Assessing the Potential of Interior Methods for Nonlinear Optimization

José Luis Morales[1], Jorge Nocedal[2], Richard A. Waltz[2], Guanghui Liu[3], and Jean-Pierre Goux[2]

[1] Departamento de Matemáticas, Instituto Tecnológico Autónomo de México, Río Hondo 1, Col Tizapán San Angel, México D.F. CP 01000, México. †
[2] Electrical and Computer Engineering Department, Northwestern University, Evanston IL 60208, USA. ‡
[3] Industrial Engineering and Management Sciences Department, Northwestern University, Evanston IL 60208, USA.‡

Abstract. A series of numerical experiments with interior point (LOQO, KNITRO) and active-set SQP codes (SNOPT, filterSQP) are reported and analyzed. The tests were performed with small, medium-size and moderately large problems, and are examined by problem classes. Detailed observations on the performance of the codes, and several suggestions on how to improve them are presented. Overall, interior methods appear to be strong competitors of active-set SQP methods, but all codes show much room for improvement.

1 Introduction

The goal of this paper is to evaluate the efficiency and robustness of interior methods for nonlinear programming, and to assess their potential in the solution of very large problems. The hope is that this study will highlight some of the strengths and weaknesses of interior methods, suggest avenues for improving their performance, and identify classes of problems for which they appear to be well suited. In order to conduct this evaluation we will perform numerical tests using two established interior point packages, LOQO and KNITRO, and will compare their performance with the state-of-the-art sequential quadratic programming codes, SNOPT and filterSQP. This will also allow us to make some new observations about the numerical behavior of these two active-set SQP algorithms.

The nonlinear programming problem will be written as

$$\min_x \quad f(x) \tag{1a}$$
$$\text{s.t.} \quad h(x) = 0 \tag{1b}$$
$$g(x) \leq 0. \tag{1c}$$

† This author was supported by CONACyT, Asociación Mexicana de Cultura AC, the Fulbright Comission, and National Science Foundation grant CCR 9907818.
‡ These authors were supported by National Science Foundation grant CCR 9907818, and by Department of Energy grant DE-FG02-87ER25047-A004.

We will consider only problems where the objective $f : \mathbb{R}^n \to \mathbb{R}$ and constraint functions $h : \mathbb{R}^n \to \mathbb{R}^m$, $g : \mathbb{R}^n \to \mathbb{R}^q$ are smooth. Our testing will be done primarily on small, medium-sized and moderately large problems from the CUTE collection. By this we mean problems that have up to 10,000 variables or constraints.

A variety of interior (or barrier) methods for nonlinear programming have been proposed in the last few years [1,5,7,12,15,17,28,29], but only a few implementations are currently available for public use. We have chosen to experiment with the software packages LOQO [28] and KNITRO [5], which are available through the NEOS system [9], as well as in machine executable form. These packages implement two distinct interior point approaches: LOQO is a line search algorithm that has much in common with interior algorithms for linear and convex quadratic programming, whereas KNITRO is a trust region method which uses sequential quadratic programming methodology to treat the barrier sub-problems. LOQO and KNITRO also differ significantly in the linear algebra techniques used for the step computation.

One of the main objectives is to try to determine if the interior methods implemented in these two packages are competitive with established codes for nonlinear programming. Several production packages, such as LANCELOT [8], MINOS [20], DONLP [24,25] and CONOPT [11] would have served our purposes, but we have chosen SNOPT [18] and filterSQP [14] because we have experience with both codes, and because they constitute two complementary approaches to SQP algorithms: SNOPT is a line search method using convex models and filterSQP is a trust region method that allows the model to be non-convex.

Before proceeding, we must clarify what this study can and cannot achieve. First, the software packages used in our tests are regularly being updated and their performance could change significantly after a new release. We can expect this situation to continue in the near future; even SNOPT, which follows a well established active-set SQP approach, is still being modified in significant ways. Therefore we warn the reader against trying to derive a ranking of the codes from this study. Not only is such a ranking dubious given that it is based on a particular set of problems, but even within this testing environment relative performance of the codes can change at any time.

Second, our goal is to assess the effectiveness of *optimization methods*, more specifically interior methods and active-set SQP methods, and not simply to evaluate the performance of specific *software implementations*. Of course, it is not possible to entirely isolate the performance of a method from its particular implementation because implementation aspects (e.g. sparse elimination techniques) impact performance in significant ways. Moreover, algorithms within a given class of methods can vary due to various, lower-level algorithmic decisions, such as the line search strategy and the choice of merit function.

Nevertheless, by judiciously controlling the testing environment, we will be able to make some observations and state a few conjectures about the

methods of interest. To focus on algorithmic features we will independently study three classes of problems: (i) unconstrained, (ii) equality constrained, and (iii) general constrained problems, excluding linear and quadratic programs. We will also pay attention to problem conditioning.

2 Overview of the Codes

We now outline the algorithms implemented in the four packages used in this study. More details about them will be presented in subsequent sections as we analyze their performance on various classes of problems.

