CGALL

PERFORMANCE PROFILE OF CONJUGATE GRADIENT ALGORITHMS

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

Abstract. In this work we present the performance profile of 23 conjugate gradient algorithms implemented in CGALL package The conjugate gradient algorithms are classified in 5 groups: ***classical*** (HS, FR, PRP, PRP+, CD, LS and DY); ***hybrid*** (hDY, hDYz, GN, HuS, TaS and LS-CD); ***scaled*** (BM, BM+, sPRP and sFR); ***modified*** (ASDC, A, ACGSD and ACGSDz) and ***parametric*** (DL and DL+). We give computational evidence that ACGSD, PRP+, hDYz and BM conjugate gradients are the top performer among the algorithms considered in this study.

1. Introduction.

In this work we present the computational performance profile of the conjugate gradient algorithms. We analyze 23 conjugate gradient algorithms which are implemented in CGALL package. CGALL is a Fortran package.

The conjugate gradient algorithms are defined by the following recurence:

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

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

where α_k is the steplength determined by the Wolfe line search and the parameter β_k is computed as in Table 1.

Table 1. Conjugate gradient algorithms considered in CGALL package.

Nr.	File	Formula	Name
1.	Z1	$\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$	Hestenes and Stiefel (HS)
2.	Z2	$\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	Fletcher and Reeves (FR)
3.	Z3	$\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$	Polak-Ribiere and Polyak (PRP)
4.	Z4	$\beta_k^{PRP+} = max\left\{0, \frac{y_k^T g_{k+1}}{g_k^T g_k}\right\}$	Polak-Ribiere and Polyak + (PRP+)
5.	Z5	$\beta_k^{CD} = -\frac{g_{k+1}^T g_{k+1}}{g_k^T d_k}$	Conjugate Descent – Fletcher (CD)
6.	Z6	$\beta_k^{LS} = -\frac{y_k^T g_{k+1}}{g_k^T d_k}$	Lui and Storey (LS)

7.	Z7	$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$	Dai and Yuan (DY)
8.	Z8	$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{y_k^T s_k}$	Dai and Liao (DL)
9.	Z9	$\beta_k^{DL+} = max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\} - t\frac{s_k^T g_{k+1}}{y_k^T s_k}$	Dai and Liao + (DL+)
10.	Z10	$\beta_k^{ASDC} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k} - \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{(y_k^T s_k)^2}$	Andrei Sufficient Descent Condition. Please, see the paper for AML, AML5382. ($y_k^T \bar{Q}_{k+1} = 0$)
11.	Z11	$\beta_k^{hDY} = max\left\{c\beta_k^{DY}, min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}$	Hybrid DY (hDY)
12.	Z12	$\beta_k^{hDYz} = max\left\{0, min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}$	Hybrid DY zero (hDYz)
13.	Z13	$\beta_k^{GN} = max\left\{-\beta_k^{FR}, min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Gilbert and Nocedal (GN)
14.	Z14	$\beta_k^{HuS} = max\left\{0, min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Hu and Storey (HuS)
15.	Z15	$\beta_k^{TaS} = \begin{cases} \beta_k^{PRP} & 0 \le \beta_k^{PRP} \le \beta_k^{FR}, \\ \beta_k^{FR} & \text{otherwise} \end{cases}$	Touati-Ahmed and Storey (TaS)
16.	Z16	$\beta_k^{LS-CD} = max\left\{0, min\left\{\beta_k^{LS}, \beta_k^{CD}\right\}\right\}$	Hybrid LS, CD (LS-CD)
17.	Z17	$\beta_k^{BM} = \frac{(\theta y_k - s_k)^T g_{k+1}}{y_k^T s_k}$	Birgin and Martinez, Scaled Perry
18.	Z18	$\beta_k^{BM+} = max\left\{0, \frac{\theta y_k^T g_{k+1}}{y_k^T s_k}\right\} - \frac{s_k^T g_{k+1}}{y_k^T s_k}$	Brigin and Martinez +
19.	Z19	$\beta_k^{sPRP} = \frac{\theta_{k+1} y_k^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k}$	Scaled Polak-Ribiere-Polyak (sPRP)
20.	Z20	$\beta_k^{sFR} = \frac{\theta_{k+1} g_{k+1}^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k}$	Scaled Fletcher-Reeves (sFR)
21.	Z21	$\beta_k^{A} = \frac{1}{y_k^T s_k}\left(y_k^T g_{k+1} - \frac{(y_k^T y_k)(s_k^T g_{k+1})}{g_k^T g_k}\right)$	Andrei (Sufficient descent condition from PRP) Please, see Remark 8.3.3 and formula (8.3.130) in the book: Neculai Andrei, "*Criticism of the Unconstrained Optimization Algorithms Reasoning*".
22.	Z22	$\beta_k^{ACGSD} = \frac{y_k^T g_{k+1}}{y_k^T s_k} - \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{(y_k^T s_k)^2}$	Andrei (Sufficient descent condition from DY) (ACGSD) Please see paper for SIOPT, #067836. ($y_k^T \bar{Q}_{k+1} g_{k+1} = 0$)
23.	Z23	$\beta_k^{ACGSDz} = max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\}\left(1 - \frac{s_k^T g_{k+1}}{y_k^T s_k}\right)$	Andrei (Sufficient descent condition from DY zero) (ACGSDz)

