	NC
No progress	LANCELOT	3	12	7	2	19
	MINOS	0	8	3	0	7
Maximum CPU	LANCELOT	36	3	0	0	6
	MINOS	0	1	7	0	1
Arith. error	LANCELOT	0	0	3	0	3
	MINOS	0	1	16	1	38
False conv.	LANCELOT	0	0	0	0	0
	MINOS	0	12	9	1	0

Table 4: Failure causes for the default versions of LANCELOT and MINOS.

We note that MINOS often fails because of arithmetic errors (typically overflows in function evaluations at infeasible points), while LANCELOT's failures are mostly caused by a lack of significant progress or excessive time requirements. The latter cause is probably the most acceptable because it suggests at least that more time might produce a solution.

4.2 Efficiency: number of function evaluations

If we now wish to compare the efficiency of the packages on problems that both could solve, we must eliminate cases that converged to different local minima. We therefore discarded problems for which the optimal objective value differed by more than 1%, keeping for further consideration the 259 unconstrained, 52 linearly constrained and 153 nonlinearly constrained problems that were *coherently solved* in that sense. We emphasize that these are the only problems considered in the rest of this numerical comparison. We also discarded the linear and quadratic problems from the comparison of the number of function evaluations, since these evaluations can be regarded as internal linear algebraic work.

We first examined the default versions of the packages again and compared the number of function evaluations required for the solution of the coherently solved problems. More precisely, for LANCELOT we counted the number of evaluations of the objective and constraints together (and their first and second derivatives) as a single evaluation. For MINOS we took the maximum between the number of objective and nonlinear constraint evaluations (and their gradients). (If both objective and constraints are nonlinear, they are evaluated essentially the same number of times.) The cumulative results are shown in Table 5, where the difference between LANCELOT and MINOS is very apparent.

	LANCELOT	MINOS
253 BC problems	72582	846865
50 LC problems	5277	44393
153 NC problems	284660	97170
(152 NC problems excluding LAUNCH)	(75246)	(95909)

Table 5: Total number of function evaluations for the default versions of LANCELOT and MINOS

These numbers are misleading because more difficult problems tend to completely dominate the comparison,[6] which gives little indicatio n of the relative efficiencies on a problem-by-problem basis. Thus we also present a more disaggregate comparison.

	LANCELOT wins	tie	MINOS wins	MINOS- LANCELOT min	q1	med	q3	max
253 BC problems	219	2	32	-1532	7	52	506	146265
50 LC problems	40	1	10	-31	4	44	439	7916
153 NC problems	118	2	33	-208153	3	41	1129	22991

Table 6: Detailed comparison of the default versions of LANCELOT and MINOS for function evaluations

In columns two to four of Table 6, we report the number of times each package wins (that is, requires fewer function values) and the number of ties. Columns five to nine give information on the distribution of the *difference* between the number of MINOS and LANCELOT evaluations *when they are different.* The minimum (min), first quartile (q1),

[6]For instance, LANCELOT finally solved problem LAUNCH after 209414 function evaluations while MINOS only required 1261. This (admittedly striking) difference in performance between the two packages on a single problem completely swamps the aggregate results for nonlinearly constrained problems. See the bracketed numbers in Table 5 when LAUNCH is excluded.

median (med), third quartile (q3) and maximum of this distribution are reported. For instance, a positive value of the median indicates that MINOS requires more function evaluations for more than half the test problems. Conversely, the more negative these indicators are, the more favourable for MINOS.

According to standard statistical practice, we base our comments mostly on the intervals between the first and third quartile as estimates of the distribution ranges. We see immediately the advantage of LANCELOT, based on this criterion. It may be concluded that if function or constraints evaluations are costly, the default version of LANCELOT is likely to be preferable to the default version of MINOS.

An important question to examine is how these results depend on the number of degrees of freedom of the problems tested (the number of variables minus the number of constraints active at the solution, see Section 2.2). To provide an answer we obtained n_S, the number of superbasic variables reported by MINOS. We then divided the three classes of bound/unconstrained, linearly and nonlinearly constrained problems each into two subsets: the problems having less than 50 degrees of freedom and those having 50 or more. Tables 7 and 8 give the same information as Tables 5 and 6 for each of the two subsets.