2.1 LOQO

LOQO 4.05 [28] originated as an interior point code for linear and quadratic programming, and has been extended to solve nonlinear problems. Our interest here is only in its behavior as a nonlinear solver. LOQO implements a line search primal-dual interior point method. It uses second derivatives of the objective function and constraints.

The nonlinear programming problem (1) is replaced by a sequence of barrier problems of the form

$$\min_{x,s} \quad f(x) - \mu \sum_{i=1}^{q} \ln s_i \tag{2a}$$

$$\text{s.t.} \quad g(x) + s = 0, \tag{2b}$$

where $\mu > 0$ is the barrier parameter, s is a vector of slacks, and q denotes the number of inequality constraints. (For simplicity of exposition, we assume in this section that all the constraints are inequalities.) The algorithm computes a search direction d by factoring a system of the form

$$\begin{bmatrix} -\nabla^2 L(x,y) & A(x)^T \\ A(x) & SY^{-1} \end{bmatrix}, \tag{3}$$

where $A(x)$ stands for the Jacobian of the constraints $g(x)$, y is the vector of Lagrange multipliers, and $L(x,y)$ denotes the Lagrangian, $L(x,y) = f(x) - y^T g(x)$. Here we use the notation that S is a diagonal matrix with $S_{ii} = s_i$ (and similarly for Y^{-1}).

Once the search direction is computed, a backtracking line search is performed to guarantee sufficient decrease in the ℓ_2 penalty function,

$$\psi(x,s;\mu,\nu) = f(x) - \mu \sum_{i=1}^{q} \ln s_i + \nu \|g(x) + s\|_2^2, \tag{4}$$

where $\nu > 0$ is the penalty parameter. This is not a standard merit function, since a minimum of ψ is normally not a KKT point for (1). Moreover,

a heuristic is used to determine the value of the penalty parameter ν; it attempts to set $\nu = 0$, and increases it as necessary to ensure that the direction d is a descent direction for ψ.

A dynamic rule (see (11)) is used to update the value of the barrier parameter μ at each iteration. The KKT system (3) is solved using a sparse LDL^T factorization [23]. In order to handle non-convex problems, LOQO monitors the Cholesky factors of

$$\nabla^2 L(x, y) + A(x)^T S^{-1} Y A(x),$$

and if these factors are not sufficiently positive, a multiple λ of the identity matrix is added to $\nabla^2 L$. Since a suitable value of λ is not known in advance, several trial values and refactorizations may be needed.

There is no global convergence theory for the algorithm implemented in LOQO, and it would be difficult to establish one without restrictive assumptions. The main obstacles are the use of a dynamic rule for changing the barrier parameter μ, and the use of the penalty function (4) as a merit function.

2.2 KNITRO

KNITRO 1.00 [5] implements a primal-dual interior point method with trust regions. It makes use of second derivative information and, as in LOQO, employs slack variables to formulate barrier subproblems of the form (2). KNITRO differs from most interior methods proposed in the literature in that it does not compute the step by factoring the primal-dual matrix (3), or a variation of it. Instead it applies an SQP-type method for equality constrained optimization to compute steps d that approximately minimize the barrier problem (2). When this minimization is completed, the barrier parameter μ is decreased by a fixed factor of 5.

Each step is computed as the sum $d = v + t$ of a normal component v whose objective is to improve feasibility, and a tangential component t which aims toward optimality [3,22]. The main cost of the iteration lies in the computation of the tangential component, which is performed using a projected conjugate gradient (CG) iteration. Each CG iteration involves two main computations: (i) a projection, which makes use of the factors of the matrix

$$\begin{bmatrix} I & 0 & A(x)^T \\ 0 & I & S \\ A(x) & S & 0 \end{bmatrix}, \tag{5}$$

and, (ii) a product of the Hessian $\nabla^2 L$ times a vector. The normal step v and the Lagrange multipliers y are inexpensive to compute, since they only requires one backsolve each using the factors of (5).

After the step d is computed, the algorithm tests whether it provides sufficient decrease in the (non-differentiable) merit function

$$\phi(x, s; \mu; \nu) = f(x) - \mu \sum_{i=1}^{q} \ln s_i + \nu \|g(x) + s\|_2, \qquad \nu > 0. \tag{6}$$

If there is no decrease, the step is rejected and the trust region is reduced. Second-order correction steps are computed under certain circumstances to prevent the rejection of steps.

A global convergence theory of an algorithm similar to that implemented in KNITRO is presented in [4], whereas rate of convergence results are given in [6].

2.3 SNOPT

SNOPT 5.3-4 [18] is a line search SQP method in which the search direction is determined by an active set method for convex quadratic programming. The line search enforces decrease of an augmented Lagrangian function, which in the case when all the constraints are inequalities, takes the form

$$\mathcal{M}(x, s, y) = f(x) - y^T(g(x) + s) + \tfrac{1}{2}(g(x) + s)^T D(g(x) + s), \tag{7}$$

where D is a diagonal matrix of penalty parameters. The code requires only first derivatives of the objective function and constraints, and maintains a dense approximation to the reduced Hessian of a *modified* Lagrangian, using the BFGS updating formula. When the number of variables is large the algorithm automatically switches to a limited memory BFGS mode.