In the scaled conjugate gradient the searching direction is computed as

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

where the parameter θ_k is the inverse of the Rayleigh quotient :

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

and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

The test problems are the unconstrained problems in the CUTE library, along with other large-scale optimization problems. 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 $n = 1000,2000,\ldots,10000.$

All algorithms implement the Wolfe line search conditions with $\sigma_1 = 0.0001$ and $\sigma_2 = 0.9$, 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.

All codes are written in double precision Fortran and compiled with f77 (default compiler settings) on an Intel Pentium 4, 1.8GHz workstation. CGALL package was designed and written by Andrei. The Dolan-Moré performance profile is given by means of PERF2N and PERFNN Fortran programs, written by Andrei.

2. Performance Profiles of Classical Conjugate Gradient Algorithms

In this section we present the performance profile of Dolan and Moré, corresponding to the classical conjugate gradient algorithms implemented in CGALL. Table 2 presents the classical conjugate gradient algorithms.

Table 2. Classical conjugate gradient algorithms.

Nr.	File	Formula	Name
1.	Z1	$\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T s_k}$	Hestenes and Stiefel (HS)
2.	Z2	$\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$	Fletcher and Reeves (FR)
3.	Z3	$\beta_k^{PRP} = \frac{y_k^T g_{k+1}}{g_k^T g_k}$	Polak-Ribiere and Polyak (PRP)
4.	Z4	$\beta_k^{PRP+} = max\left\{0, \frac{y_k^T g_{k+1}}{g_k^T g_k}\right\}$	Polak-Ribiere and Polyak + (PRP+)
5.	Z5	$\beta_k^{CD} = -\frac{g_{k+1}^T g_{k+1}}{g_k^T d_k}$	Conjugate Descent – Fletcher (CD)

6.	Z6	$\beta_k^{LS} = -\frac{y_k^T g_{k+1}}{g_k^T d_k}$	Lui and Storey (LS)
7.	Z7	$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k}$	Dai and Yuan (DY)

