NUMERICAL COMPARISON OF CONJUGATE GRADIENT ALGORITHMS FOR UNCONSTRAINED OPTIMIZATION

Neculai Andrei
Research Institute for Informatics,
Center for Advanced Modeling and Optimization,
8-10, Averescu Avenue, Bucharest 1, Romania,
E-mail: nandrei@ici.ro

Abstract. Conjugate gradient methods represent an important class of unconstrained optimization algorithms with strong local and global convergence properties and modest memory requirements. This family of algorithms includes a lot of variants, well known in the literature, with important convergence properties and numerical efficiency. The purpose of this paper is to present these algorithms as well as their Dolan and Moré's performances to solve a large variety of large-scale unconstrained optimization problems. Some comparisons with well established limited memory quasi-Newton and truncated Newton methods are also presented.

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

1. Introduction. Conjugate gradient methods represent an important class of unconstrained optimization algorithms with strong local and global convergence properties and modest memory requirements. An excellent survey of development of different versions of nonlinear conjugate gradient methods, with special attention to global convergence properties, is presented by Hager and Zhang [23]. This family of algorithms includes a lot of variants, well known in the literature, with important convergence properties and numerical efficiency. The purpose of this paper is to present these algorithms as well as their performances to solve a large variety of large-scale unconstrained optimization problems.
For solving the nonlinear unconstrained optimization problem

$$\min\{f(x): x \in R^n\}, \tag{1}$$

where $f: R^n \to R$ is a continuously differentiable function bounded from below, starting from an initial guess $x_0 \in R^n$ a nonlinear conjugate gradient method, generates a sequence $\{x_k\}$ as

$$x_{k+1} = x_k + \alpha_k d_k, \tag{2}$$

where $\alpha_k > 0$ is obtained by line search, and the directions d_k are generated as

$$d_{k+1} = -g_{k+1} + \beta_k s_k, \quad d_0 = -g_0. \tag{3}$$

In (3) β_k is known as the conjugate gradient parameter, $s_k = x_{k+1} - x_k$ and $g_k = \nabla f(x_k)$. Consider $\|.\|$ the Euclidean norm and define $y_k = g_{k+1} - g_k$. The line search in the conjugate gradient algorithms often is based on the standard Wolfe conditions:

$$f(x_k + \alpha_k d_k) - f(x_k) \le \rho \alpha_k g_k^T d_k, \tag{4}$$

$$g_{k+1}^T d_k \ge \sigma g_k^T d_k, \tag{5}$$

where d_k is a descent direction and $0 < \rho \le \sigma < 1$. For some conjugate gradient algorithms, stronger versions of the Wolfe conditions are needed to ensure convergence and to enhance

stability. According to formula for β_k computation, the conjugate gradient algorithms can be classified as: classical, hybrid, scaled, modified and parametric. In the following we shall present these algorithms and insist on their numerical Dolan and Moré's performances profiles for solving large-scale unconstrained optimization problems.
The history of conjugate gradient method begins with the seminal paper of Hestenes and Stiefel [24] who presented an algorithm for solving symmetric, positive definite linear algebraic systems. In 1964 Fletcher and Reeves [19] extended the domain of application of conjugate gradient method to nonlinear problems, thus starting the nonlinear conjugate gradient research direction. The main advantages of the conjugate gradient method are its low memory requirements, and its convergence speed. A large variety of nonlinear conjugate gradient algorithms are known. For each of them convergence results have been proved in mild conditions which refer to the Lipschitz and boundedness assumptions. To prove the global convergence of nonlinear conjugate gradient methods, often the Zoutendijk condition is used combined with analysis showing that the sufficient descent condition $g_k^T d_k \le -c\|g_k\|^2$ holds, and that there exists a constant δ such that $\|d_k\|^2 \le \delta k$. Often, the convergence analysis of conjugate gradient algorithms, for general nonlinear functions, follows insights developed by Gilbert and Nocedal [20]. The idea is to bound the change $u_{k+1} - u_k$ in the normalized direction $u_k = d_k / \|d_k\|$, which is used to conclude, by contradiction, that the gradients cannot be bounded away from zero.

2. Classical conjugate gradient algorithms. These algorithms are defined by (2) and (3), where the parameter β_k is computed as in Table 1. Observe that these algorithms can be classified as algorithms with $\|g_{k+1}\|^2$ in the numerator of β_k and algorithms with $g_{k+1}^T y_k$ in the numerator of parameter β_k.

Table 1. Classical conjugate gradient algorithms.

Nr.	Formula	Author(s)
1.	$\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$	Hestenes and Stiefel [24] (HS) The first conjugate gradient algorithm for linear algebraic systems.
2.	$\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	Fletcher and Reeves [19] (FR) The first conjugate gradient algorithm for nonlinear functions.
3.	$\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$	Polak-Ribiere [33] and Polyak [34] (PRP)
4.	$\beta_k^{PRP+} = max\left\{0, \frac{y_k^T g_{k+1}}{g_k^T g_k}\right\}$	Polak-Ribiere and Polyak + (PRP+) suggested by Powell [35]
5.	$\beta_k^{CD} = -\frac{g_{k+1}^T g_{k+1}}{g_k^T d_k}$	Conjugate Descent (CD) introduced by Fletcher [18]
6.	$\beta_k^{LS} = -\frac{y_k^T g_{k+1}}{g_k^T d_k}$	Liu and Storey [27] (LS)
7.	$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$	Dai and Yuan [13] (DY)

The FR, CD and DY with $\|g_{k+1}\|^2$ in the numerator of β_k have strong convergence theory, but all these methods are susceptible to jamming. They begin to take small steps without

making any significant progress to the minimum. On the other hand, HS, PRP and LS methods with $g_{k+1}^T y_k$ in the numerator of parameter β_k, have a built-in restart feature that addresses the jamming phenomenon. When the step s_k is small, the factor $y_k = g_{k+1} - g_k$ in the numerator of β_k tends to zero. Therefore, β_k becomes small and the new direction d_{k+1} in (3) is essentially the steepest descent direction $-g_{k+1}$. With other words, HS, PRP and LS methods automatically adjust β_k to avoid jamming, and their performances are better than the performance of methods with $\|g_{k+1}\|^2$ in the numerator of β_k.

3. Hybrid conjugate gradient methods. These algorithms have been devised to exploit the attractive features of the classical conjugate gradient algorithms. They are defined by (2) and (3) where the parameter β_k is as in Table 2. There are two classes of hybrid algorithms. The first class of the hybrid algorithms combines in a *projective* manner the algorithms having $\|g_{k+1}\|^2$ in the numerator of β_k with the algorithms having $g_{k+1}^T y_k$ in the numerator of parameter β_k. The second class of hybrid algorithms, more recent established, considers *convex combinations* of algorithms with $\|g_{k+1}\|^2$ in the numerator of β_k and the algorithms having $g_{k+1}^T y_k$ in the numerator of parameter β_k. In general, the performances of hybrid conjugate gradient algorithms are higher than the performances of classical conjugate gradient algorithms.

Table 2. Hybrid conjugate gradient algorithms.

Nr.	Formula	Author(s)
1.	$\beta_k^{hDY} = max\left\{c\beta_k^{DY}, min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}$	Hybrid Dai-Yuan [15](hDY)
2.	$\beta_k^{hDYz} = max\left\{0, min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}$	Hybrid Dai-Yuan zero [15] (hDYz)
3.	$\beta_k^{GN} = max\left\{-\beta_k^{FR}, min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Gilbert and Nocedal [20] (GN)
4.	$\beta_k^{HuS} = max\left\{0, min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Hu and Storey [25] (HuS)
5.	$\beta_k^{TaS} = \begin{cases} \beta_k^{PRP} & 0 \le \beta_k^{PRP} \le \beta_k^{FR}, \\ \beta_k^{FR} & \text{otherwise} \end{cases}$	Touati-Ahmed and Storey [39] (TaS)
6.	$\beta_k^{LS-CD} = max\left\{0, min\left\{\beta_k^{LS}, \beta_k^{CD}\right\}\right\}$	Hybrid Liu-Storey, Conjugate-Descent (LS-CD)
7.	$\beta_k^{CCOMB} = (1-\theta_k)\frac{g_{k+1}^T y_k}{g_k^T g_k} + \theta_k \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$, $\theta_k = \frac{(y_k^T g_{k+1})(y_k^T s_k) - (y_k^T g_{k+1})(g_k^T g_k)}{(y_k^T g_{k+1})(y_k^T s_k) - \|g_{k+1}\|^2 \|g_k\|^2}$. If $\theta_k \le 0$, then set $\theta_k = 0$, i.e. $\beta_k^{CCOMB} = \beta_k^{PRP}$; if $\theta_k \ge 1$, then take $\theta_k = 1$, i.e. $\beta_k^{CCOMB} = \beta_k^{DY}$.	Convex combination of PRP and DY where θ_k is obtained by conjugacy condition. Andrei [7] (CCOMB)
8.	$\beta_k^{NDOMB} = (1-\theta_k)\frac{g_{k+1}^T y_k}{g_k^T g_k} + \theta_k \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$,	Convex combination of PRP and DY where θ_k is obtained using the Newton direction. Andrei [7] (NDOMB)

	$\theta_k = \dfrac{(y_k^T g_{k+1} - s_k^T g_{k+1})\|g_k\|^2 - (g_{k+1}^T y_k)(y_k^T s_k)}{\|g_{k+1}\|^2 \|g_k\|^2 - (g_{k+1}^T y_k)(y_k^T s_k)}$. If $\theta_k \le 0$, then set $\theta_k = 0$, i.e. $\beta_k^{NDOMB} = \beta_k^{PRP}$; if $\theta_k \ge 1$, then take $\theta_k = 1$, i.e. $\beta_k^{NDOMB} = \beta_k^{DY}$.	
9.	$\beta_k^{NDHSDY} = (1-\theta_k)\dfrac{g_{k+1}^T y_k}{y_k^T s_k} + \theta_k \dfrac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$, $\theta_k = -\dfrac{s_k^T g_{k+1}}{g_k^T g_{k+1}}$. If $\theta_k \le 0$, then set $\theta_k = 0$, i.e. $\beta_k^{NDHSDY} = \beta_k^{HS}$; if $\theta_k \ge 1$, then take $\theta_k = 1$, i.e. $\beta_k^{NDHSDY} = \beta_k^{DY}$.	Convex combination of HS and DY, where θ_k is obtained using the Newton direction. Andrei [8] (NDHSDY)

