Remark on "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound Constrained Optimization"

José Luis Morales* Jorge Nocedal †

March 9, 2011

Abstract

This remark describes an improvement and a correction to Algorithm 778. It is shown that the performance of the algorithm can be improved significantly by making a relatively simple modification to the subspace minimization phase. The correction concerns an error caused by the use of routine `dpmeps` to estimate machine precision.

Keywords: nonlinear programming, constrained optimization, infeasibility

AMS Subject Classification: 90C26, 90C30, 90C55

1 Introduction

We describe two modifications to Algorithm 778 (see [6]) that give rise to significant improvements in performance. The first modification consists of a refinement of the subspace minimization phase of the algorithm. The new implementation promotes larger steps during the subspace minimization, and thereby faster identification of the optimal active set. The second modification was necessitated by the fact that the MINPACK-2 routine **dpmeps** employed in Algorithm 778 produced an incorrect estimate of machine accuracy on some compilers.

*Departamento de Matemáticas, Instituto Tecnológico Autónomo de México. This author was supported by Asociación Mexicana de Cultura AC and CONACyT-NSF grant J110.388/2006.

†Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL, USA. This author was supported by National Science Foundation grant DMS-0810213 and by Department of Energy grant DE-FG02-87ER25047-A004.

1.1 Subspace Minimization Phase

As indicated in Section 4 of [6], the algorithm performs the subspace minimization phase by minimizing a (primal) model of the objective function. Since the subspace minimizer may violate the bounds of the problem, the algorithm truncates the step towards this minimizer so as to satisfy these bounds; see [1, Section 2]. On some problems, however, this truncation strategy may result in a very short subspace minimization step that provides negligible progress toward the solution.

To avoid this drawback, we implemented a more sophisticated subspace minimization strategy. After the subspace minimizer (say, $\hat{x}$) is obtained, we compute the orthogonal projection of $\hat{x}$ onto the feasible region given by $l \leq x \leq u$; let us denote this point by $\bar{x}_{k+1}$. The search direction of the algorithm is (normally) defined as $d_k = \bar{x}_{k+1} - x_k$, and the new iterate x_{k+1} is obtained by a backtracking line search along d_k; see [1, eqn(2.5)]. An exception is when the direction d_k is not a direction of strong descent for the objective function. In this case the algorithm reverts to the strategy described in previous paragraph, i.e., $\bar{x}_{k+1}$ is computed by truncating the path from x_k to $\hat{x}$ so as to satisfy the constraints.

It is not surprising that the procedure described above improves the performance of the algorithm since it it known that it is productive to explore the boundary of the feasible region during the subspace minimization phase [5, 4]. Our technique performs a minimal exploration (it only considers the cut-back and projection points) so as to keep the cost of the subspace minimization phase to a minimum. We have observed that a more sophisticated minimization along the boundary of the feasible region does not yield significantly better performance.

1.2 Machine Accuracy

The second modification of the code is straightforward. Instead of using the routine **dpmeps** to compute machine precision, we make use of the Fortran 90 intrinsic function **EPSILON(x)**. The version of **dpmeps** used in the original version of the code underestimated machine precision by two orders of magnitude on some Linux compilers. This caused severe deterioration of performance on some problems because two important components of the algorithm rely on machine precision. One component is a safeguarding rule in the quasi-Newton updating formula, and another component controls one of the stop tests for the algorithm; see [6, eqn(3)].

1.3 Numerical Results

We have tested the new and old versions of Algorithm 778 on 38 bound constrained problems in the `CUTEr` collection [3] written in the AMPL modeling language [2]. The problems were selected to be large enough so that CPU times could be measured reliably; the number of variables ranges from 2,000 to 100,000. The old version of Algorithm 778 makes use of the correct machine precision. In order to illustrate the performance of each method in terms of function evaluations and computing time, we introduce the following two quantities:

$$\rho_f = \log_2(\mathrm{FEV}_{new}/\mathrm{FEV}_{old}) \qquad \text{and} \qquad \rho_t = \log_2(\mathrm{CPU}_{new}/\mathrm{CPU}_{old}),$$

where FEV_{new} denotes the total number of function evaluations required by the new version of the algorithm, and CPU_{new} denotes total computing time — and similarly for the old version of the algorithm. Figure 1 plots ρ_f and Figure 2 plots ρ_t. In these figures, we refer to the algorithm published in [6]) as "`L-BFGS-B`" and to the new version as "`modified L-BFGS-B`". Both versions were run with a limited memory parameter of $m = 20$. One can gauge the success of an algorithm by the area of the graphs on its side of the half-space. It is clear that the improvements in performance are rather substantial on these problems.

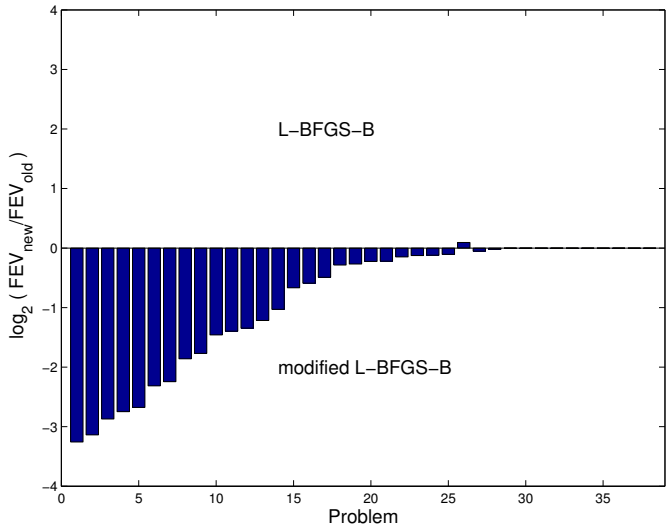

Figure 1: Comparison in terms of function evaluations

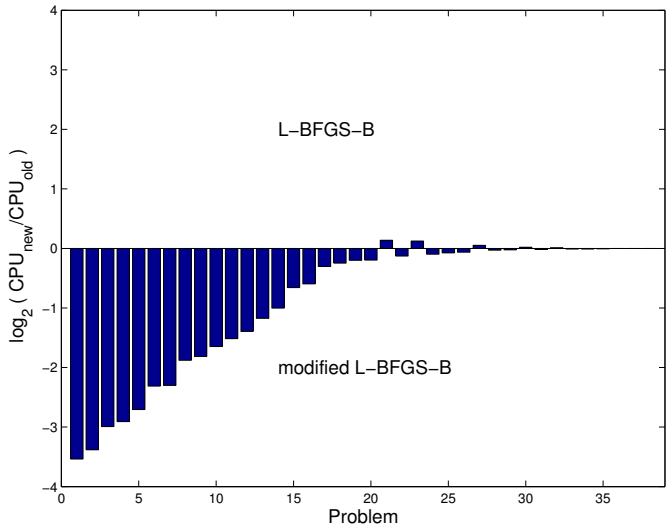

Figure 2: Comparison in terms of CPU time

References

[1] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[2] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993. `www.ampl.com`.

[3] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr and sifdec: A Constrained and Unconstrained Testing Environment, revisited. *ACM Trans. Math. Softw.*, 29(4):373–394, 2003.

[4] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Softw.*, 29(4):353–372, 2003.

[5] C. J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization*, 9(4):1100–1127, 1999.

[6] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.