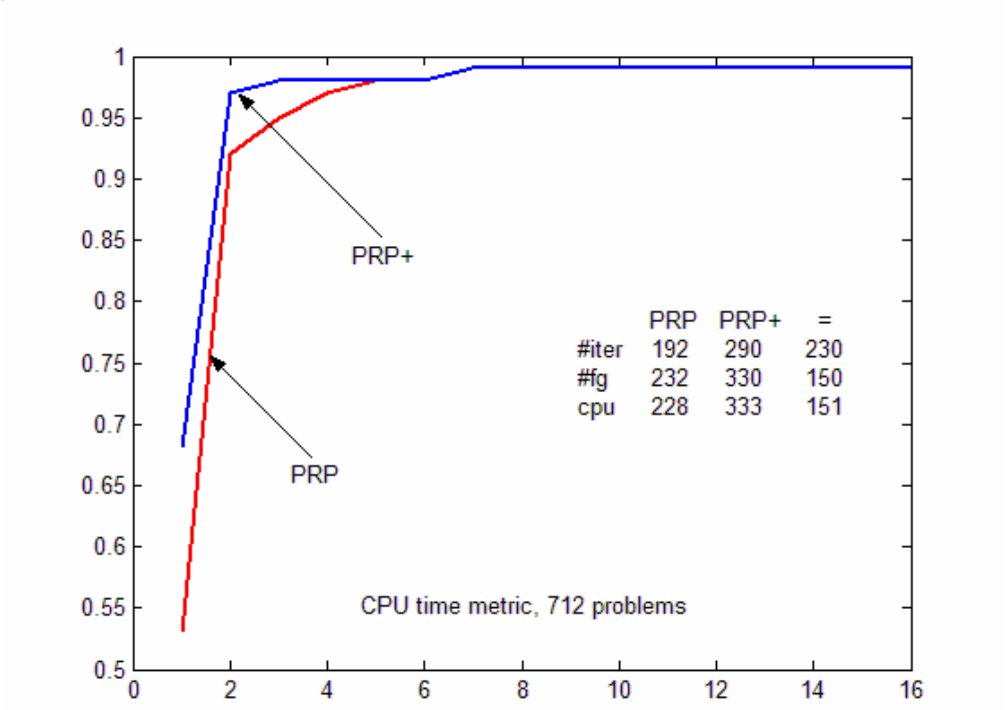

Fig. 2.1. Performance profile of Polak-Ribière-Polyak (PRP) and Polak-Ribière-Polyak+ (PRP+).

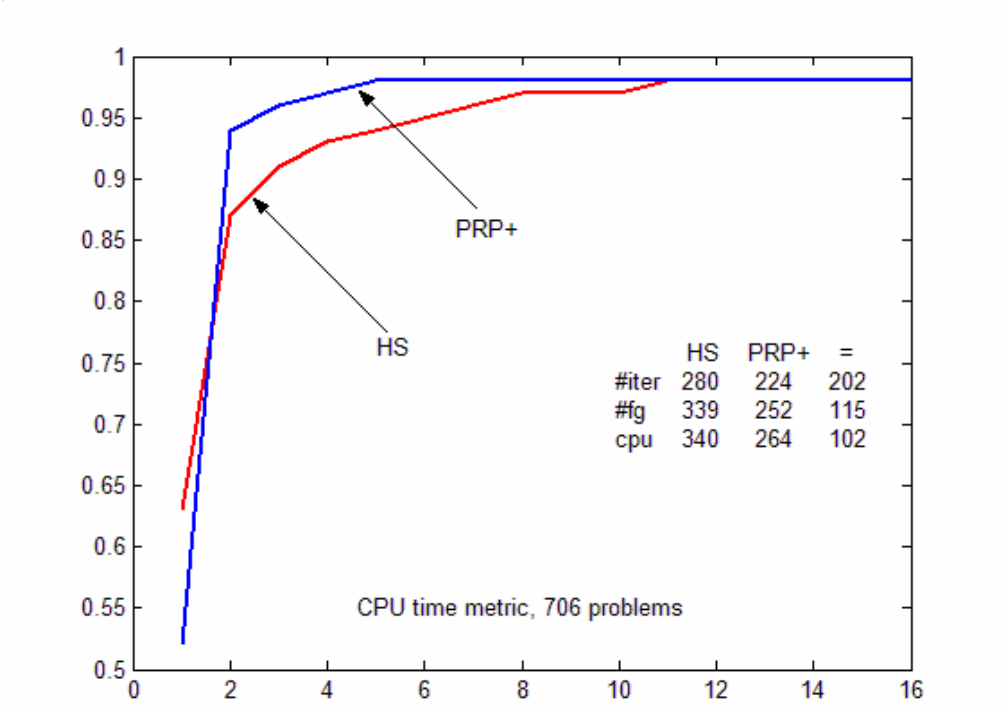

Fig. 2.2. Performance profile of Hestenes-Stiefel (HS) and Polak-Ribière-Polyak+ (PRP+).