	LANCELOT	MINOS
$<$ 50 superbasics		
157 BC problems	66159	115871
39 LC problems	1516	14097
138 NC problems	67620	39654
$\geq$ 50 superbasics		
92 BC problems	6325	728083
11 LC problems	323	29064
7 NC problems	457	55772

Table 7: Total number of function evaluations for the default versions of LANCELOT and MINOS according to the number of superbasic variables

	LANCELOT wins	tie	MINOS wins	MINOS – LANCELOT min	q1	med	q3	max
$<$ 50 superbasics								
157 BC problems	132	1	24	-1532	3	20	63	43473
39 LC problems	28	1	10	-31	-1	21	88	3731
138 NC problems	106	2	30	-23627	4	41	110	3016
$\geq$ 50 superbasics								
92 BC problems	83	1	8	-145	214	831	3460	146265
11 LC problems	11	0	0	332	345	2056	2989	7916
7 NC problems	7	0	0	597	597	6350	8874	22991

Table 8: Detailed comparison of the default versions of LANCELOT and MINOS for function evaluations, according to the number of superbasic variables

The conclusion, unsurprisingly, is that the performance of LANCELOT relative to MINOS improves when the number of degrees of freedom (superbasic variables) increases.

4.3 Efficiency: CPU time

We now turn to the comparison of CPU times, again restricted to the set of problems that were coherently solved by both packages. The cumulative times are reported in Table 9, while the disaggregate results are presented in Table 10. All reported times are in seconds, and two times are reported equal when they differ by less than 5% or less than half a second.

	LANCELOT	MINOS
73 LP problems	45812	466
197 QP problems	2222	4703
253 BC problems	2758	14388
50 LC problems	1526	219
153 NC problems	15754	1093

Table 9: Total CPU time for the default versions of LANCELOT and MINOS

	LANCELOT wins	tie	MINOS wins	MINOS- LANCELOT min	q1	med	q3	max
73 LP problems	0	11	62	-10695	-348	-55	-8	0
197 QP problems	43	116	38	-858	-6	1	12	2893
253 BC problems	53	170	30	-257	-2	2	24	3685
50 LC problems	6	30	14	-1087	-33	-3	5	15
153 NC problems	10	113	30	-10608	-24	-2	-1	360

Table 10: Detailed comparison of the default versions of LANCELOT and MINOS for CPU time

Recall that the comparison involves problems whose objective and constraints are very cheap to evaluate. The CPU times therefore mostly reflect the amount of computation that is internal to the packages themselves. If the functions were more expensive to compute, the CPU time performances would more closely match those reported above for function evaluations.

Except for linear problems, where the superiority of MINOS is clear, the interquartile distances show that both packages have a rather comparable performance in CPU time, with a slight advantage for LANCELOT in the quadratic and unconstrained cases, and for MINOS in the nonlinearly constrained case.

Tables 11 and 12 give the CPU performances according to the number of degrees of freedom of each problem. The picture is very clear and quite striking here: MINOS takes less time on problems with relatively few degrees of freedom while the opposite is true for problems where this number is higher. When there are many degrees of freedom, MINOS remains significantly better only for the class of problems involving linear constraints.

	LANCELOT	MINOS
$\boxed{< 50 \text{ superbasics}}$		
73 LP problems	45812	466
143 QP problems	1553	84
157 BC problems	331	31
39 LC problems	152	12
138 NC problems	12088	169
$\boxed{\geq 50 \text{ superbasics}}$		
54 QP problems	668	4619
92 BC problems	2423	14338
11 LC problems	1343	205
7 NC problems	95	922

Table 11: Total CPU time for the default versions of LANCELOT and MINOS according to the number of superbasic variables

In the detailed comparison of Table 12, one notices that LANCELOT may be very slow on the few problems where it is slower than MINOS. A first, somewhat extreme, example is the solution of the BIGBANK linearly constrained problem, which features a nonlinear objective function and linear constraints of the network type. LANCELOT takes 1188 seconds, while MINOS, which exploits both the sparsity and the linearity of the constraints, requires only 100 seconds. A second such example is the CORKSCRW(4506) nonlinearly constrained problem, with a nonlinear objective function, 3000 linear equality, 500 nonlinear inequality and 4000 bound constraints. LANCELOT converges in 10694 seconds while MINOS apparently exploits the fact that there are n active constraints at the solution (no degrees of freedom), and only takes 86 seconds to solve the problem! On the other hand, MINOS is also sometimes much slower than LANCELOT. This happens, for instance, on the GRIDNETB(7564) quadratic problem , where MINOS takes 2998 seconds while LANCELOT takes 104. Note that this problem has 3721 degrees of freedom at the solution, which is the largest number amongst all the tested quadratic programs. Another cause of relative slowness for MINOS is strong nonlinearity, which appears, for example, in the unconstrained least-squares problems MSQRTALS(529) and MSQRTBLS(529), where it requires 3374 and 3751 seconds, respectively, compared to the 87 and 65 seconds needed for LANCELOT to converge.