4. Scaled conjugate gradient algorithms. The algorithms in this class generates a sequence x_k of approximations to the minimum x^* of f, in which

$$x_{k+1} = x_k + \alpha_k d_k, \tag{6}$$

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k, \tag{7}$$

where θ_{k+1} is a parameter. The iterative process is initialized with an initial point x_0 and $d_0 = -g_0$. Observe that if $\theta_{k+1} = 1$, then we get the classical conjugate gradient algorithms according to the value of the scalar parameter β_k. On the other hand, if $\beta_k = 0$, then we get another class of algorithms according to the selection of the parameter θ_{k+1}. Considering $\beta_k = 0$, there are two possibilities for θ_{k+1}: a positive scalar or a positive definite matrix. If $\theta_{k+1} = 1$, then we have the steepest descent algorithm. If $\theta_{k+1} = \nabla^2 f(x_{k+1})^{-1}$, or an approximation of it, then we get the Newton or the quasi-Newton algorithms, respectively. Therefore, we see that in the general case, when $\theta_{k+1} \neq 0$ is selected in a quasi-Newton manner, and $\beta_k \neq 0$, (7) represents a combination between the quasi-Newton and the conjugate gradient methods. However, if θ_{k+1} is a matrix containing some useful information about the inverse Hessian of function f, we are better off using $d_{k+1} = -\theta_{k+1} g_{k+1}$ since the addition of the term $\beta_k s_k$ in (7) may prevent the direction d_k from being a descent direction unless the line search is sufficiently accurate. Therefore, in the following we shall consider θ_{k+1} as a positive scalar which contains some useful information to the inverse Hessian of function f.

To determine β_k consider the following procedure [1-4]. As we know, the Newton direction for solving (1) is given by $d_{k+1} = -\nabla^2 f(x_{k+1})^{-1} g_{k+1}$. Therefore, from the equality

$$-\nabla^2 f(x_{k+1})^{-1} g_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k s_k,$$

we get:

$$\beta_k = \frac{s_k^T \nabla^2 f(x_{k+1}) \theta_{k+1} g_{k+1} - s_k^T g_{k+1}}{s_k^T \nabla^2 f(x_{k+1}) s_k}. \tag{8}$$

Using the Taylor development, after some algebra we get:

$$\beta_k = \frac{(\theta_{k+1} y_k - s_k)^T g_{k+1}}{y_k^T s_k}, \tag{9}$$

where $y_k = g_{k+1} - g_k$. Birgin and Martínez [10], who firstly introduced scaled conjugate gradient algorithms, arrived at the same formula for β_k, but using a geometric interpretation

of quadratic function minimization. The parameter β_k in (7) can be defined, as in Table 3, where the scaling parameter θ_k is computed as:

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k}. \tag{10}$$

Table 3. Scaled conjugate gradient algorithms.

Nr.	Formula	Author(s)
1.	$\beta_k^{BM} = \frac{g_{k+1}^T(\theta_k y_k - s_k)}{y_k^T s_k}$	Scaled Perry. Suggested by Birgin and Martínez [10] and Andrei [1-4] (BM)
2.	$\beta_k^{BM+} = max\left\{0, \frac{\theta_k g_{k+1}^T y_k}{y_k^T s_k}\right\} - \frac{g_{k+1}^T s_k}{y_k^T s_k}$	Scaled Perry+. Suggested by Birgin and Martínez [10] (BM+)
3.	$\beta_k^{sPRP} = \frac{\theta_k g_{k+1}^T y_k}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Polak-Ribière-Polyak. Suggested by Birgin and Martínez [10] and Andrei [1-4] (sPRP)
4.	$\beta_k^{sFR} = \frac{\theta_k g_{k+1}^T g_{k+1}}{\alpha_k \theta_{k-1} g_k^T g_k}$	Scaled Fletcher-Reeves. Suggested by Birgin and Martínez [10] and Andrei [1-4] (sFR)
5.	$\beta_k^{sHS} = \theta_{k+1} \frac{g_{k+1}^T y_k}{y_k^T s_k}$	Scaled Hestenes-Steifel (sHS)[1]

Another scaled conjugate gradient algorithm has been presented by Andrei [1-4]. This is a scaled memoryless BFGS preconditioned conjugate gradient algorithm. The basic idea is to combine the scaled memoryless BFGS method and the preconditioning technique in the frame of conjugate gradient method. The preconditioner, which is also a scaled memoryless BFGS matrix, is reset when the Powell restart criterion holds. The parameter scaling the gradient is selected as the spectral gradient (10).

Algorithm SCALCG [1-4]

Step 1. Initialization. Select $x_0 \in R^n$, and the parameters $0 < \sigma_1 \le \sigma_2 < 1$. Compute $f(x_0)$ and $g_0 = \nabla f(x_0)$. Set $d_0 = -g_0$ and $\alpha_0 = 1/\|g_0\|$. Set $k = 0$.

Step 2. Line search. Compute α_k satisfying the Wolfe conditions (4) and (5). Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1}), g_{k+1}$ and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 3. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 4. Scaling factor computation. Compute θ_k using (10).

Step 5. Restart direction. Compute the (restart) direction d_k as:

$$d_{k+1} = -\theta_{k+1} g_{k+1} + \theta_{k+1}\left(\frac{g_{k+1}^T s_k}{y_k^T s_k}\right) y_k - \left[\left(1 + \theta_{k+1}\frac{y_k^T y_k}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} - \theta_{k+1}\frac{g_{k+1}^T y_k}{y_k^T s_k}\right] s_k.$$

Step 6. Line search. Compute the initial guess: $\alpha_k = \alpha_{k-1}\|d_{k-1}\|_2 / \|d_k\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Step 7. Store: $\theta = \theta_k$, $s = s_k$ and $y = y_k$.

Step 8. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$.

Step 9. Restart. If the Powell restart criterion: $\left|g_{k+1}^T g_k\right| \geq 0.2\left\|g_{k+1}\right\|^2$, is satisfied, then go to step 4 (a restart step); otherwise continue with step 10 (a standard step).
Step 10. Standard direction. Compute the direction d_k as:

$$d_{k+1} = -v + \frac{(g_{k+1}^T s_k)w + (g_{k+1}^T w)s_k}{y_k^T s_k} - \left(1 + \frac{y_k^T w}{y_k^T s_k}\right)\frac{g_{k+1}^T s_k}{y_k^T s_k} s_k,$$

where v and w are computed as:

$$v = \theta g_{k+1} - \theta\left(\frac{g_{k+1}^T s}{y^T s}\right) y + \left[\left(1 + \theta\frac{y^T y}{y^T s}\right)\frac{g_{k+1}^T s}{y^T s} - \theta\frac{g_{k+1}^T y}{y^T s}\right] s,$$

and

$$w = \theta y_k - \theta\left(\frac{y_k^T s}{y^T s}\right) y + \left[\left(1 + \theta\frac{y^T y}{y^T s}\right)\frac{y_k^T s}{y^T s} - \theta\frac{y_k^T y}{y^T s}\right] s,$$

with saved values θ, s and y.

Step 11. Line search. Compute the initial guess: $\alpha_k = \alpha_{k-1}\left\|d_{k-1}\right\|_2 / \left\|d_k\right\|_2$. Using this initialization compute α_k satisfying the Wolfe conditions. Update the variables $x_{k+1} = x_k + \alpha_k d_k$. Compute $f(x_{k+1})$, g_{k+1} and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.
Step 12. Test for continuation of iterations. If this test is satisfied the iterations are stopped, else set $k = k + 1$ and go to step 9. ■

To a great extent, SCALCG algorithm is very close to the Perry/Shanno computational scheme [32,36,37]. SCALCG is a scaled memoryless BFGS preconditioned algorithm where the scaling factor is the inverse of a scalar approximation of the Hessian. If the Powell restart criterion $\left|g_{k+1}^T g_k\right| \geq 0.2\left\|g_{k+1}\right\|^2$ is used, for general functions f bounded from below with bounded second partial derivatives and bounded level set, using the same arguments considered by Shanno in [37] it is possible to prove that the iterates either converge to a point x^* satisfying $\left\|g(x^*)\right\| = 0$, or the iterates cycle.

5. Modified conjugate gradient algorithms. We know a large variety of modified conjugate gradient algorithms. All of them are designed to improve the performances of the classical computational schemes using the idea of preconditioning or the modification of classical schemes in order to satisfy the sufficient descent condition. The algorithms in this class are characterized by (2) and (3), where the parameter β_k is computed as in Table 4.

Table 4. Modified conjugate gradient algorithms.

Nr.	Formula	Author(s)
1.	$d_{k+1} = -g_{k+1} + \bar{\beta}_k^N d_k$, $d_0 = -g_0$, $\bar{\beta}_k^N = max\left\{\beta_k^N, \eta_k\right\}$, $\eta_k = \frac{-1}{\left\|d_k\right\| min\left\{\eta, \left\|g_k\right\|\right\}}$, $\eta = 0.01$ $\beta_k^N = \frac{1}{y_k^T d_k}\left(y_k - 2\frac{\left\|y_k\right\|^2}{y_k^T d_k} d_k\right)^T g_{k+1}$,	Introduced by Hager şi Zhang [21, 22]. (CG_DESCENT) This scheme is obtained by deleting a term from the search direction for the memoryless quasi-Newton scheme of Perry [32] and Shanno [36, 37]. In [9] Andrei proved that this computational scheme is also a modification of HS formula.