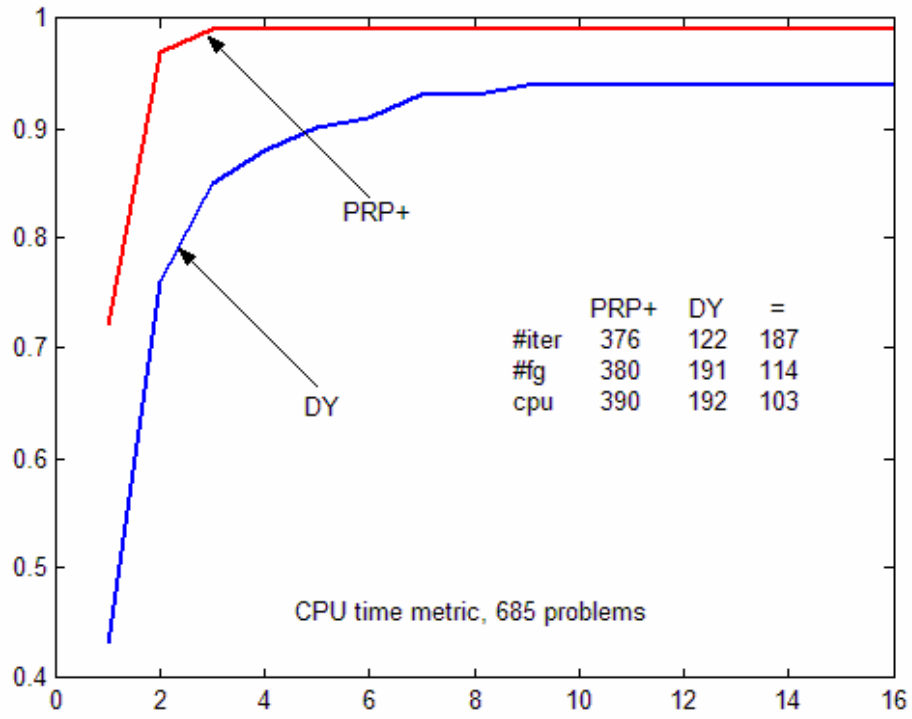

Fig. 2.3. Performance profile of Polak-Ribière-Polyak+ (PRP+) and Dai-Yuan (DY).

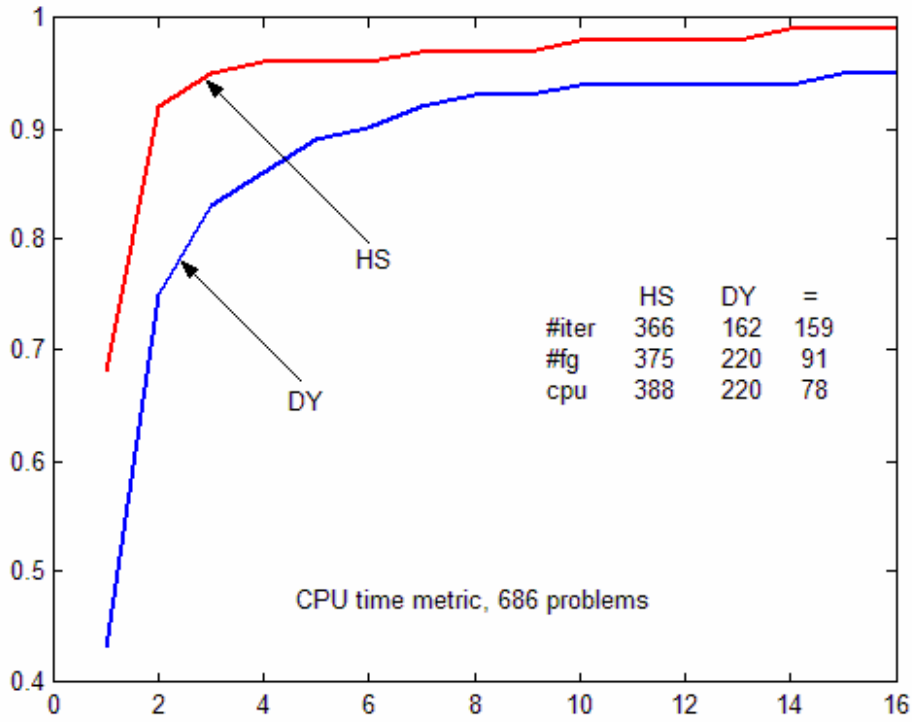

Fig. 2.4. Performance profile of Hestenes-Stiefel (HS) and Dai-Yuan (DY).