All iterates satisfy the linear constraints in the problem. To promote global convergence, the iteration may enter *flexible mode* in which the constraints are relaxed using slack variables. "Linear flexible mode" is invoked to determine if the linear constraints are compatible, and "nonlinear feasible mode" is used when the quadratic subproblem is infeasible or unbounded, or when the Lagrange multiplier estimates y become large.

To ensure that the quasi-Newton approximations are positive definite, SNOPT modifies the correction vectors used in the update, if necessary; under certain conditions it may even skip the update. Global convergence results are difficult to establish for SNOPT, mainly because the merit function (7) is treated as a function of x, s, y, and is therefore not minimized by a solution point.

2.4 FilterSQP

The FilterSQP 1.0 [14] package implements a trust region SQP method which makes use of second derivatives of the objective function and constraints. The step is computed by means of an active set method for indefinite quadratic

programming (QP). If a direction of negative curvature is detected, the QP solver follows it until a constraint or the trust region is encountered. This process continues until a local minimum of the quadratic model is obtained. A novel feature is the use of a "filter", a step-acceptance mechanism that eliminates the need for a merit function [14]. If a step is not accepted by the filter, a second-order correction step is typically attempted. The trust region is defined by an ℓ_∞ norm, and its update strategy is simpler than in trust region methods that use merit functions.

The algorithm contains a *feasibility restoration phase* which is invoked when the constraints of the subproblem are infeasible, or when the filter mechanism requires that progress on the constraints be made. The feasibility restoration phase has some analogies with the flexible mode employed in SNOPT; among the main differences are the use of a trust region, and the fact that the objective function f is ignored during the feasibility restoration phase, whereas it is part of the objective function in the flexible mode of SNOPT.

Both SNOPT and filterSQP employ sophisticated update strategies in their active set quadratic programming solvers to carry information from one active set to the next. Global convergence results have been established for variations of the filter mechanism implemented in filterSQP [13], but this is still an ongoing area of research.

3 Testing Environment

We performed tests using 481 problems from the CUTE collection [2] that have been translated into AMPL [16] by Benson and Vanderbei [27]. We selected a subset of the 740 problems available in the Benson-Vanderbei test set as of Dec., 2000, according to the following criteria. All linear programming (LP) and quadratic programming (QP) were excluded since we feel that these problems can be treated more efficiently by specifically designed methods and software. We also excluded all feasibility problems (without an objective function). Finally, to have a more manageable set of general constrained problems, we excluded all problems whose only constraints were simple bounds.

Table 1 in section 6 gives the number of problems in each category; see www.ece.northwestern.edu/~nocedal/knitro. Since CPU times can be dominated by set-up costs for small problems, we will only report CPU times for problems with more than 100 variables. Function evaluations will be reported for all problems.

It was impossible to apply a uniform stopping test on all codes, since their form cannot be changed by the user. Nevertheless, we tried to make them as similar as possible, and in fact modified KNITRO's stopping test to be almost identical to that of SNOPT. We did notice differences in the precision that was obtained by the various methods in our tests, but they are perhaps not significant enough to be of concern (unless otherwise noted).

All the tests were performed on a Sun Ultra 5, with 385Mb of memory running SunOS 5.7. The four codes were run in double precision using all their default settings. A limit of 30 minutes of CPU time, and 1000 iterations, were imposed for each problem; if one of these limits was reached the code was considered to have failed. The stopping tolerance was set at 10^{-6} on all solvers.

The fact that SNOPT uses only first-derivative information[1] and approximates second derivatives by quasi-Newton updating introduces an undesirable disparity with the other three codes. We will nevertheless take advantage of this difference to assess the effectiveness of quasi-Newton approximations.

4 Numerical Results

We will now proceed with the examination of results, by problem class. In order to facilitate their interpretation, we highlight the key features of each method when applied to a specific problem class.

4.1 Unconstrained Problems

When applied to an unconstrained problem, the algorithms implemented in LOQO, KNITRO, and filterSQP reduce to Newton methods, whereas SNOPT amounts to a quasi-Newton method.

LOQO is an (exact) Newton method with a backtracking line search requiring sufficient decrease in f. The search direction is computed using a sparse LDL^T factorization of the Hessian $\nabla^2 f(x)$. Negative curvature is handled by adding a multiple of the identity to the Hessian matrix, if necessary.

KNITRO is an (inexact) Newton method with a spherical trust region. The step is computed by an unpreconditioned conjugate gradient iteration, which is terminated using Steihaug's rule [26]. The step is accepted if it provides sufficient reduction in f.

SNOPT is a quasi-Newton method with a backtracking line search requiring sufficient reduction in f. The Hessian approximation is computed by standard BFGS updating if $n \leq 75$, and by a (restarted) limited memory BFGS updating otherwise. Positive definiteness of the Hessian approximation is enforced by modifying the update formula when necessary. An evaluation of the limited memory strategy used in SNOPT is given in [19].

FilterSQP is an (exact) Newton method using an ℓ_∞ trust region. The filter mechanism reduces, in the unconstrained case, to a standard sufficient decrease condition on f. The search direction is computed by minimizing a quadratic model of the objective subject to the trust region constraints. Indefiniteness is exploited by ensuring that the step is a local minimizer of the quadratic model of f subject to the trust region constraint.

