PERFORMANCE PROFILES OF LINE-SEARCH ALGORITHMS FOR UNCONSTRAINED OPTIMIZATION

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania,
Academy of Romanian Scientists,
54, Splaiul Independentei, Bucharest 5, Romania.
E-mail: nandrei@ici.ro

The most important line-search algorithms for solving large-scale unconstrained optimization problems we consider in this paper are the *quasi-Newton methods*, *truncated Newton* and *conjugate gradient*. These methods proved to be efficient, robust and relatively inexpensive in term of computation. In this paper we compare the Dolan-Moré [11] performance profile of line-search algorithms implemented in the following software packages: Limited quasi-Newton LBFGS(m=3) by Nocedal [21], Liu and Nocedal [17]; Truncated Newton TN by Nash [19], Nash and Nocedal [20]; Conjugate gradient CONMIN by Shanno[22, 23], Shanno and Phua [24], CG_DESCENT by Hager and Zhang [13,14], SCALCG by Andrei [1-4] and NDHSDY by Andrei [6].

MSC2000**:** 49M07, 49M10, 90C06, 65K
*Keywords and phrases***:** unconstrained optimization, conjugate gradient, hybrid conjugate gradient, scaled conjugate gradient, conjugacy condition, numerical comparisons, Dolan- Dolan-Moré profile.

For solving large-scale unconstrained optimization problem $\min\left\{ f(x) : x \in R^n \right\}$, where $f : R^n \to R$ is a continuously differentiable function, bounded from below, two main approaches are operational to date. The first one is based on the line-search concept which assures the global convergence of the methods. The idea is to use a quadratic model in order to generate a search direction and then find a suitable stepsize along the direction. Although it is successful at most time it does not use sufficiently the quadratic model, especially for the steplength determination. The second approach is based on the trust-region concept in which the quadratic model is minimized subject to the constraint the new iterate to stay in a local neighborhood of the current iterate.

In this paper we focus on line-search algorithms for solving large-scale unconstrained optimization problems. Line search methods generate the iterates by the algorithm $x_{k+1} = x_k + \alpha_k d_k$, where d_k is a search direction and $\alpha_k > 0$ is chosen so that $f(x_{k+1}) < f(x_k)$. The most important line-search algorithms for solving large-scale problems we consider in this paper, are the *quasi-Newton methods*, *truncated Newton* and *conjugate gradient*. These methods proved to be efficient, robust and relatively inexpensive in term of computation. The quasi-Newton and truncated Newton methods are versions of basic Newton method.

Quasi-Newton methods gradually build up an approximate Hessian matrix (or an approximate inverse Hessian matrix) by using the gradient information from some of the previous iterates. Given the current iterate x_k and the approximate Hessian matrix B_k at x_k, the so called the Newton system $B_k d_k = -\nabla f(x_k)$ is solved in order to generate the direction d_k. The best known quasi-Newton method is BFGS. However, the BFGS approach is not affordable due to the memory requirements. The limited BFGS variant introduced by Nocedal [21] overcomes this difficulty by approximating the product $d_k = -H_k \nabla f(x_k)$, where H_k is a positive definite approximation to the inverse of the Hessian at x_k, in terms of the most recently

computed m pairs $\{s_i, y_i\}$, where $s_i = x_{i+1} - x_i$ and $y_i = \nabla f(x_{i+1}) - \nabla f(x_i)$. When the $m+1$ pair is computed, the oldest pair is discarded and its location in the memory is replaced by the new one.
On the other hand, the truncated Newton method stops the solving of the Newton system as soon as a suitable termination criterion is satisfied. In these methods the direction d_k satisfies the condition $\left\|\nabla^2 f(x_k)d_k + \nabla f(x_k)\right\| \le \eta_k \left\|\nabla f(x_k)\right\|$, for some $\eta_k \in (0,1)$, known as the "forcing" sequence. Dembo, Eisenstat and Steihaug [10] choose the forcing terms as:

$$\eta_k = \min\left\{\frac{1}{2}, c\left\|\nabla f(x_k\right\|^r\right\},$$

where c is a positive constant and $0 < r \le 1.$ Other formula for η_k selection is given in [12].
The conjugate gradient algorithms are another class of algorithms for large-scale unconstrained optimization in which the search direction is computed as $d_{k+1} = -g_{k+1} + \beta_k d_k$. The parameter β_k defines the algorithm, and plenty of conjugate gradient methods are known. Mainly, these algorithms are motivated by the success of linear conjugate gradient method in minimizing quadratic functions with positive definite Hessians. In these methods the direction is a combination of the negative gradient with another direction selected in such a manner that the searching direction is descendent. An excellent survey of conjugate gradient methods, with a special attention on their global convergence, is given by Hager and Zhang [15]. The conjugate gradient methods can be classified as: classical conjugate gradient algorithms, hybrid, scaled, modified and parametric. A description of these algorithms and their performance profiles are presented in [7].
In this paper we compare the Dolan-Moré [11] performance profile of line-search algorithms implemented in the following software packages, as in Table 1.