2.	$\beta_k^{ACGA} = \frac{1}{y_k^T s_k}\left(y_k - \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k\right)^T g_{k+1}$	Suggested by Andrei [9] (ACGA)
3.	$\beta_k^{ACGA+} = \max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\}\left(1 - \frac{s_k^T g_{k+1}}{y_k^T s_k}\right)$	Suggested by Andrei [9] (ACGA+)
4.	$d_{k+1} = -\theta_{k+1} g_{k+1} + \beta_k^{CGSD} d_k$, $d_0 = -g_0$, $\beta_k^{CGSD} = \frac{1}{y_k^T s_k}\left(g_{k+1} - \frac{g_{k+1}^T y_k}{y_k^T s_k} s_k\right)^T g_{k+1}$ $\theta_{k+1} = \frac{\|g_{k+1}\|^2}{y_k^T g_{k+1}}$	Introduced by Andrei [5] (CGSD) as a modification of DY method.
5.	$\beta_k^{APRP} = \frac{1}{y_k^T s_k}\left(y_k - \frac{\|y_k\|^2}{\|g_k\|^2} s_k\right)^T g_{k+1}$	Suggested by Andrei [9] (APRP) This is a modification of PRP method.

Maximization in formula for $\overline{\beta}_k^N$ computation scheme by Hager and Zhang plays the role of the truncation operation like in the PRP+ scheme, for example. Hager and Zhang obtained this algorithm by deleting a term from the search direction for the memoryless quasi-Newton scheme of Perry [32] and Shanno [36]. In [21] Hager and Zhang proved the global convergence with inexact line search showing that for any line search and any function, the sufficient descent condition $g_k^T d_k \le -(7/8)\|g_k\|^2$ is satisfied and the jamming is avoided essentially due to the $y_k^T g_{k+1}$ term in the formula for β_k^N.

The ACGA and ACGA+ computational schemes are a modification of the DY conjugate gradient algorithm, designed to satisfy the sufficient descent condition. In [9] Andrei proved that for uniformly convex functions under strong Wolfe condition the ACGA is globally convergent. The CGSD algorithm is also a modification of Dai and Yuan conjugate gradient algorithm. In [9] Andrei proved the global convergence of CGSD for general nonlinear functions under the Wolfe conditions.

One of the best conjugate gradient algorithm in this class is CONMIN by Shanno [36] and Shanno and Phua [38]. Using the Hestenes and Stiefel formula for updating β_k, Perry [32] suggested a formula for computing the search direction d_{k+1} which satisfy a system of linear equations, similar but not identical, to the quasi-Newton equation. Shanno [36] reconsiders the method of Perry and interprets it as a memoryless BFGS updating formula. In this algorithm g_{k+1} is modified by a positive definite matrix which best estimates the inverse Hessian, without any additional storage requirements. For convex functions, under inexact line search Shanno [37] proved the global convergence of CONMIN.

6. Parametric conjugate gradient algorithms. The parametric conjugate gradient algorithms have been introduced in the same way that the quasi-Newton methods have been combined to get the Broyden or the Huang families. These algorithms are defined by (2) and (3) where the parameter β_k is as in Table 5.

Table 5. Parametric conjugate gradient algorithms.

Nr.	Formula	Author(s)
1.	$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{y_k^T s_k}$, $t > 0$ is a constant	Dai and Liao [12] (DL)

2.	$\beta_k^{DL+} = \max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\} - t\frac{s_k^T g_{k+1}}{y_k^T s_k}$, $t > 0$ is a constant.	Dai and Liao + [12] (DL+)
3.	$\beta_k^{YT} = \frac{g_{k+1}^T (z_k - ts_k)}{d_k^T z_k}$, where $z_k = y_k + \frac{\delta \xi_k}{s_k^T u_k} u_k$, $\xi_k = 6(f_k - f_{k+1}) + 3(g_k + g_{k+1})^T s_k$, $\delta \geq 0$ is a constant and $u_k \in R^n$ satisfies $s_k^T u_k \neq 0$; for example $u_k = d_k$.	Suggested by Yabe and Takano [41] (YT) based on a modified secant condition given by Zhang *et al*. [40]
4.	$\beta_k^{YT+} = \max\left\{0, \frac{g_{k+1}^T z_k}{d_k^T z_k}\right\} - t\frac{g_{k+1}^T s_k}{d_k^T z_k}$.	Suggested by Yabe and Takano plus [41] (YT+)
5.	$\beta_k = \frac{\|g_{k+1}\|^2}{\lambda_k \|g_k\|^2 + (1-\lambda_k) d_k^T y_k}$, $\lambda_k \in [0,1]$. The FR algorithm corresponds to $\lambda_k = 1$. The DY algorithm correspond to $\lambda_k = 0$.	Suggested by Dai and Yuan [14]
6.	$\beta_k = \frac{\mu_k \|g_{k+1}\|^2 + (1-\mu_k) g_{k+1}^T y_k}{\lambda_k \|g_k\|^2 + (1-\lambda_k) d_k^T y_k}$, $\lambda_k, \mu_k \in [0,1]$.	Suggested by Nazareth [30] This two parameter family includes the methods: FR, DY, PRP and HS in extreme cases.
7.	$\beta_k = \frac{\mu_k \|g_{k+1}\|^2 + (1-\mu_k) g_{k+1}^T y_k}{(1-\lambda_k-\omega_k)\|g_k\|^2 + \lambda_k d_k^T y_k - \omega_k d_k^T g_k}$, $\lambda_k, \mu_k \in [0,1]$ and $\omega_k \in [0, 1-\lambda_k]$.	Suggested by Dai and Yuan [14] This three parameter family includes the six classical conjugate gradient algorithms, as well as the previous one-parameter and two-parameter families.

7. Performance profiles. In this section we present the computational performance of a Fortran implementation of conjugate gradient algorithms on a set of 750 unconstrained optimization test problems. The test problems are the unconstrained problems in the CUTE [11] library, along with other large-scale optimization problems presented in [6]. We selected 75 large-scale unconstrained optimization problems in extended or generalized form. For each function we have considered ten numerical experiments with the number of variables $n = 1000, 2000, \ldots, 10000$. CG_DESCENT is authored by Hager and Zhang [21,22], CONMIN by Shanno and Phua [38]. The CG_DESCENT code contains the variant CG_DESCENT(w) implementing the Wolfe line search and the variant CG_DESCENT(aw) implementing an approximate Wolfe line search. The Wolfe conditions implemented in CG_DESCENT(w) can compute a solution with an accuracy on the order of the square root of the machine epsilon. In contrast, the approximate Wolfe line search implemented in CG_DESCENT(aw) can compute a solution with an accuracy of the order of machine epsilon. The rest of all algorithms considered in this study are authored by Andrei. All codes are written in double precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8GHz workstation.

All algorithms implement the Wolfe line search conditions with $\rho = 0.0001$ and $\sigma = 0.9$, and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$, where $\|.\|_\infty$ is the maximum absolute component of a vector.
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 $i = 1,\ldots,750$, respectively. We say that, in the particular problem i, the performance of ALG1 was better than the performance of ALG2 if:

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3}$$

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.
The performances of these algorithms have been evaluated using the profiles of Dolan and Moré [17] corresponding to this set of 750 test problems we extracted from the CUTE collection [11] and from [6]. 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.

In the first set of numerical experiments we compare the classical conjugate gradient algorithms. Figure 1 shows the CPU time performance profiles of these algorithms.