5 Conclusions

The numerical tests described here confirm our belief that to a large extent, LANCELOT and MINOS complement each other. Our experiments *with the "default" versions of the packages* have shown MINOS to be a clear winner for linear programs. For the other classes, MINOS has proved to be slightly more reliable on linearly constrained problems, while LANCELOT has a small advantage for unconstrained and nonlinearly constrained cases. However the difference in reliability is not very large. From the efficiency point of view, LANCELOT is a clear winner if the evaluations of the problem functions are expensive

	LANCELOT wins	tie	MINOS wins	MINOS – LANCELOT min	q1	med	q3	max
< 50 superbasics								
73 LP problems	0	11	62	-10695	-348	-55	-8	0
143 QP problems	5	110	28	-828	-28	-1	-1	2
157 BC problems	1	142	14	-257	-5	-2	-1	1
39 LC problems	0	28	11	-78	-34	-3	-1	0
138 NC problems	4	107	27	-10608	-24	-6	-1	9
$\geq$ 50 superbasics								
54 QP problems	38	6	10	-253	1	9	17	2893
92 BC problems	50	26	16	-221	0	4	41	3685
11 LC problems	6	2	3	-1087	-107	6	11	15
7 NC problems	6	1	0	4	5	80	210	360

Table 12: Detailed comparison of the default versions of LANCELOT and MINOS for CPU time

or, to a lesser extent, if the number of degrees of freedom in the problem is large. MINOS, on the other hand, appears to be faster when function evaluations are inexpensive or when there are few degrees of freedom. Again, these conclusions are only valid for *default versions* of the packages.

None of the conclusions is very surprising, considering the differences in the algorithms themselves. However, the results are potentially very useful to users of mathematical programming software in that they give a quantitative indication of the type of differences in reliability and efficiency that can be expected.

Ideally, one would pursue these experiments by selecting, for each package, other algorithmic variants than the defaults, as this of course substantially affects both reliability and efficiency. For LANCELOT, such a study has been presented by Conn et al. (1996). A key comparison there is the effect of running LANCELOT with and without second derivatives, where it is observed that the use of exact second derivatives is only slightly superior to the use of the best (partitioned, symmetric-rank-one) secant update formula. However, comparing several variants of MINOS with several variants of LANCELOT would be a major exercise, unlikely to reveal options that perform consistently better than those used here. Since our results clearly show that there is a role to play for both packages, the authors feel that, although continued experience will undoubtedly result in deeper understanding, the best advice they can offer to practitioners is to try both packages themselves, and determine which, and with what options, is best for their particular applications.

References

B. M. Averick and J. J. Moré. User guide for the MINPACK-2 test problem collection. Technical Report ANL/MCS-TM-157, Argonne National Laboratory, Illinois, USA, 1991.

B. M. Averick, R. G. Carter and J. J. Moré. The MINPACK-2 test problem collection (preliminary version). Technical Report ANL/MCS-TM-150, Argonne National Laboratory, Illinois, USA, 1991.

I. Bongartz, A. R. Conn, N. I. M. Gould and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, **21**(1), 123–160, 1995.

I. Bongartz, A. R. Conn, N. I. M. Gould, M. A. Saunders and Ph. L. Toint. A numerical comparison between the LANCELOT and MINOS packages for large-scale nonlinear optimization: the complete results. Technical Report 97/14, Department of Mathematics, FUNDP, Namur, Belgium, 1997.

A. G. Buckley. Test functions for unconstrained minimization. Technical Report CS-3, Computing Science Division, Dalhousie University, Dalhousie, Canada, 1989.

A. R. Conn, N. I. M. Gould and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, **25**(2), 433–460, 1988. See also same journal **26**, 764-767, 1989.