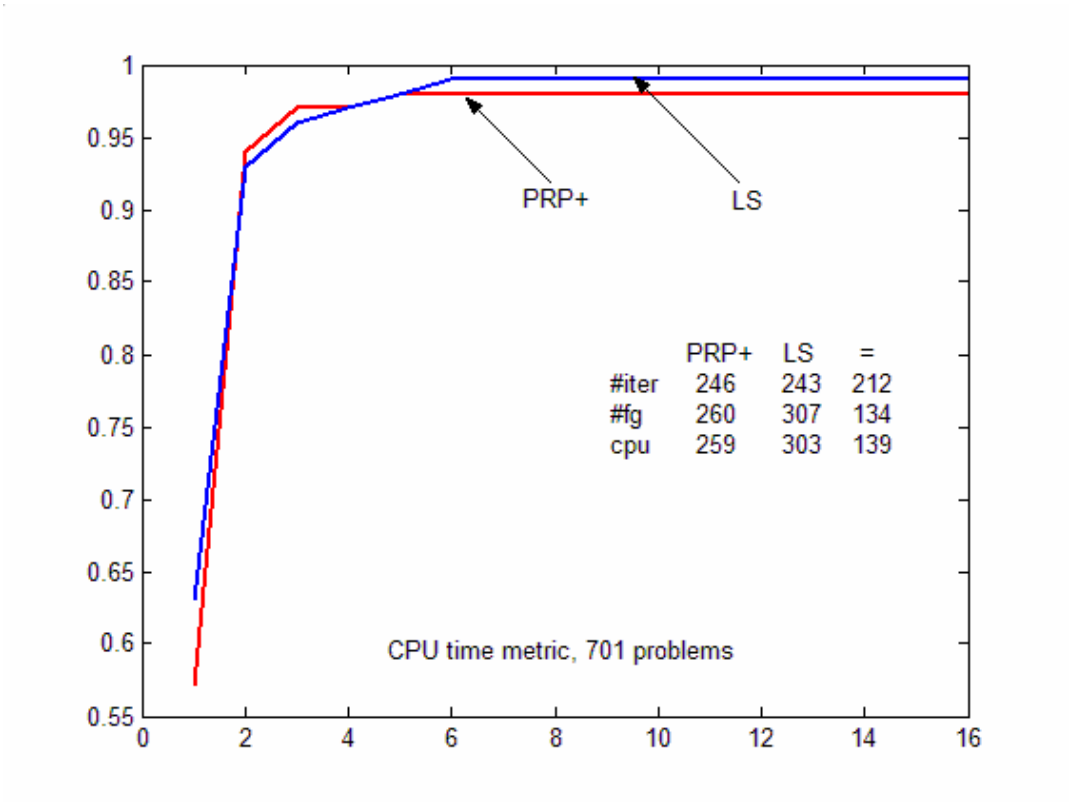

Fig. 2.5. Performance profile of Polak-Ribière-Polyak+ (PRP+) and Liu-Storey (LS).

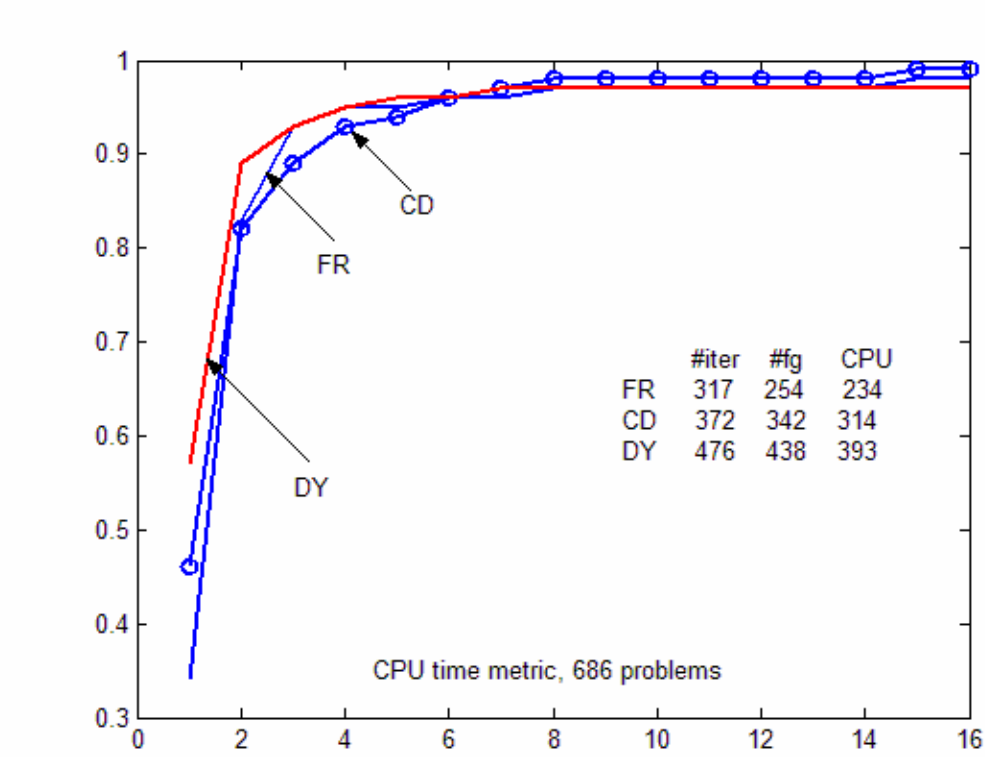

Fig. 2.6 Performance profile of Feltcher-Reeves (FR), Conjugate-Descent (CD) and Dai-Yuan (DY).