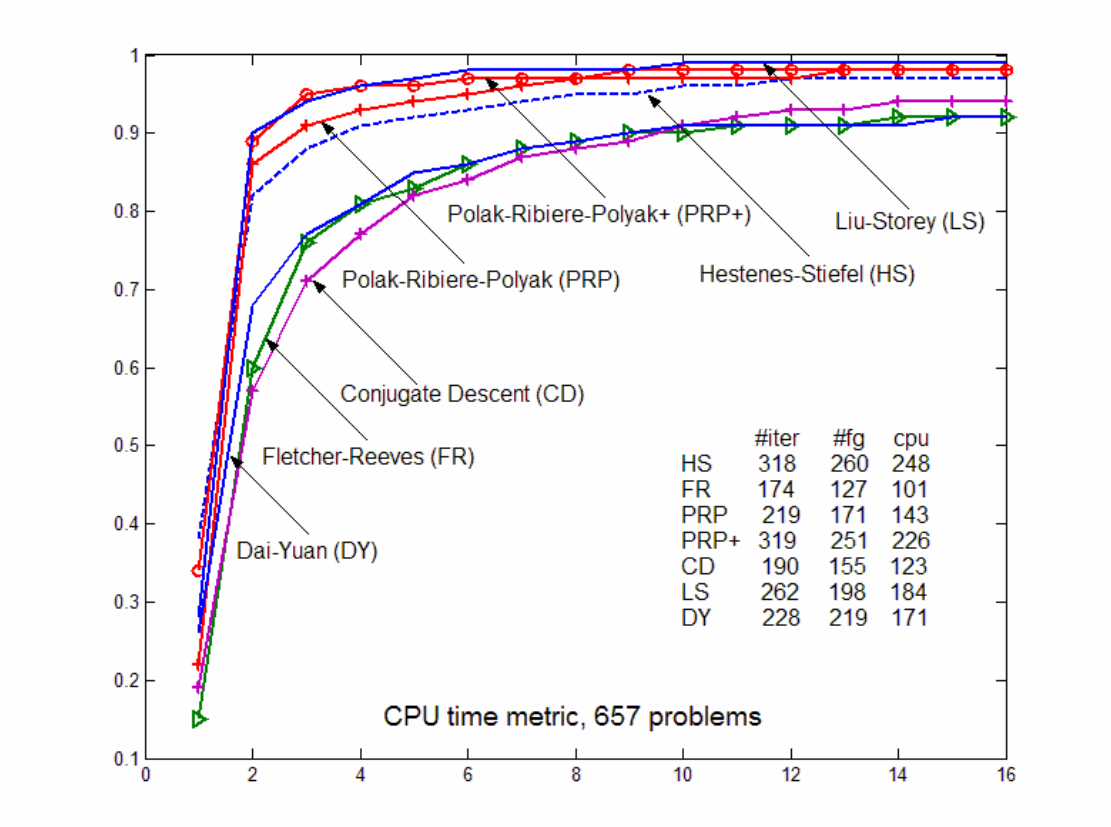
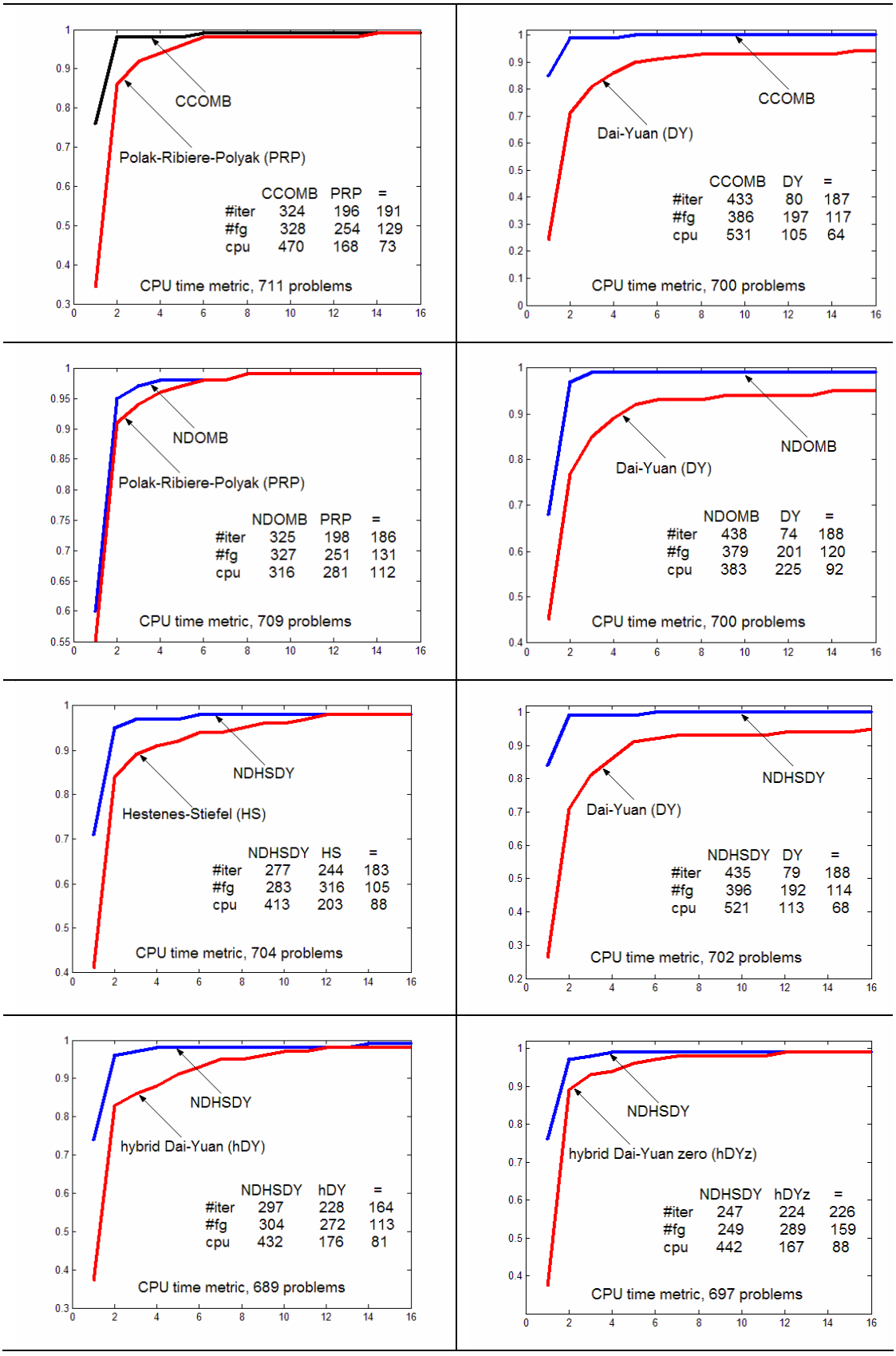


Fig. 1. Performance profiles of HS, FR, PRP, PRP+, CD, LS and DY.

From Figure 1 we see that the first set of methods FR, CD and DY although they have strong convergence properties, they may not perform well in practice due to jamming. In contrast, although the second set of methods HS, PRP and LS in general may not converge, they often perform better than the methods in the first set.

Figure 2 presents the performance profiles of some hybrid conjugate gradient algorithms.



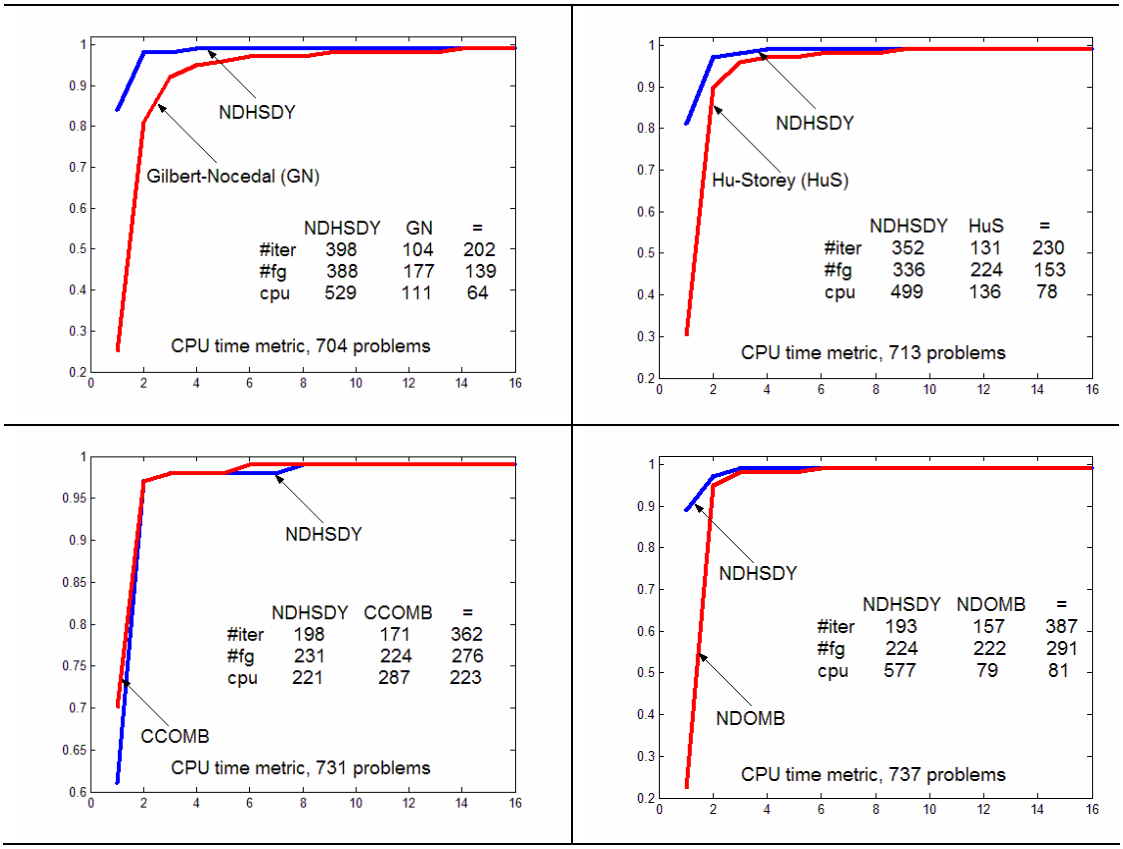
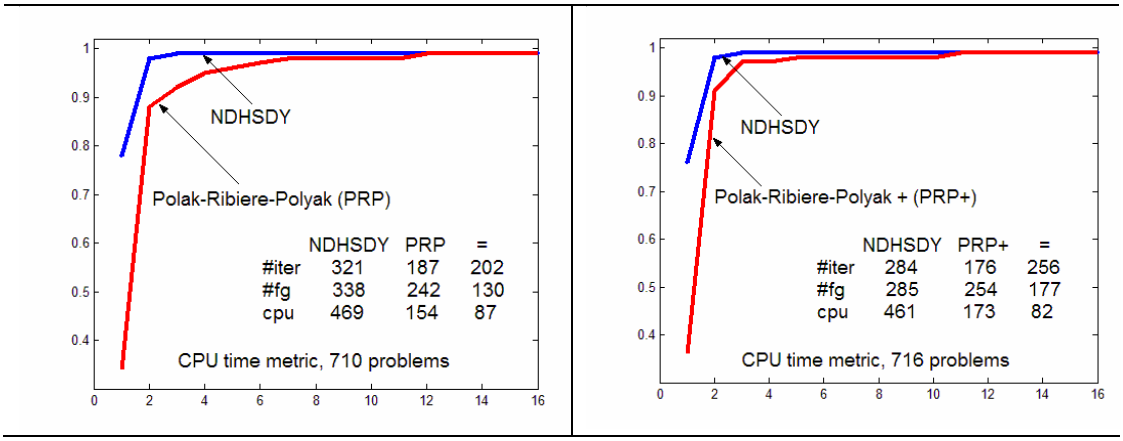


Fig. 2. Performance profiles of some hybrid conjugate gradient algorithms.