[1] Our numerical tests were conducted in the fall of 2000; at that time a second-derivative version of SNOPT was not available.

All the results in this paper will be presented using the performance profiles proposed by Dolan and Moré [10]. Let $t_{p,s}$ denote the time to solve problem p by solver s. Then define the ratios

$$r_{p,s} = \frac{t_{p,s}}{t_p^*}, \tag{8}$$

where t_p^* is the lowest time required by any code to solve problem p. If a code does not solve a problem, the ratio $r_{p,s}$ is assigned a large number, M. Then, define the logarithmic performance profile for each code s, as

$$\pi_s(\tau) = \frac{\text{no. of problems s.t. } \log_2(r_{p,s}) \leq \tau}{\text{total no. of problems}}, \qquad \tau \geq 0. \tag{9}$$

This performance profile will also be used to analyze the number of function evaluations required by the four codes.

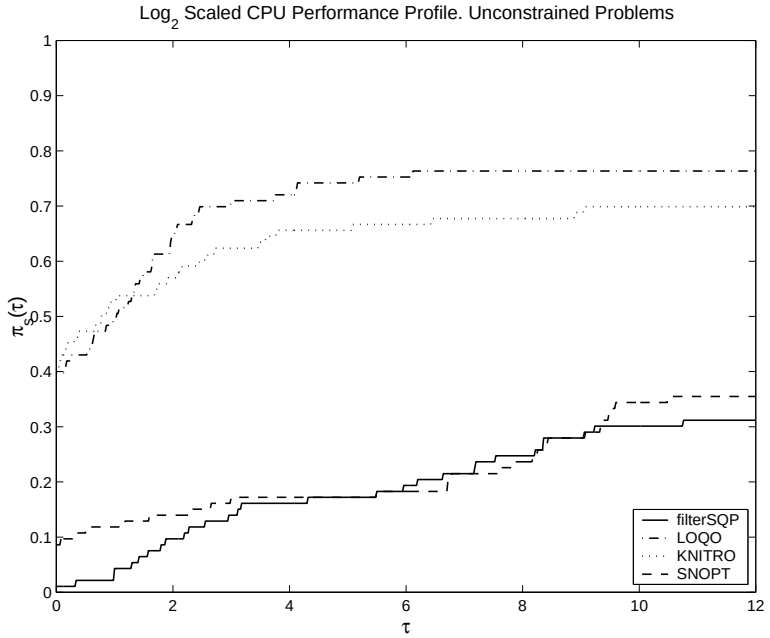

Fig. 1. Performance, in terms of CPU time, on 180 unconstrained problems.

In Figures 1 and 2 we plot performance profiles for CPU time and function evaluations, for the unconstrained problems. We recall that CPU profiles are reported only for problems with $n \geq 100$; therefore to study robustness of the codes it is preferable to consider the profiles reporting function evaluations.

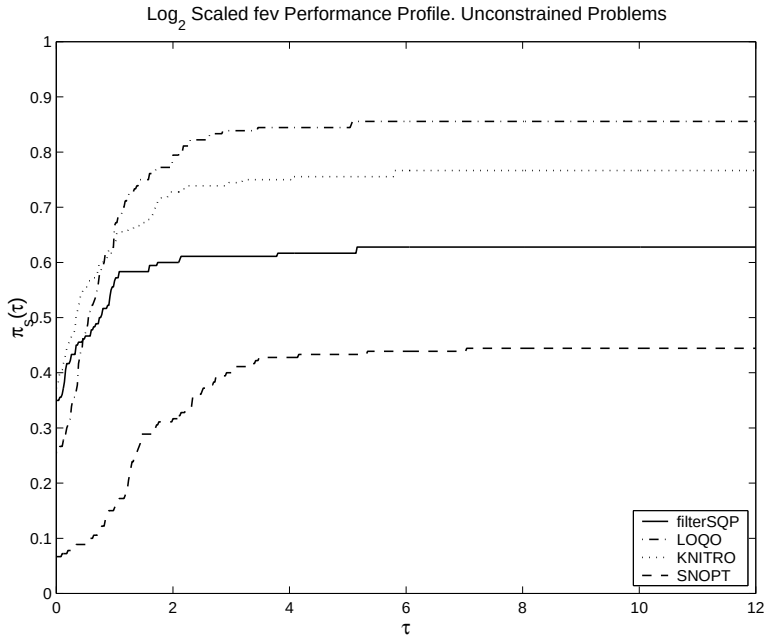

Fig. 2. Peformance, in terms of function evaluations, on 93 unconstrained problems.