Table 1. Line-search methods and software package.

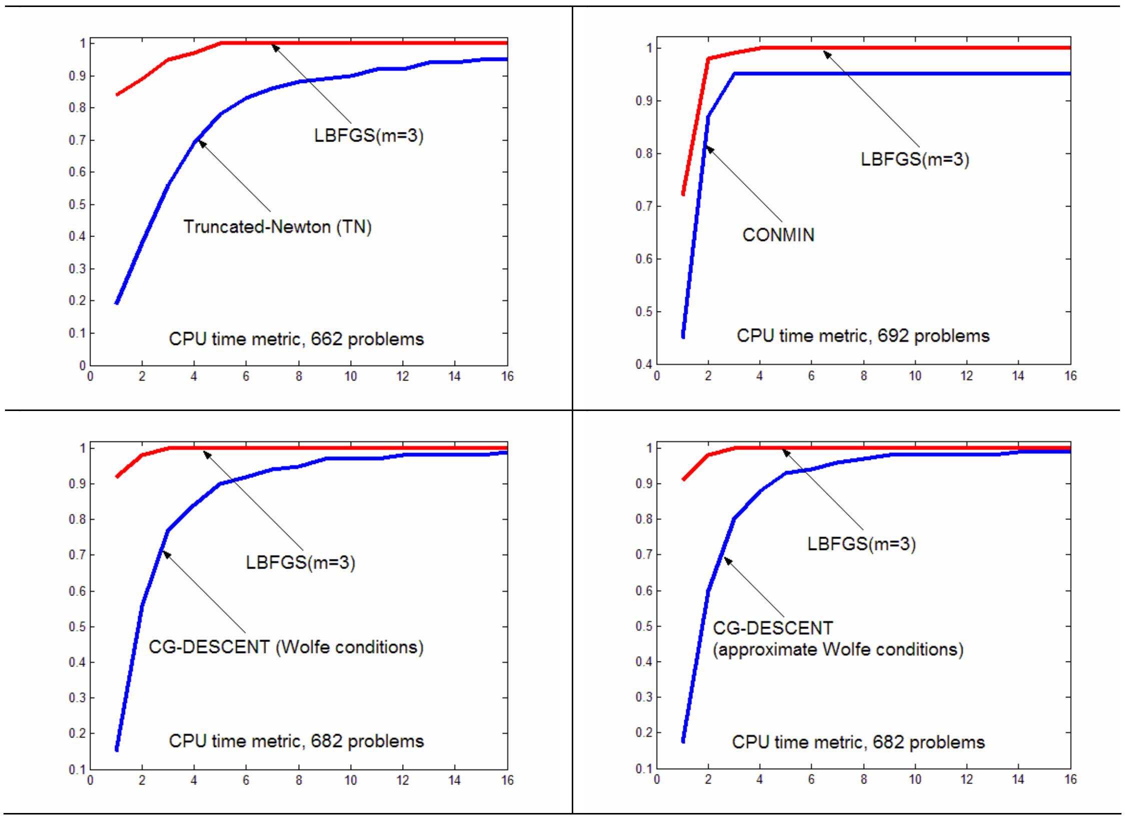
Method	Package	Author(s)
Limited quasi-Newton	LBFGS(m=3)	Nocedal [21] Liu and Nocedal [17]
Truncated Newton	TN	Nash [19] Nash and Nocedal [20]
Conjugate gradient	CONMIN	Shanno[22, 23], Shanno and Phua [24]
	CG_DESCENT	Hager and Zhang [13,14]
	SCALCG	Andrei [1-4]
	NDHSDY	Andrei [6]