Figure 3 presents the performance profiles of NDHSDY versus the classical conjugate gradient algorithms: PRP, PRP+, LS and CD. It seems that the best algorithm is the hybrid algorithm NDHSDY given by a convex combination of HS and DY, where the parameter in the convex combination is obtained using the Newton direction.



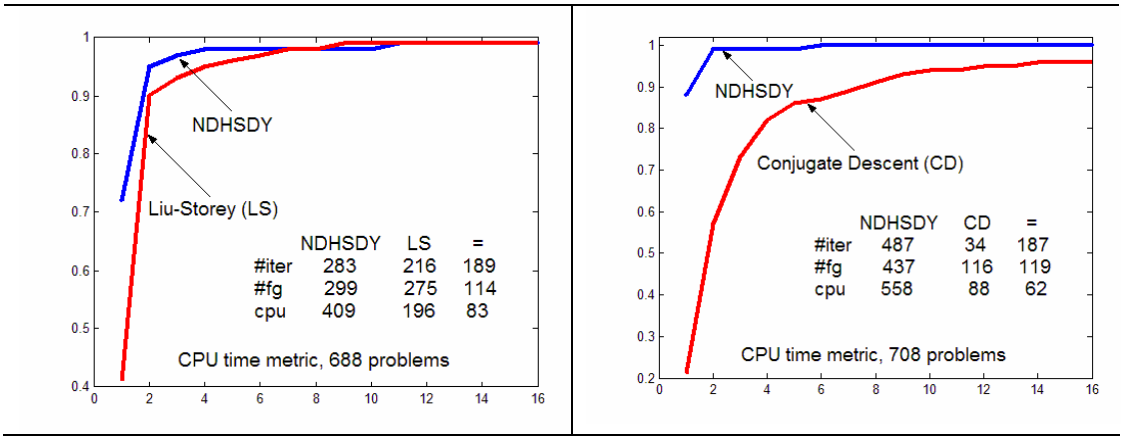


Fig. 3. Performance profiles of NDHSDY versus some classical conjugate gradient algorithms.

In the next set of numerical experiments we compare the scaled conjugate gradient algorithms. Figure 4 shows the performance profiles of SCALCG, BM, BM+, sPRP and sFR. We see that SCALCG algorithm is top performer among the scaled conjugate gradient algorithms.

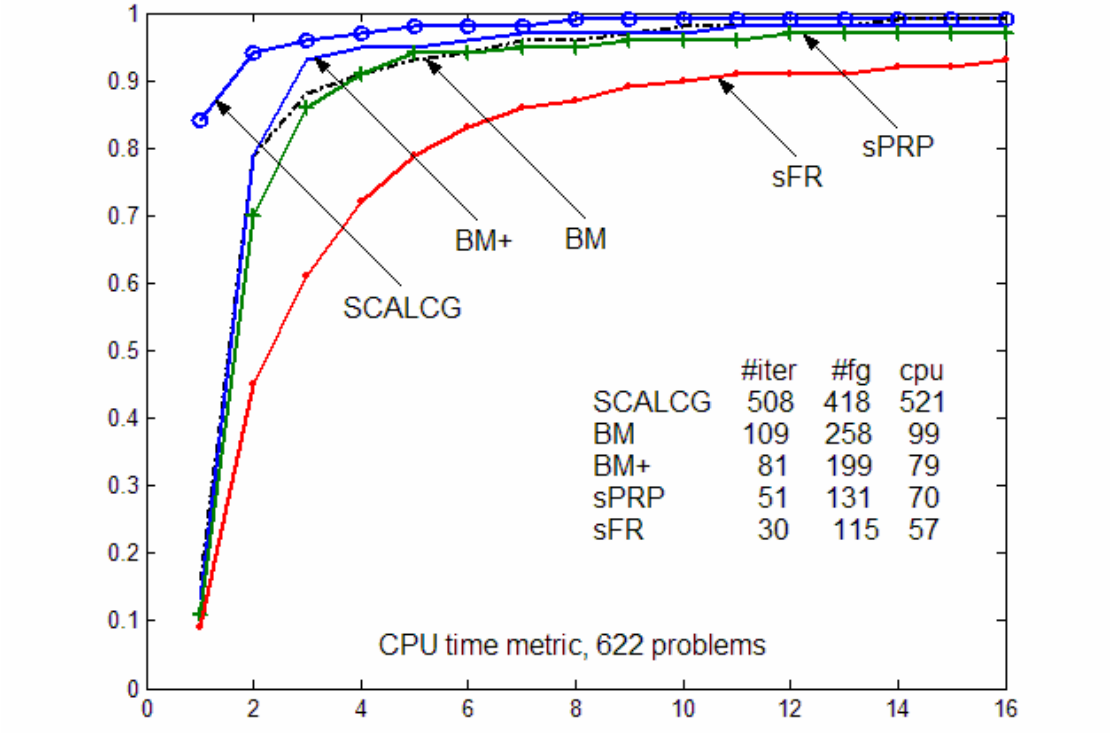


Fig. 4. Performance profiles of scaled conjugate gradient algorithms.

Figure 5 shows the performance profile of SCALCG versus classical conjugate gradient algorithms PRP and PRP+, as well as the hybrid algorithms CCOMB and NDHSDY.

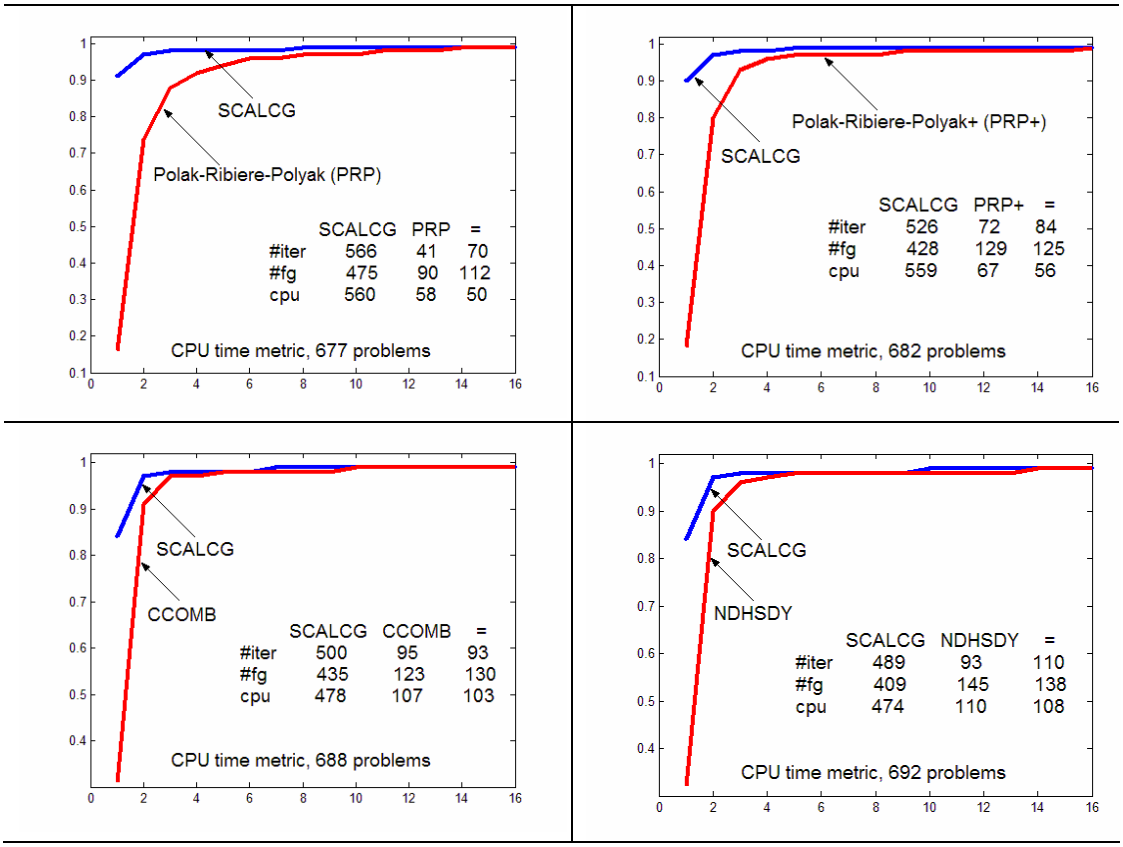


Fig. 5. Performance profiles of SCALCG versus PRP, PRP+, CCOMB and NDHSDY.

In the following we compare the modified conjugate gradient algorithms CG_DESCENT(w), ACGA, ACGA+, CGSD and APRP. Figure 6 presents the performance profiles of these algorithms.

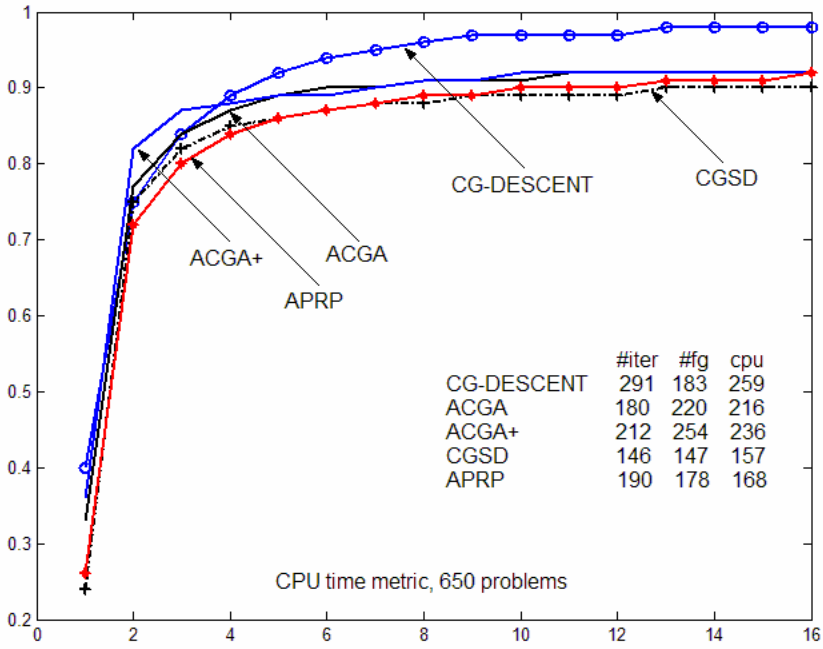


Fig. 6. Performance profiles of CG_DESCENT, ACGA, ACGA+, CGSD and APRP.