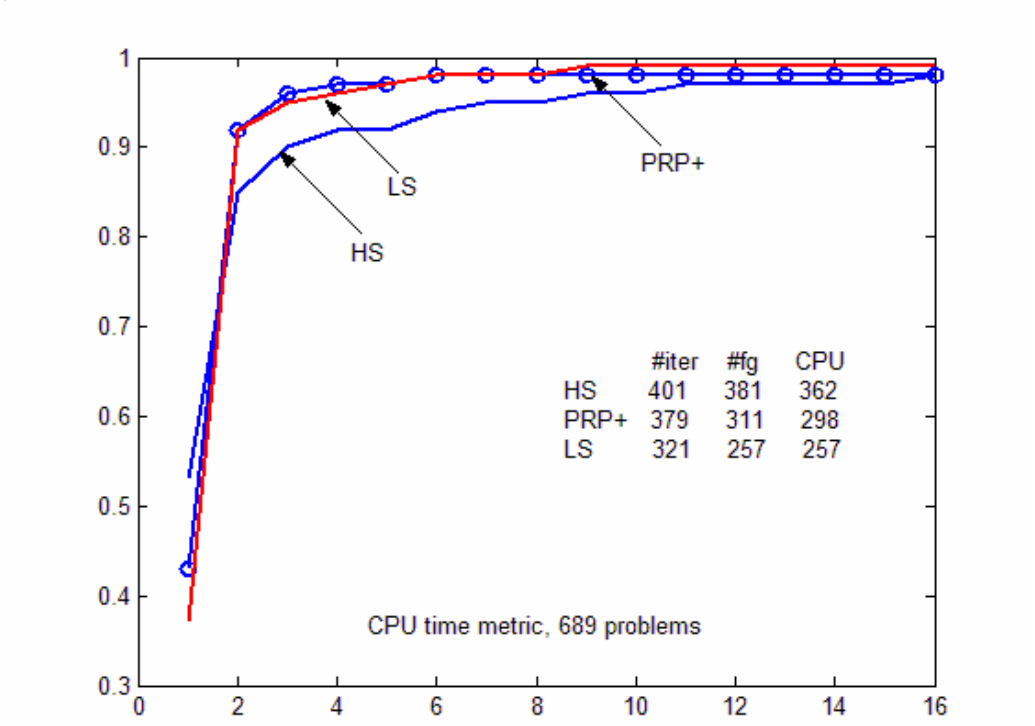

Fig. 2.7 Performance profile of Hestenes-Stiefel (HS), Polak-Ribière-Polyak+ (PRP+) and Liu-Storey (LS).