Concerning the conjugate gradient algorithms we selected only the most successful variants CONMIN, CG_DESCENT, SCALCG and NDHSDY. CONMIN is a preconditioned memoryless BFGS conjugate gradient algorithm. CG_DESCENT is a modification of Hestenes and Stiefel [16] method, SCALCG is a scaled conjugate gradient algorithm and NDHSDY is a hybrid conjugate gradient algorithm as a convex combination of Hestenes-Stiefel and Dai-Yuan [9] conjugate gradient algorithms where the parameter in convex combination is computed using the Newton direction [6].
To see the performance profiles of these algorithms we selected 75 large-scale unconstrained optimization problems in extended or generalized form, most of them from CUTE library [8], along with other large-scale optimization problems presented in [5]. Each problem is tested 10 times for a gradually increasing number of variables: $n = 1000, 2000, \ldots, 10000.$ Therefore we have a set of 750 unconstrained optimization test problems.
All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and the same stopping criterion $\left\|g_k\right\|_\infty \le 10^{-6}$, where $\left\|.\right\|_\infty$ is the maximum absolute

component of a vector. In LBFGS the step length is computed by means of the line search routine MCVSRCH, which is a slight modification of the routine CSRCH written by Moré and Thuente [18]. The line search in TN package is a variant of Gill and Murray. CONMIN, SCALCG and NDHSDY implement the Wolfe line search conditions using the cubic interpolation. In CG_DESCENT the Wolfe line search is implemented in two manners: the *classical Wolfe line search* and the *approximate Wolfe line search* which is based on a very fine interpretation of the numerical issue concerning the first Wolfe condition. The comparisons of algorithms are given in the following context. Let f_i^{ALG1} and f_i^{ALG2} be the optimal value found by ALG1 and ALG2, for problem $i = 1,\dots,750,$ respectively. We say that, in the particular problem $i,$ the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \qquad (31)$$

and the number of iterations, or the number of function-gradient evaluations, or the CPU time of ALG1 was less than the number of iterations, or the number of function-gradient evaluations, or the CPU time corresponding to ALG2, respectively. In this numerical study we declare that a method solved a particular problem if the final point obtained has the lowest functional value among the tested methods (up to 10^{-3} tolerance as it is specified in (31)). Clearly, this criterion is acceptable for users that are interested in minimizing functions and not finding critical points. All codes are written in double precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8GHz workstation.
In the first set of numerical experiments we compare LBFGS(m=3) to TN, CONMIN, CG_DESCENT, SCALCG and NDHSDY, respectively. Figure 1 shows the performance profiles of these algorithms. For each algorithm, we plot the fraction of problems for which the algorithm is within a factor of the best CPU time. The left side of these Figures gives the percentage of the test problems, out of 750, for which an algorithm is more successful; the right side gives the percentage of the test problems that were successfully solved be each of the algorithms. Mainly, the right side represents a measure of an algorithm's robustness.



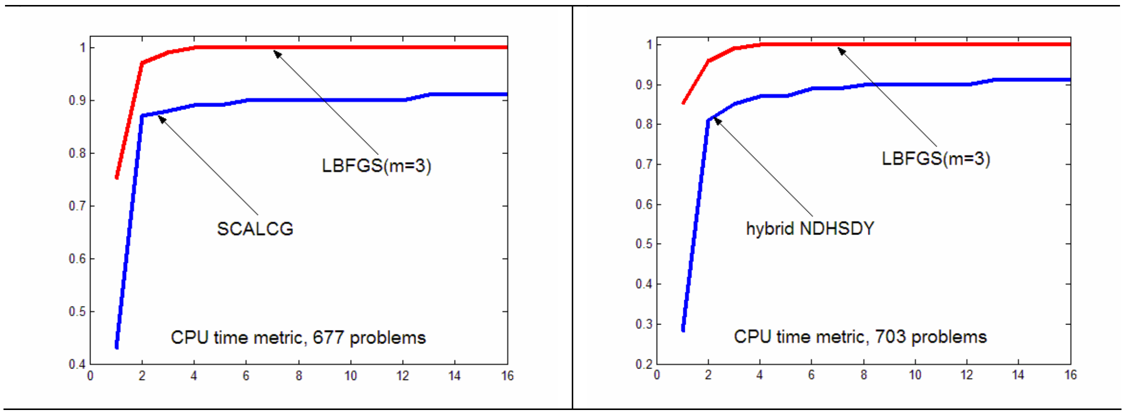


Fig.1. Performance profiles of LBFGS(m=3) versus TN, CONMIN, CG_DESCENT, SCALCG and NDHSDY.

Observe that, the limited memory BFGS method is top performer in this class of algorithms. Close to LBFGS is CONMIN, but concerning the robustness, CG_DESCENT tends to the same robustness as LBFGS. Figure 2 shows the cumulative performance profiles of these algorithms.