Figure 7 presents the performance profiles of CG_DESCENT(w) and PRP, PRP+, NDHSDY and SCALCG.

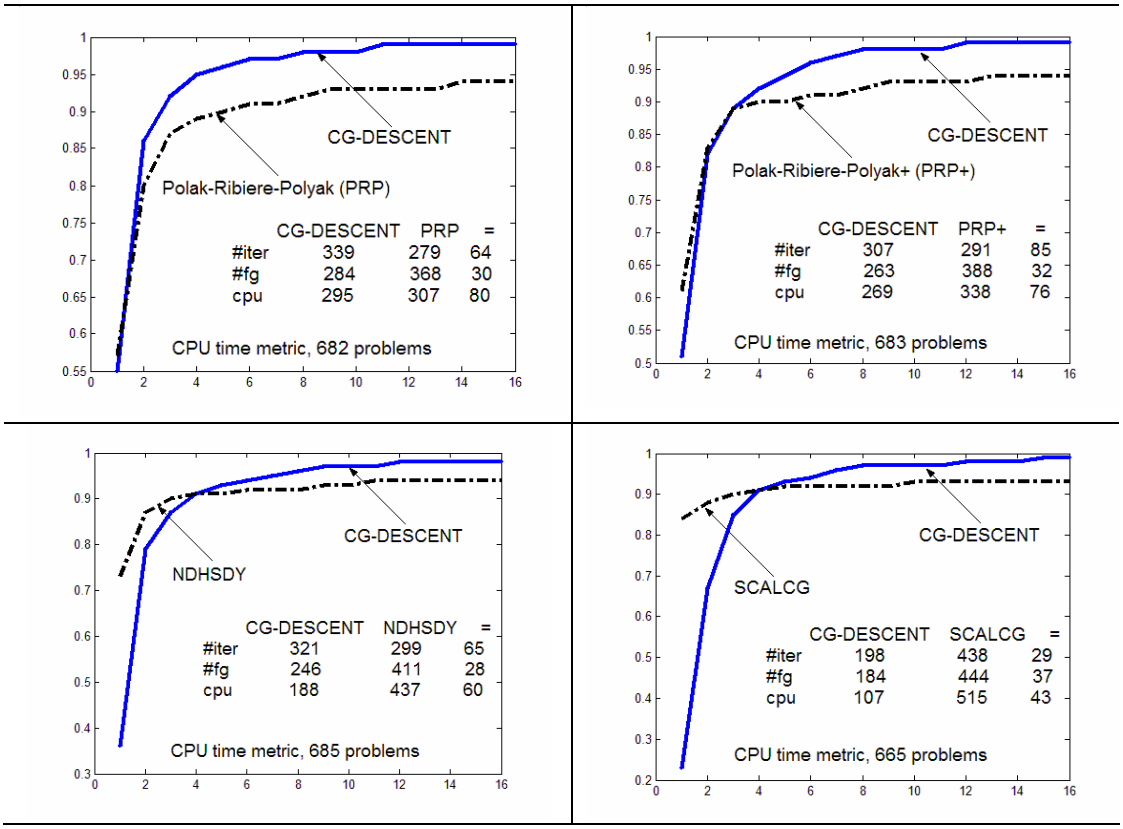


Fig. 7. Performance profiles of CG_DESCENT(w) versus PRP, PRP+, NDHSDY and SCALCG.

Now, comparing CONMIN with some other modified conjugate gradient algorithms: ACGA, ACGA+, CGSD and APRP, the following performance profiles have been obtained, as in Figure 8.

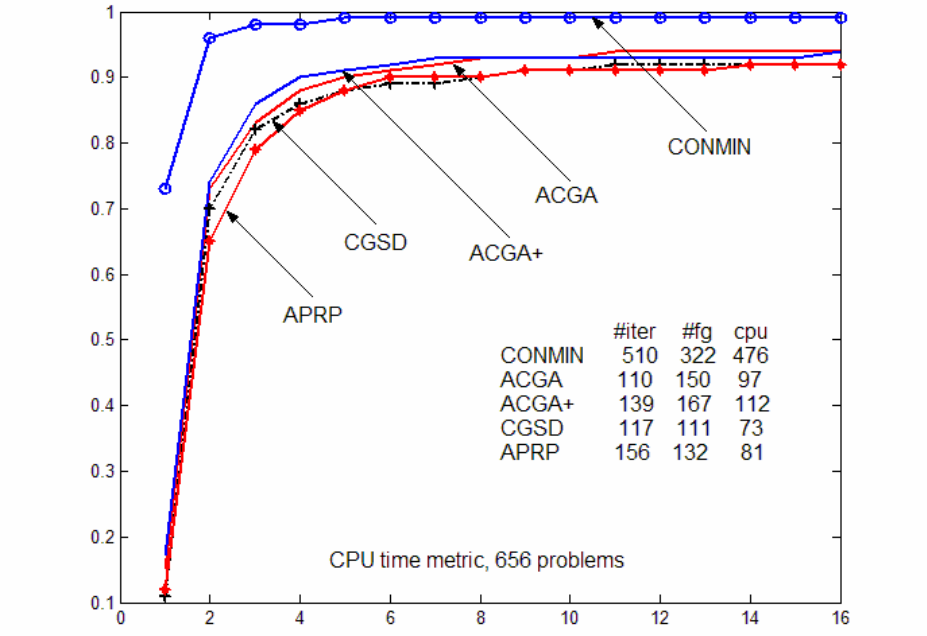


Fig. 8. Performance profiles of CONMIN, ACGA, ACGA+, CGSD and APRP.

We see that CONMIN is top performer. Figure 9 presents the performances profiles of CONMIN and PRP, NDHSDY, SCALCG and CG_DESCENT.

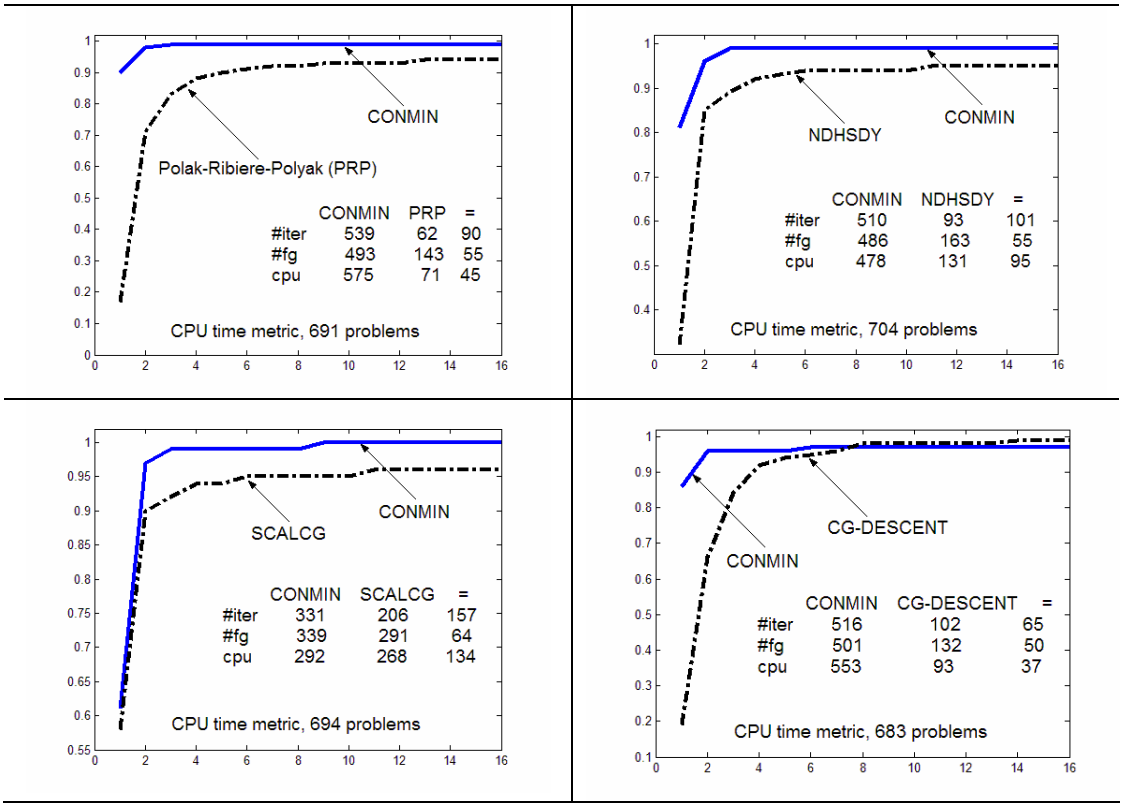
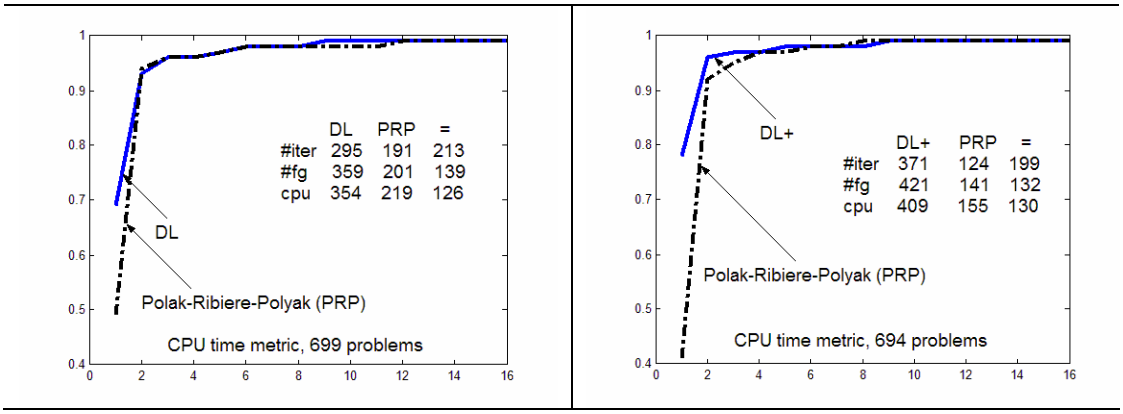


Fig. 9. Performance profiles of CONMIN versus PRP, NDHSDY, SCALCG and CG_DESCENT.