We make the following observations based on these figures, and on a problem by problem examination of the runs. (i) The CPU times for filterSQP are unexpectedly high, and are almost certainly caused by inefficiencies in the factorization algorithm. (ii) The trade-offs between the iterative method used in KNITRO for the step computation, and the direct method in LOQO, can be clearly observed in these test problems (consider e.g. the range $\tau \in [0, 2]$). An examination of KNITRO's output shows that the problems for which it required significantly more time than LOQO are mostly ill-conditioned, demanding a large number of CG iterations. KNITRO's performance would improve on difficult problems with the use of a preconditioner. (iii) SNOPT required consistently more function evaluations than the other codes, as is expected from a quasi-Newton method. Computing times would likely improve by reducing the amount of correction pairs stored in the limited memory matrix, but robustness could deteriorate. (iv) LOQO performs well in terms of function evaluations[2] and time, but the results must be interpreted with caution because its stopping test for unconstrained problems is often less demanding than used in the other codes. Using uniform criteria, the four codes

[2] LOQO 4.05 makes one unnecessary function evaluation at every iteration, something that could easily be corrected. We adjusted for this by discounting this extra evaluation in our results.

are likely to be equally robust. (v) KNITRO and filterSQP are very similar in terms of function evaluations. The differences in the shape of the trust region and the inexactness vs exactness of the Newton iteration do not impact the number of function evaluations in these problems.

4.2 Equality Constrained Problems

When applied to equality constrained problems (1a)-(1b), the algorithms implemented in the four codes belong to the class of sequential quadratic programming (SQP) methods. There are, nevertheless, important algorithmic differences.

LOQO is a line search SQP method. The search direction is obtained by applying Newton's method to the KKT conditions of (1a)-(1b). This gives an iteration of the form

$$\begin{bmatrix} -\nabla^2 L(x,y) & A(x)^T \\ A(x) & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} \nabla L(x,y) \\ -h(x) \end{bmatrix}, \tag{10}$$

where A denotes the Jacobian of the equality constraints. The linear system in (10) is first reduced, and then solved using a sparse factorization; in case of indefiniteness, a multiple of the identity matrix is added to $\nabla^2 L(x,y)$. The steplength is required to provide sufficient reduction in the classical ℓ_2 merit function.

KNITRO is an SQP method with trust regions. The step d is computed by the Byrd-Omojokun approach [3,22]: d is the sum of a normal component v and a tangential component t, as described in section 2.2. The tangential component is computed using an (unpreconditioned) projected conjugate gradient iteration, in which projections are performed using the factors of a matrix of the form (5), but were the slacks are not present. The CG iteration follows the first direction of negative curvature to the boundary of the trust region. The merit function has the form (6).

SNOPT is a line search, quasi-Newton, SQP method. The search direction is obtained by solving a system of the form (10) in which $\nabla^2 L$ is replaced by a positive definite quasi-Newton matrix. The line search requires sufficient decrease in the augmented Lagrangian merit function. The elastic mode handles problems in which the constraints are ill-conditioned.

In filterSQP, the step is computed by solving an SQP subproblem with trust regions. Negative curvature is exploited since the step is always a local minimizer of the quadratic model where no directions of negative curvature exist. Step acceptance is determined by the filter mechanism.

The results on equality constrained problems are given in Figures 3 and 4. They should be taken with caution because the sample of problems is small.

(i) Once more, we observe large CPU times for filterSQP, even though it performs well in terms of function evaluations. The cause, as in the unconstrained case, is likely to be inefficiencies in the factorization. FilterSQP

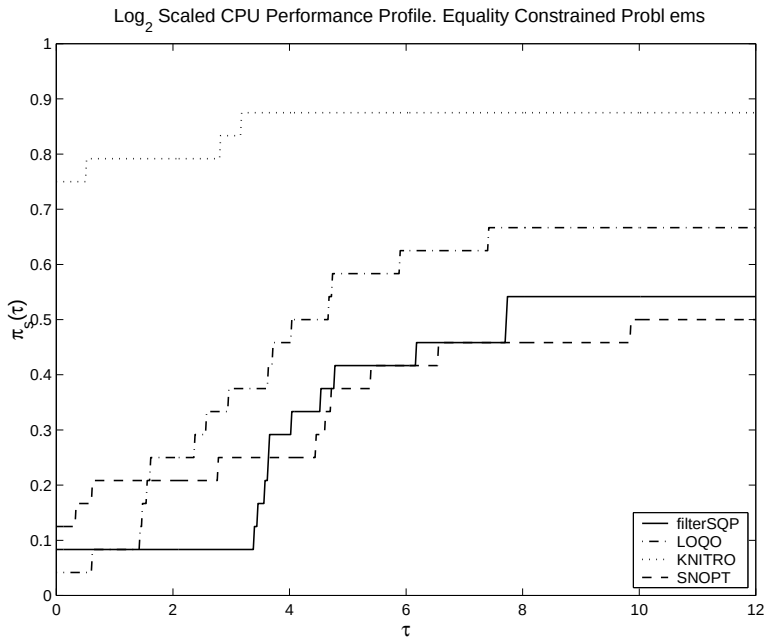

Fig. 3. Comparison, in terms of CPU time, on 24 equality constrained problems.

applies many second-order corrections on some problems, but we have verified that they are not the cause for the large times. (ii) It is not surprising that SNOPT does not perform well in terms of function evaluations, due to the lack of second-derivative information. (iii) One of the distinctive features of KNITRO, namely its step computation mechanism using the CG method and projections, is fully active in equality constrained problems. The results indicate that it performs efficiently on this test set. (iv) the filter mechanism is also active now, and appears to work well, but a precise evaluation of its benefits is difficult to make: whereas FilterSQP is clearly more economical in terms of function evaluations than LOQO and SNOPT, a problem-by-problem comparison with KNITRO's runs suggests that both codes have smilar step-acceptance rates. The efficiency of the filter mechanism must, therefore, be further investigated, preferably using a code the provides both filter and merit function options.