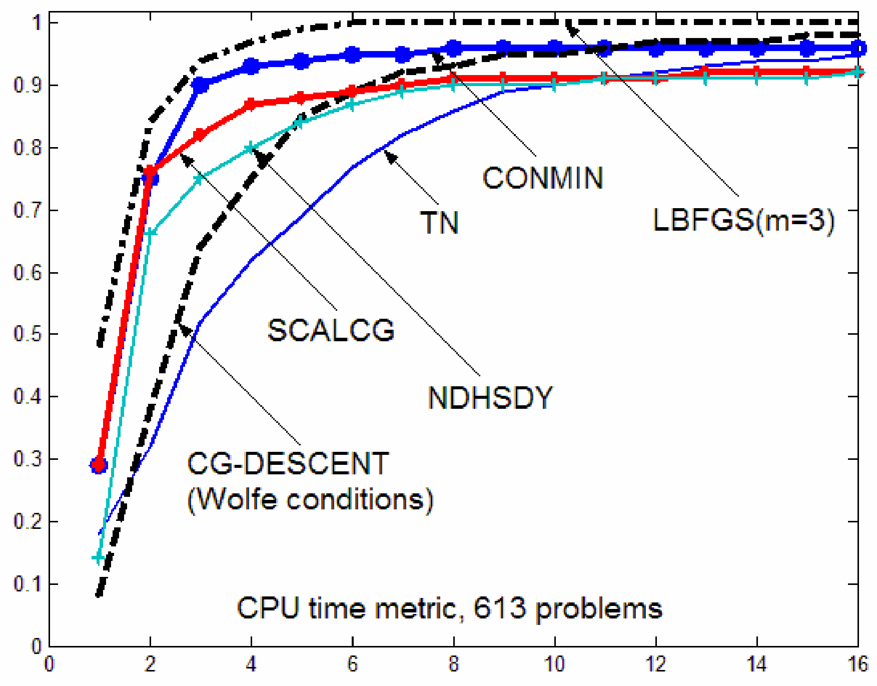


Fig. 2. Cumulative performance profiles.

In the second set of numerical experiments we compare CONMIN to TN, CG_DESCENT, SCALCG and NDHSDY. Figure 3 presents the performance profiles of these algorithms.

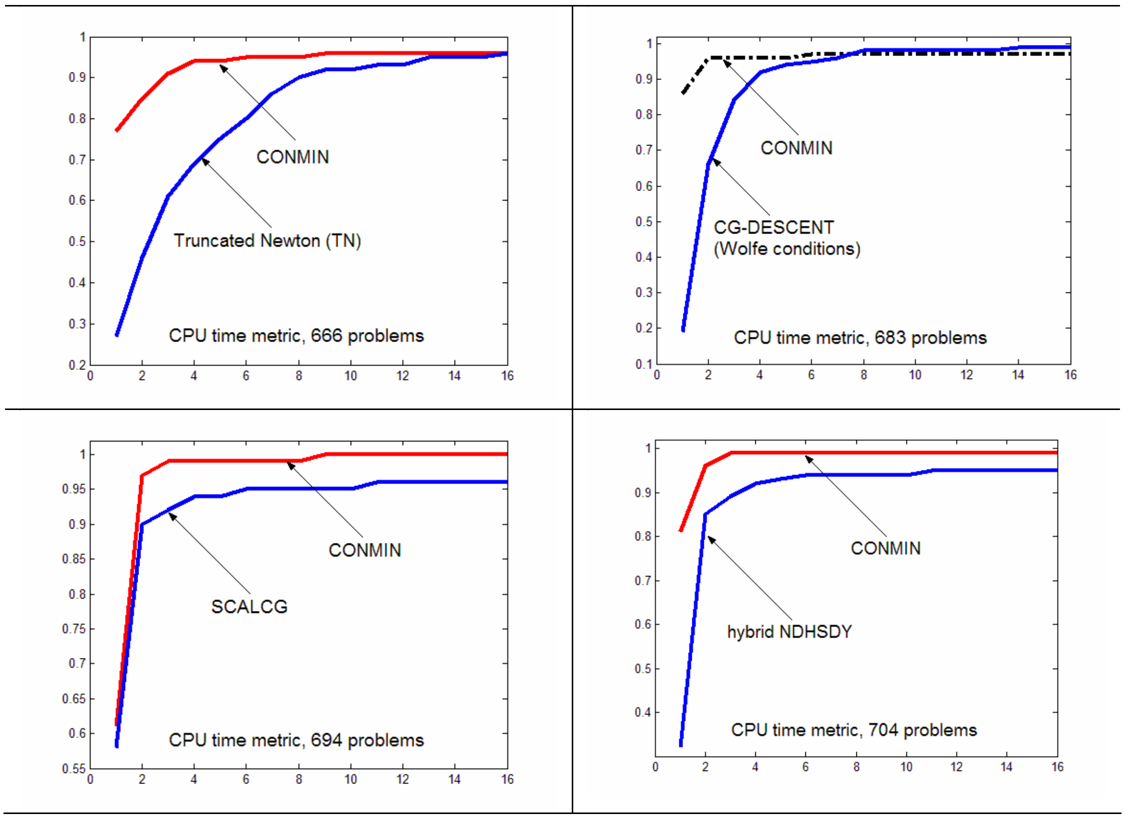


Fig. 3. Performance profiles of CONMIN versus TN, CG_DESCENT, SCALCG and NDHSDY.