Finally, let us consider the parametric conjugate gradient algorithms DL(t=1) and DL+(t=1). Figure 10 shows the performance profiles of DL and DL+ versus PRP, SCALCG and CONMIN.



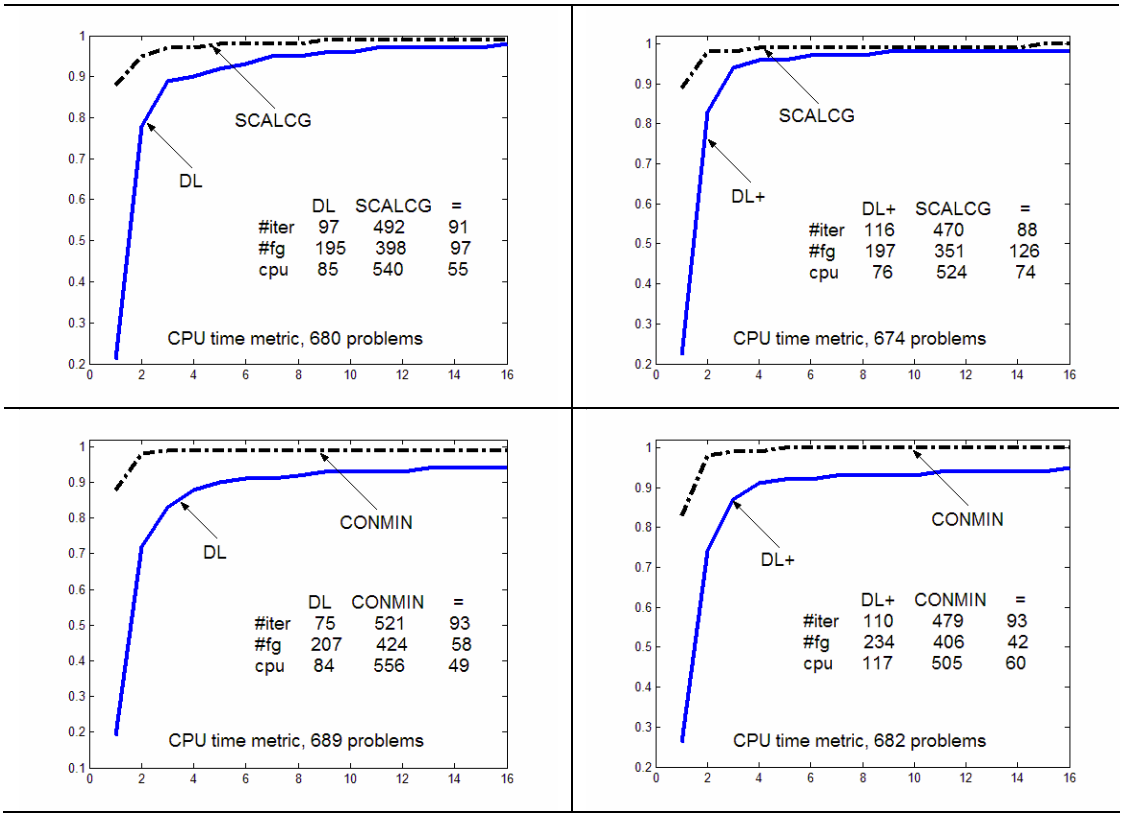


Fig. 10. Performance profiles of DL(t=1) and DL+(t=1) versus PRP, SCALCG and CONMIN.

8. Conclusion and discussion. Conjugate gradient algorithms are one of the most elegant and probably the simplest algorithms for computational nonlinear optimization. Their theory is well established (please, see [23]) and they proved to be surprisingly effective in solving real practical applications. The computational study presented here, which include 29 conjugate gradient algorithms, shows that the most effective are CONMIN, CG_DESCENT and SCALCG. Close to these algorithms is NDHSDY, a convex combination of HS and DY conjugate gradient algorithms in which the parameter is computed using the Newton direction. Concerning the robustness, CG_DESCENT is on the first place.

This computational study involves a large variety of nonlinear test functions. However, to conclude about the effectiveness of these algorithms, the test functions must be organized on some classes with well established characteristics, and to see which conjugate gradient algorithm is more successful. This remains to be explored.

It is worth seeing a comparison between the most successful conjugate gradient algorithms and quasi-Newton limited BFGS algorithm of Nocedal [31]. 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 [31] 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. Figure 11 shows the performance profiles of CONMIN, SCALCG, CG_DESCENT and

NDHSDY versus L-BFGS (m=3) an implementation given by Liu and Nocedal [26] using the line search of Moré and Thuente [28].

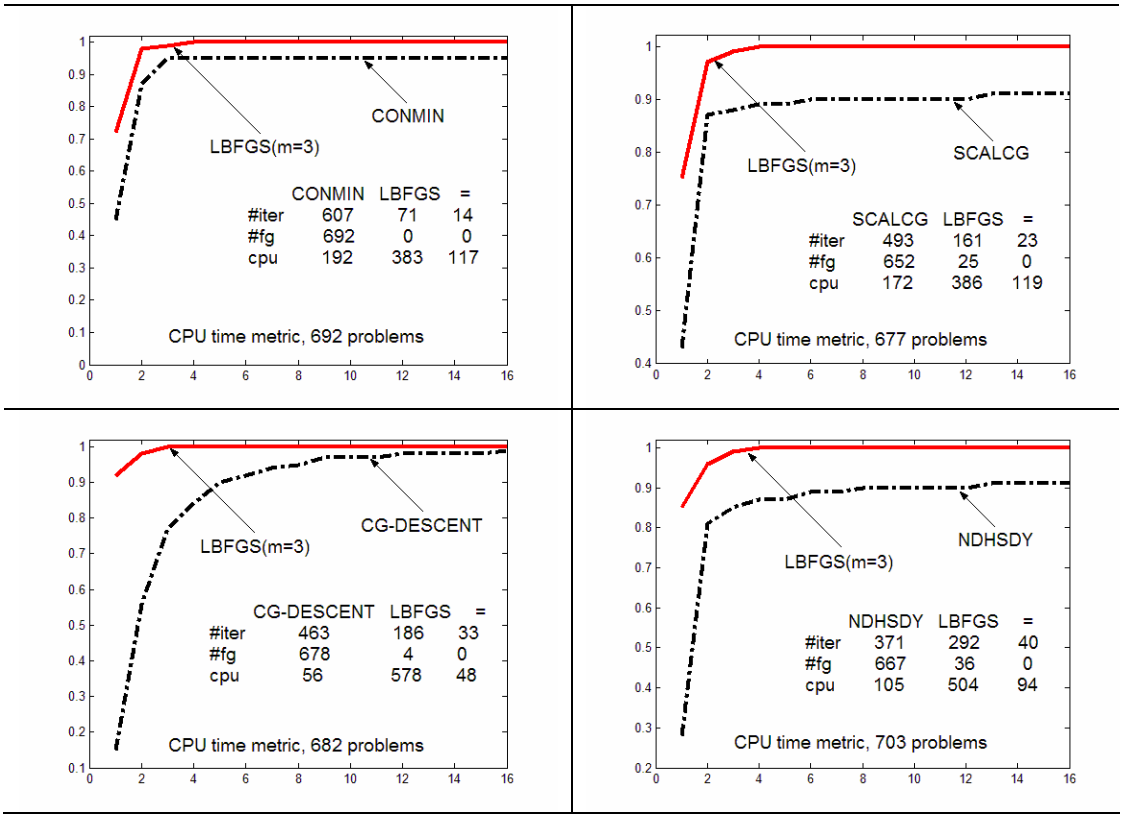


Fig. 11. Performance profiles of LBFGS(m=3) versus CONMIN, SCALCG, CG_DESCENT and NDHSDY conjugate gradient algorithms.

From Figure 11 we see that LBFGS (m=3) is way more successful than any conjugate gradient algorithm. Closest to LBFGS is CONMIN.

It is worth presenting a comparison of conjugate gradient algorithms with truncated Newton method TN by Nash [29]. The truncated Newton method uses an approximation of the Hessian matrix, and stops the solving process of the Newton system $B_k d_k = -\nabla f(x_k)$ as soon as a suitable termination criterion is satisfied. The truncated Newton method in TN implementation is preconditioned by a BFGS limited-memory quasi-Newton method with a further diagonal scaling. The Newton system is solved by means of a preconditioned conjugate gradient method. 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 [16] 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 $0 < r \le 1$.
Figure 12 shows the performance profiles of TN versus CONMIN, CG_DESCENT, SCALCG and NDHSDY.