5 General Nonlinear Programs

We will now consider problems containing inequality constraints, a setting in which the four methods exhibit major algorithmic differences.

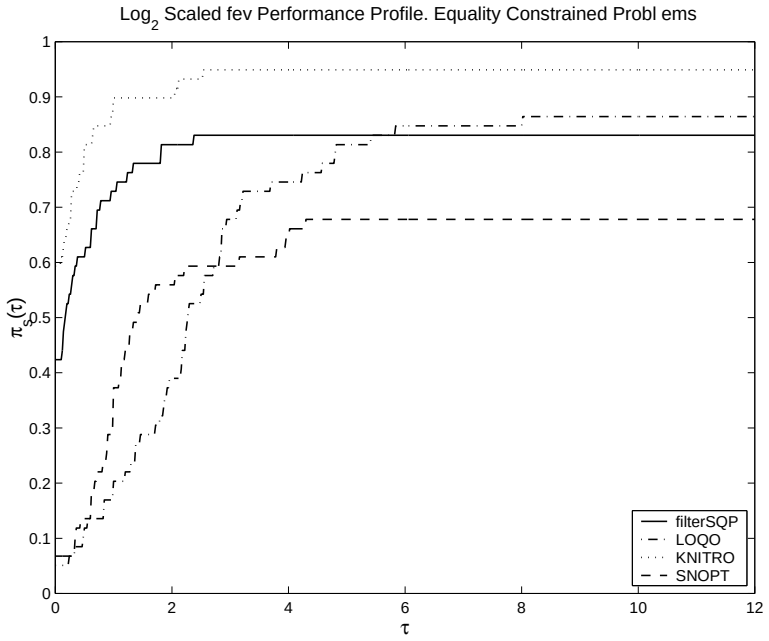

Fig. 4. Comparison, in terms of function evaluations, on 59 equality constrained problems.

LOQO is a line search, primal-dual interior method for nonlinear programming. The cost of the iteration is dominated by the factorization of the primal-dual matrix (3). The barrier parameter μ is redefined at every iteration using the rule:

$$\mu = 0.1 \min\left(0.05, \frac{1-\xi}{\xi}, 2\right)^3 \frac{s^T y}{q}, \quad \text{with} \quad \xi = \frac{\min_i s_i y_i}{y^T s/q}, \tag{11}$$

where q is the number of inequality constraints. This rule permits increases in the barrier parameter from one iteration to the next. A sufficient reduction in the penalty function (4) is required at each iteration.

KNITRO is a trust region, primal-dual interior method. It can be considered a path-following interior method since the barrier parameter is changed only after each barrier subproblem is solved to a given accuracy, and at that point it is decreased by a (fixed) factor of 5. KNITRO uses the nondifferentiable merit function (6), and applies second-order correction (SOC) steps to avoid the Maratos effect. The cost of the iteration is spread out, as in the equality constrained case, between three computations:

1. the factorization of the projection matrix (5),

2. $ncg+2$ backsolves using the factors of (5), where ncg denotes the number of CG steps. (One backsolve to compute the normal step v, one to compute Lagrange multipliers y, and ncg backsolves to perform projections during the CG iterations (iii) ncg products of the Hessian $\nabla^2 L$ with a vector.

   FilterSQP is a trust region SQP method, and SNOPT is a line search SQP method. Their main features have been described in the previous sections. For problems with general constraints, the active-set quadratic programming solvers of both methods are fully active.

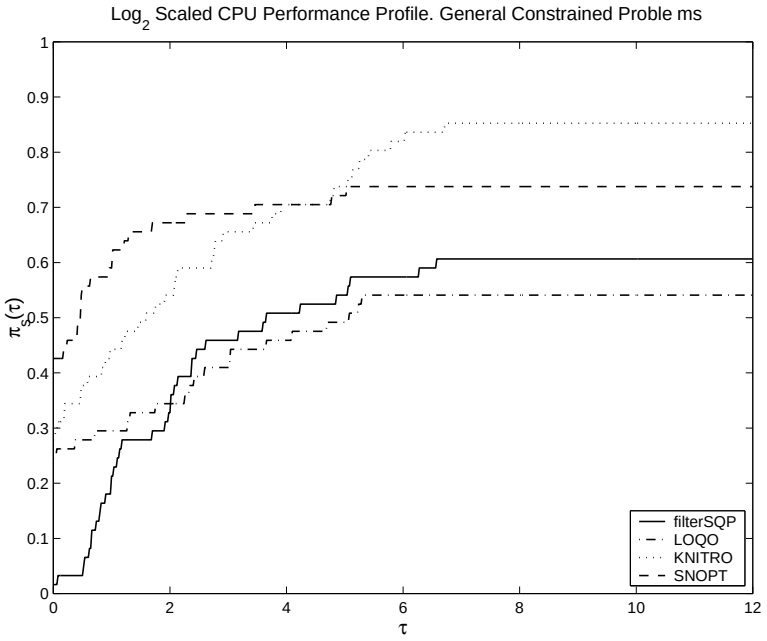

Fig. 5. Comparison, in terms of CPU time, on 61 constrained problems.