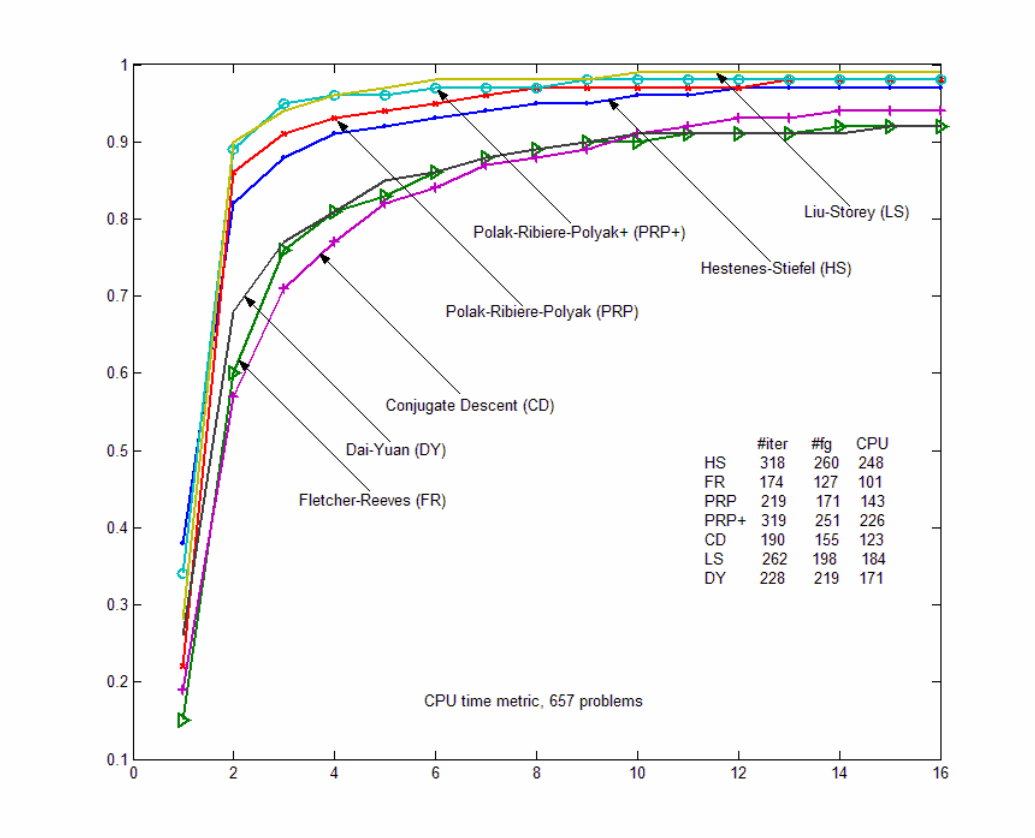

Fig. 2.8. Performance profile of HS, FR, PRP, PRP+, CD, LS, DY.

Hager and Zhang[1] present an excellent survey of conjugate gradient algorithms insisting on their global convergence properties. The conjugate gradient algorithms are classified in two large groups:

- *methods with* $\|g_{k+1}\|^2$ *in the numerator of* β_k *, like FR, CD, DY and*
- *methods with* $g_{k+1}^T y_k$ *in the numerator of* β_k *, like HS, PRP, PRP+, LS.*

Despite the strong convergence theory that has been developed for methods with $\|g_{k+1}\|^2$ in the numerator of β_k, all these methods are susceptible to jamming, i.e. they begin to take small steps without making a significant progress to the minimum. On the other hand, the methods with $g_{k+1}^T y_k$ in the numerator of β_k possess a built-in restart feature that addresses the jamming phenomenon. The methods with $g_{k+1}^T y_k$ in the numerator of β_k automatically adjust β_k to avoid jamming, having better computational performances than the performance of methods with $\|g_{k+1}\|^2$ in the numerator of β_k. This is illustrated in Figure 1.8 above, where the CPU performance profile of all classical conjugate gradient algorithms is presented.

We notice in Figure 1.8 that the performance profile of the classical methods, are grouped as the classification of Hager and Zhang. The methods with $g_{k+1}^T y_k$ in the numerator of β_k (HS, PRP, PRP+, LS) are more robust.

In Figure 1.7 we see that the LS and PRP+ algorithms are top performer in their class. They are the most robust, having the best computational performances.

3. Performance Profiles of Hybrid Conjugate Gradient Algorithms

In this section we consider the hybrid conjugate gradient algorithms where the parameter β_k is computed as in Table 3. These methods represent a combination of the classical conjugate gradient methods proposed to exploit the attractive properties both of methods with $\|g_{k+1}\|^2$ in the numerator of β_k and the methods with $g_{k+1}^T y_k$ in the numerator of β_k.

Table 3. Hybrid conjugate gradient algorithms.

Nr.	File	Formula	Author(s)
1.	Z11	$\beta_k^{hDY} = max\left\{c\beta_k^{DY}, min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}$	Hybrid DY (hDY)
2.	Z12	$\beta_k^{hDYz} = max\left\{0, min\left\{\beta_k^{HS}, \beta_k^{DY}\right\}\right\}$	Hybrid DY zero (hDYz)
3.	Z13	$\beta_k^{GN} = max\left\{-\beta_k^{FR}, min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Gilbert and Nocedal (GN)
4.	Z14	$\beta_k^{HuS} = max\left\{0, min\left\{\beta_k^{PRP}, \beta_k^{FR}\right\}\right\}$	Hu and Storey (HuS)
5.	Z15	$\beta_k^{TaS} = \begin{cases} \beta_k^{PRP} & 0 \le \beta_k^{PRP} \le \beta_k^{FR}, \\ \beta_k^{FR} & \text{otherwise} \end{cases}$	Touati-Ahmed and Storey (TaS)
6.	Z16	$\beta_k^{LS-CD} = max\left\{0, min\left\{\beta_k^{LS}, \beta_k^{CD}\right\}\right\}$	Hybrid LS, CD (LS-CD)