A. R. Conn, N. I. M. Gould and Ph. L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project. *In* R. Glowinski and A. Lichnewsky, eds, 'Computing Methods in Applied Sciences and Engineering', pp. 42–54, SIAM, Philadelphia, 1990.

A. R. Conn, N. I. M. Gould and Ph. L. Toint. A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, **28**(2), 545–572, 1991.

A. R. Conn, N. I. M. Gould and Ph. L. Toint. LANCELOT*: a Fortran package for large-scale nonlinear optimization (Release A).* Number 17 *in* 'Springer Series in Computational Mathematics'. Springer Verlag, Heidelberg, Berlin, New York, 1992*a*.

A. R. Conn, N. I. M. Gould and Ph. L. Toint. On the number of inner iterations per outer iteration of a globally convergent algorithm for optimization with general nonlinear equality constraints and simple bounds. *In* D. F. Griffiths and G. A. Watson, eds, 'Numerical Analysis 1991', number 260 *in* 'Pitman Research Notes in Mathematics Series', pp. 49–68, Longman Scientific and Technical, Harlow, England, 1992*b*.

A. R. Conn, N. I. M. Gould and Ph. L. Toint. Numerical experiments with the LANCELOT package (Release A) for large-scale nonlinear optimization. *Mathematical Programming*, **73**(1), 73–110, 1996.

A. R. Conn, N. I. M. Gould, M. Lescrenier, and Ph. L. Toint. Performance of a multi-frontal scheme for partially separable optimization. *In* 'Advances in optimization and numerical analysis, Proceedings of the Sixth workshop on Optimization and Numerical Analysis, Oaxaca, Mexico', number 275 *in* 'Mathematics and its Applications', pp. 79–96, Kluwer Academic Publishers, Dordrecht, 1994.

R. S. Dembo. A primal truncated-Newton algorithm with application to large-scale non-linear network optimization. Technical Report 72, Yale School of Management, Yale University, New Haven, USA, 1984.

I. S. Duff, A. M. Erisman and J. K. Reid. *Direct methods for sparse matrices.* Oxford University Press, Oxford, 1986.

D. M. Gay. Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Newsletter, December 1985.

P. E. Gill, W. Murray and M. A. Saunders. SNOPT: an SQP algorithm for large-scale constrained optimization. Technical Report SOL97-3, Department of EESOR, Stanford University, Stanford, California 94305, USA, 1997.

A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, **39**, 429–448, 1982.

M. Gulliksson. *Algorithms for Nonlinear Least Squares with Applications to Orthogonal Regression.* PhD thesis, Institute of Information Processing, University of Umeå, S-901 87 Umeå, Sweden, 1990.

W. Hock and K. Schittkowski. *Test Examples for Nonlinear Programming Codes.* Number 187 *in* 'Lecture Notes in Economics and Mathematical Systems'. Springer Verlag, Heidelberg, Berlin, New York, 1981.

J. J. Moré. Recent developments in algorithms and software for trust region methods. *In* A. Bachem, M. Grötschel and B. Korte, eds, 'Mathematical Programming: The State of the Art', pp. 258–287, Springer Verlag, Heidelberg, Berlin, New York, 1983.

J. J. Moré and G. Toraldo. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, **1**(1), 93–113, 1991.

J. J. Moré, B. S. Garbow and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, **7**(1), 17–41, 1981.

B. A. Murtagh and M. A. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, **14**(1), 41–72, 1978.

B. A. Murtagh and M. A. Saunders. A projected Lagrangian algorithm and its implementation for sparse non-linear constraints. *Mathematical Programming Studies*, **16**, 84–117, 1982.

B. A. Murtagh and M. A. Saunders. MINOS 5.4 USER'S GUIDE (revised). Technical Report SOL83-20R, Department of Operations Research, Stanford University, Stanford, California 94305, USA, 1993. Revised 1995.

K. Schittkowski. *More Test Examples for Nonlinear Programming Codes.* Number 282 *in* 'Lecture Notes in economics and mathematical systems'. Springer Verlag, Heidelberg, Berlin, New York, 1987.

Ph. L. Toint. Test problems for partially separable optimization and results for the routine PSPMIN. Technical Report 83/4, Department of Mathematics, FUNDP, Namur, Belgium, 1983.

Ph. L. Toint and D. Tuyttens. On large-scale nonlinear network optimization. *Mathematical Programming, Series B*, **48**(1), 125–159, 1990.