The results are presented in Figures 5 and 6.

(i) We observe that SNOPT performs quite well compared to the other three codes despite using only first derivatives. This is remarkable, and contrasts with our observations for unconstrained and equality constrained problems. To test whether this was due to the fact that most of our problems are not very large, we selected 16 problems from the CUTE collection and created three versions of each, with increasing dimensions ranging generally in the thousands. We compared KNITRO and SNOPT, but were not able to discern any clear trend. Whereas in some of the problems SNOPT's cpu time

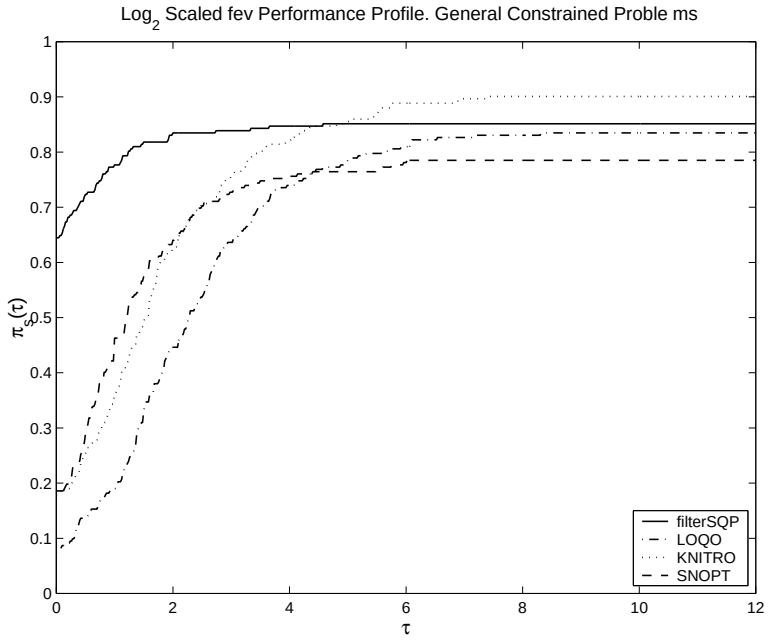

Fig. 6. Comparison, in terms of function evaluations, on 242 constrained problems.

increased much more rapidly than KNITRO's time, on about as many problems SNOPT remained the fastest of the two codes. (ii) The times of filterSQP are high, especially since it requires a small number of iterations compared to the other solvers. On the other hand, filterSQP performs exceptionally well in terms of function evaluations. Whether this can be attributed to efficiencies in the filter mechanism, remains to be established. (iii) SNOPT's performance in terms of function evaluations must be interpreted with caution because the code does not report them in the same manner as the other codes, and may undercount them in some cases.

We payed close attention to the relative performance of LOQO and KNITRO, to try to evaluate their main algorithmic features. (iv) We observed that KNITRO tends to perform fewer iterations (iterations are not reported in our tables). Considering, for example, all problems with $n \geq 50$ we observed that KNITRO required fewer iterations in about 2/3 of the problems. The difference persisted when looking at problems sets of other dimensions. KNITRO's advantage in this respect could be attributed to the step computation procedure, or more likely, to its path-following approach combined with a conservative rule for decreasing the barrier parameter. (v) LOQO's time per iteration was lower than KNITRO's in a significant percentage of the problems. LOQO's advantage in this respect is attributed to a faster fac-

torization, in some cases, and to a large cost of KNITRO's CG steps, in other cases (usually problems with an ill-conditioned reduced Hessian). (vi) We also noted that KNITRO's strategy of setting the initial value barrier parameter always as $\mu = 1$ was highly detrimental in some cases; the adaptive choice used by LOQO appears to be more effective in the early iterations.

6 Final Remarks

Our experiments suggest that interior methods constitute a powerful approach for solving medium to moderately large nonlinear programming problems. The two (rather different) interior algorithms implemented in LOQO and KNITRO, appear to be competitive, in terms of robustness and efficiency, with the active-set SQP algorithms implemented in SNOPT and filterSQP. In Table 1 we summarize robustness statistics for the four codes, focusing also on the problems that are not small ($n \geq 100$).

		FILTER	LOQO	KNITRO	SNOPT	sample size
unconstrained	all	113	154	138	80	180
	$n \geq 100$	31	71	65	33	93
equality constrained	all	49	51	56	40	59
	$n \geq 100$	13	16	21	12	24
general constrained	all	206	202	218	190	242
	$n \geq 100$	37	33	52	45	61

Table 1. Number of problems successfully solved by each code

Our numerical experiments indicate that the interior methods also performed efficiently on small problems, as well as in the simpler classes of unconstrained and equality constrained problems. Therefore, LOQO and KNITRO constitute effective general purpose optimization solvers.