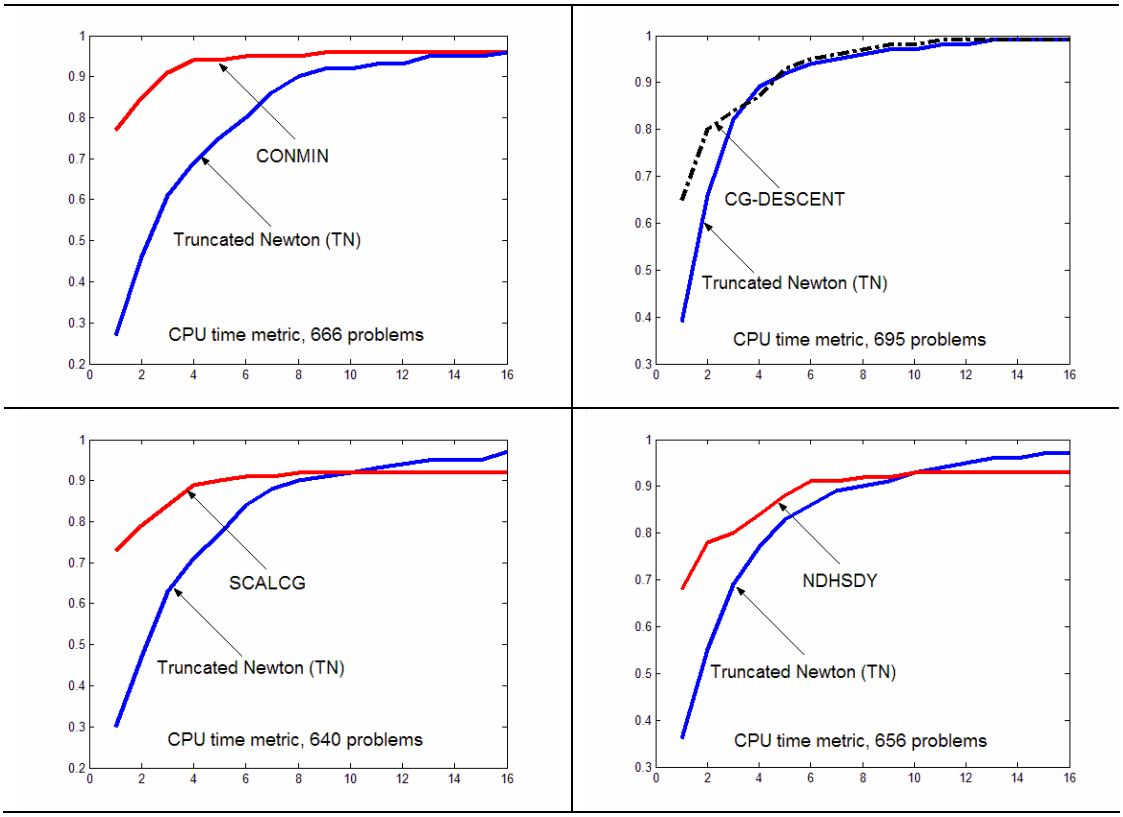


Fig. 12. Performance profiles of TN versus CONMIN, CG_DESCENT(w), SCALCG and NDHSDY algorithms.

Even that conjugate gradient methods are relevant nonlinear optimization methods, there are some open problems which deserve additional research.

1) In contrast to the quasi-Newton methods for which the steplength for the vast majority of iterations is equal to 1, the steplength in conjugate gradient methods differ from 1, being larger or smaller up to two order of magnitude depending on how the problem is scaled. In conjugate gradient methods the size of α_k vary in a very unpredictabe way.

2) Another open problem is the preconditioning of conjugate gradient algorithms. The scaled conjugate gradient algorithms by Birgin and Martínez [10] and Andrei [1-4] introduce a scaling of g_{k+1} in the direction d_{k+1} computation. However, if the definition of θ_{k+1} in (7) does contain enough information about the inverse Hessian of the minimizing function, then better is to use the search direction $d_{k+1} = -\theta_{k+1} g_{k+1}$, since the addition of the term $\beta_k s_k$ in (7) may prevent d_{k+1} to be a descent direction unless the line search is sufficiently accurate. In scaled conjugate gradient algorithms there is a very delicate balance between $-\theta_{k+1} g_{k+1}$ and $\beta_k s_k$, which brings into attention the preconditioning question.

3) Another open problem with conjugate gradient methods is that the structure of the minimizing problem is not taken into account to design more efficient computational schemes. This is in sharp contrast to quasi-Newton or truncated Newton methods.

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., *A Dai-Yuan conjugate gradient algorithm with sufficient descent and conjugacy conditions for unconstrained optimization.* Applied Mathematics Letters, accepted.
[6] ANDREI, N., *Test functions for unconstrained optimization*. http:// www.ici.ro/ camo/ neculai /SCALCG /evalfg.for
[7] ANDREI, N., *New hybrid conjugate gradient algorithms for unconstrained optimization*. ICI Thechnical Report, September 26, 2007.
[8] ANDREI, N., *Another hybrid conjugate gradient algorithm for unconstrained optimization.* ICI Technical Report, July 18, 2007.
[9] ANDREI, N., *Another nonlinear conjugate gradient algorithm for unconstrained optimization*. ICI Thechnical Report, May 17, 2007.
[10] BIRGIN, E., MARTÍNEZ, M., *A spectral conjugate gradient method for unconstrained optimization*, Applied Math. and Optimization (2001) 43, pp.117-128.
[11] 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.
[12] DAI, Y.H., LIAO, L.Z., *New conjugacy conditions and related nonlinear conjugate gradient methods.* Appl. Math. Optim., 43, (2001), pp.87-101.
[13] DAI, Y.H., YUAN, Y., *A nonlinear conjugate gradient method with a strong global convergence property,* SIAM J. Optim., 10 (1999) pp.177-182.
[14] DAI, Y.H., YUAN, Y., *A three-parameter family of hybrid conjugate gradient method.* Mathematics of Computation, 70 (2001) pp.1155-1167.
[15] DAI, Y.H., YUAN, Y., *A class of globally convergent conjugate gradient methods.* Sci. China Ser. A, 46 (2003), pp.251-261.
[16] DEMBO, R.S., EISENSTAT, S.C., STEIHAUG, T., *Inexact Newton methods.* SIAM J. Num. Anal. 19 (1982), pp.400-408.
[17] DOLAN, E.D., MORÉ, J.J., *Benchmarcking optimization software with performance profiles*. Mathematical Programming, vol. 91, pp.201-213, 2002.
[18] FLETCHER, R., *Practical Methods of Optimization.* Second edition. John Wiley & Sons, Chichester, 1987.
[19] FLETCHER, R., REEVES, C., *Function minimization by conjugate gradients.* Comput. J., 7 (1964), pp.149-154.
[20] GILBERT, J.C., NOCEDAL J., *Global convergence properties of conjugate gradient methods.* SIAM Journal on Optimization, 2 (1992), pp.21-42.
[21] 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.
[22] 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.
[23] HAGER, W.W., ZHANG, H., *A survey of nonlinear conjugate gradient methods,* Pacific Journal of Optimization, 2 (2006), pp.35-58.
[24] HESTENES, M.R., STIEFEL, E., *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards Sec. B. 48 (1952) 409-436.
[25] HU, Y.F., STOREY, C., *Global convergence result for conjugate gradient methods.* J. Optim. Theory Appl., 71 (1991), pp.399-405.
[26] LIU, D., NOCEDAL, J., *On the limited memory BFGS method for large scale optimization*, Mathematical Programming B 45 (1989) 503-528.
[27] LIU, D.C., STOREY, *Efficient generalized conjugate gradient algorithms. Part 1: Theory.* Journal on Optimization Theory and Applications. 69 (1991), pp.129-137.
[28] MORÉ, J.J., THUENTE, D.J., *Line search algorithms with guaranteed sufficient decrease*. ACM Transactions on Mathematical Software, 20, 1994, pp.286-307.
[29] NASH, S.G., *Preconditioning of truncated-Newton methods.* SIAM J. Sci. Comp. 6 (1985) pp. 599-616.
[30] NAZARETH, J.L., *Conjugate gradient methods.* Encyclopedia of Optimization, C.Floudas and P. Pardalos, (Eds.) Kluwer Academic Publishers, Boston, 1999.
[31] NOCEDAL, J., *Updating quasi-Newton matrices with limited starage.* Mathematics of Computation, 35 (1980), pp.773-782.
[32] PERRY, A., *A modified conjugate gradient algoritm.* Operations Research, 26 (1978), pp.1073-1078.
[33] POLAK, E., RIBIERE, G., *Note sur la convergence de directions conjugées.* Rev. Francaise Informat. Recherche Operationelle, 3e Année 16, (1969) pp.35-43.

[34] POLYAK, B.T., *The conjugate gradient method in extreme problems.* URSS Comp. Math. Math. Phys., 9, (1969), pp.94-112
[35] POWELL, M.J.D., *Nonconvex minimization calculations and the conjugate gradient method*. Numerical Analysis (Dundee, 1983), Lecture Notes in Mathematics, vol.1066, Springer Verlag, Berlin, 1984, pp.122-141.
[36] SHANNO, D.F., *Conjugate gradient methods with inexact searches*. Mathematics of Operations Research, vol. 3 (1978), pp.244-256.
[37] SHANNO, D.F., *On the convergence of a new conjugate gradient algorithm.* SIAM J. Numer. Anal., 15 (1978), pp.1247-1257.
[38] SHANNO, D.F., PHUA, K.H., *Algorithm 500, Minimization of unconstrained multivariate functions*, ACM Trans. on Math. Software, 2 (1976) 87-94.
[39] TOUATI-AHMED, D., STOREY, C., *Efficient hybrid conjugate gradient techniques.* Journal of Optimization Theory and Applications, 64 (1990), pp.379-397.
[40] ZHANG, J.Z., DENG, N.Y., CHEN, L.H., *New quasi-Newton equation and related methods for unconstrained optimization.* J. Optim. Theory Appl., 102 (1999), pp.147-167.
[41] YABE, H. AND TAKANO, M., *Global convergence properties of nonlinear conjugate gradient methods with modified secant conditions.* Comput. Optim. Appl., (COAP) 28, (2004), pp.203-225.

October 10, 2007
Paper for "Studies in Informatics and Control"