The figures indicate that relative to CPU time metric, under the stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, LBFGS(m=3) is fastest, followed by CONMIN and followed by SCALCG. LBFGS appears to generate the best search direction, on average.

References

[1] ANDREI, N., *Scaled memoryless BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Optimization Methods and Software, 22 (2007), pp.561-571.
[2] ANDREI, N., *A scaled BFGS preconditioned conjugate gradient algorithm for unconstrained optimization.* Applied Mathematics Letters, 20 (2007), pp.645-650.
[3] ANDREI, N., *Scaled conjugate gradient algorithms for unconstrained optimization.* Computational Optimization and Applications, accepted.
[4] ANDREI, N., *A scaled nonlinear conjugate gradient algorithm for unconstrained optimization.* Optimization. A journal of mathematical programming and operations research, accepted.
[5] ANDREI, N., *Test functions for unconstrained optimization.* http:// www.ici.ro/ camo/ neculai /SCALCG /evalfg.for
[6] ANDREI, N., *Another hybrid conjugate gradient algorithm for unconstrained optimization.* ICI Technical Report, July 18, 2007.
[7] ANDREI, N., *Performance profiles of conjugate gradient algorithms for unconstrained optimization.* This issue.
[8] BONGARTZ, I., CONN, A.R., GOULD, N.I.M., TOINT, P.L., *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Software, 21, (1995) 123-160.
[9] DAI, Y.H., YUAN, Y., *A nonlinear conjugate gradient method with a strong global convergence property,* SIAM J. Optim., 10 (1999) pp.177-182.
[10] DEMBO, R.S., EISENSTAT, S.C., STEIHAUG, T., *Inexact Newton methods.* SIAM J. Num. Anal. 19 (1982), pp.400-408.
[11] DOLAN, E.D., MORÉ, J.J., *Benchmarcking optimization software with performance profiles.* Mathematical Programming, vol. 91, pp.201-213, 2002.
[12] EISENSTAT, S.C., WALKER, H.F., *Choosing the forcing terms in an inexact Newton method.* SIAM J. Sci. Comp. 17 (1996) pp. 16-32.

[13] HAGER, W.W., ZHANG, H., *A new conjugate gradient method with guaranteed descent and an efficient line search*. SIAM Journal on Optimization, 16 (2005) 170-192.

[14] HAGER, W.W., ZHANG, H., *Algorithm 851: CG_DESCENT, A conjugate gradient method with guaranteed descent.* ACM Transactions on Mathematical Software, 32 (2006), 113-137.

[15] HAGER, W.W., ZHANG, H., *A survey of nonlinear conjugate gradient methods,* Pacific Journal of Optimization, 2 (2006), pp.35-58.

[16] HESTENES, M.R., STIEFEL, E., *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards Sec. B. 48 (1952) 409-436.

[17] LIU, D., NOCEDAL, J., *On the limited memory BFGS method for large scale optimization*, Mathematical Programming B 45 (1989) 503-528.

[18] MORÉ, J.J., THUENTE, D.J., *Line search algorithms with guaranteed sufficient decrease*. ACM Transactions on Mathematical Software, 20, 1994, pp.286-307.

[19] NASH, S.G., *Preconditioning of truncated-Newton methods.* SIAM J. Sci. Comp. 6 (1985) pp. 599-616.

[20] NASH S.G. , NOCEDAL, J., *A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization.* SIAM J. Opt. 1 (1991) pp.358-372.

[21] NOCEDAL, J., *Updating quasi-Newton matrices with limited starage.* Mathematics of Computation, 35 (1980), pp.773-782.

[22] SHANNO, D.F., *Conjugate gradient methods with inexact searches*. Mathematics of Operations Research, vol. 3 (1978), pp.244-256.

[23] SHANNO, D.F., *On the convergence of a new conjugate gradient algorithm.* SIAM J. Numer. Anal., 15 (1978), pp.1247-1257.

[24] SHANNO, D.F., PHUA, K.H., *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Software, 2 (1976) 87-94.

March 5, 2008