It remains to be seen how interior methods perform on larger problems. Our set of inequality constrained problems contained 29 problems with more than 1000 variables, and only 6 problems with more than 10,000 variables. Based on this small sample, the interior methods appear to scale up well compared to the SQP codes. The observations made in the previous sections indicate, however, that a number of algorithmic and implementation features in the interior methods deserve further attention. In particular, KNITRO would benefit from a preconditioning strategy, whereas LOQO's barrier parameter update strategy could be improved.

7 Acknowledgments.

We are grateful to Sven Leyffer, Philip Gill, and Robert Vanderbei for assistance with their codes, and to David Gay for answering our questions on fine points of AMPL. We thank Hande Benson for translating the CUTE collection into AMPL, something that greatly facilitated our study.

References

1. P. Armand, J.-Ch. Gilbert, and S. Jan-Jégou. A feasible BFGS interior point algorithm for solving strongly convex minimization problems. *SIAM Journal on Optimization*, 11:199–222, 2000.
2. 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.
3. R. H. Byrd. Robust trust region methods for constrained optimization. Third SIAM Conference on Optimization, Houston, Texas, May 1987.
4. R. H. Byrd, J. Ch. Gilbert, and J. Nocedal. A trust region method based on interior point techniques for nonlinear programming. *Mathematical Programming*, 89(1):149–185, 2000.
5. R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 2000.
6. R. H. Byrd, G. Liu, and J. Nocedal. On the local behavior of an interior point method for nonlinear programming. In D. F. Griffiths and D. J. Higham, editors, *Numerical Analysis 1997*, pages 37–56. Addison Wesley Longman, 1997.
7. A. R. Conn, N. I. M. Gould, D. Orban, and Ph. L. Toint. A primal-dual trust-region algorithm for non-convex nonlinear programming. *Mathematical Programming*, 87(2):215–249, 2000.
8. A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT*: a Fortran package for Large-scale Nonlinear Optimization (Release A)*. Springer Series in Computational Mathematics. Springer Verlag, Heidelberg, Berlin, New York, 1992.
9. J. Czyzyk, M. Mesnier, and J. J. Moré. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998.
10. E. D. Dolan and J. J. Moré. Benchmarking optimization software with COPS. Mathematics and Computer Science Technical Report ANL/MCS-246, Argonne National Laboratory, Argonne, Illinois, USA, 2000.
11. A. Drud. CONOPT – a large scale GRG code. *ORSA Journal on Computing*, 6:207–216, 1994.
12. A. S. El-Bakry, R. A. Tapia, T. Tsuchiya, and Y. Zhang. On the formulation and theory of the Newton interior-point method for nonlinear programming. *Journal of Optimization Theory and Applications*, 89(3):507–541, June 1996.
13. R. Fletcher, N. I. M. Gould, S. Leyffer, and Ph. L. Toint. Global convergence of trust-region SQP-filter algorithms for nonlinear programming. Technical Report 99/03, Department of Mathematics, University of Namur, Namur, Belgium, 1999.
14. R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Numerical Analysis Report NA/171, Department of Mathematics, University of Dundee, Dundee, Scotland, 1997.

15. A. Forsgren and P. E. Gill. Primal-dual interior methods for nonconvex nonlinear programming. *SIAM Journal on Optimization*, 8(4):1132–1152, 1998.
16. R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993.
17. D. M. Gay, M. L. Overton, and M. H. Wright. A primal-dual interior method for nonconvex nonlinear programming. In Y. Yuan, editor, *Advances in Nonlinear Programming*, pages 31–56, Dordrecht, The Netherlands, 1998. Kluwer Academic Publishers.
18. P. E. Gill, W. Murray, and M. A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. Technical Report 97-2, Dept. of Mathematics, University of California, San Diego, USA, 1997.
19. J. L. Morales. A numerical study of limited memory BFGS methods, 2001. to appear in Applied Mathematics letters.
20. B. A. Murtagh and M. A. Saunders. MINOS 5.4 user's guide. Technical report, SOL 83-20R, Systems Optimization Laboratory, Stanford University, 1983. Revised 1995.
21. J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, 1999.
22. E. O. Omojokun. *Trust region algorithms for optimization with nonlinear equality and inequality constraints*. PhD thesis, University of Colorado, Boulder, Colorado, USA, 1989.
23. D. Shanno and R. Vanderbei. Interior-point methods for nonconvex nonlinear programming: Orderings and higher-order methods. Technical Report SOR-99-05, Statistics and Operations Research, Princeton University, 1999.
24. P. Spellucci. donlp2. program and users guide available as donlp2.tar from netlib/opt.
25. P. Spellucci. A new technique for inconsistent QP problems in the SQP method. To appear in Math. Meth. of OR, 47, bind 3, 1998.
26. T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983.
27. R. J. Vanderbei. AMPL models. http://www.sor.princeton.edu/ rvdb/ampl/nlmodels.
28. R. J. Vanderbei and D. F. Shanno. An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.
29. H. Yamashita, H. Yabe, and T. Tanabe. A globally and superlineraly convergent primal-dual point trust region method for large scale constrained optimization. Technical report, Mathematical Systems, Inc., Sinjuku-ku, Tokyo, Japan, 1997.