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

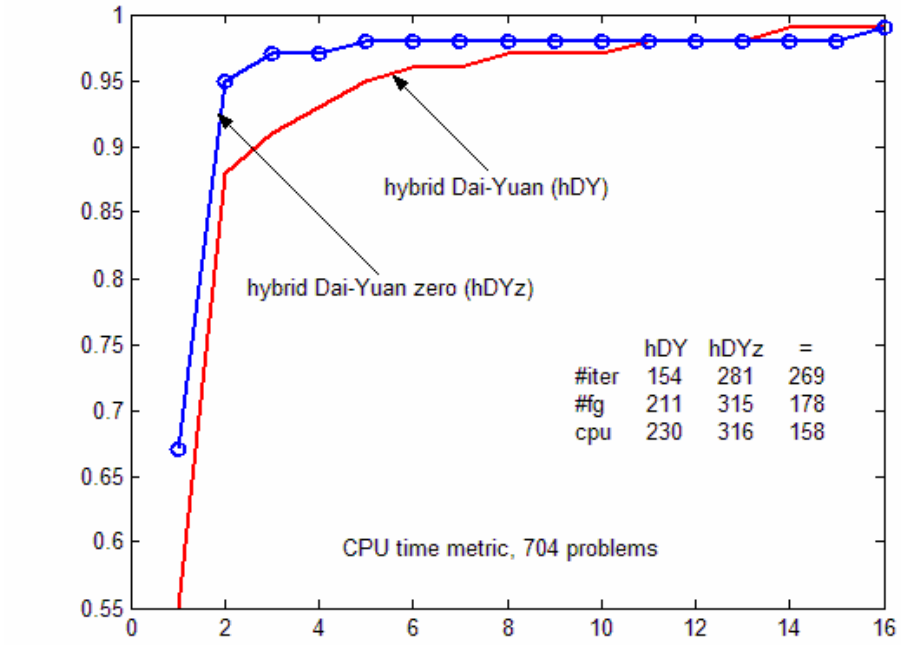

Fig. 3.1. Performance profile of Hybrid Dai-Yuan (hDY) and Hybrid Dai-Yuan zero (hDYz).

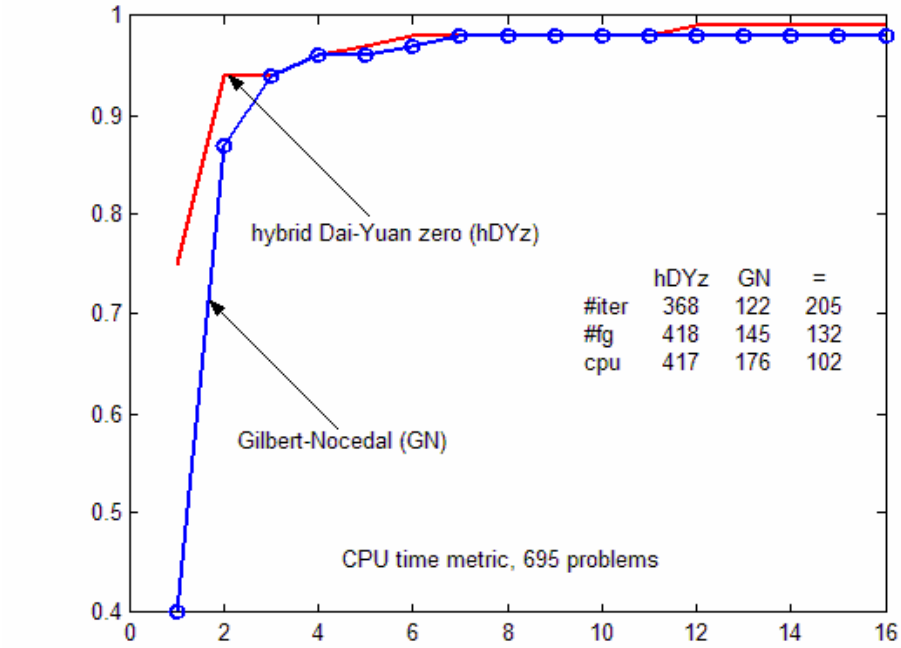

Fig. 3.2. Performance profile of Hybrid Dai-Yuan zero (hDYz) and Gilbert-Nocedal (GN).

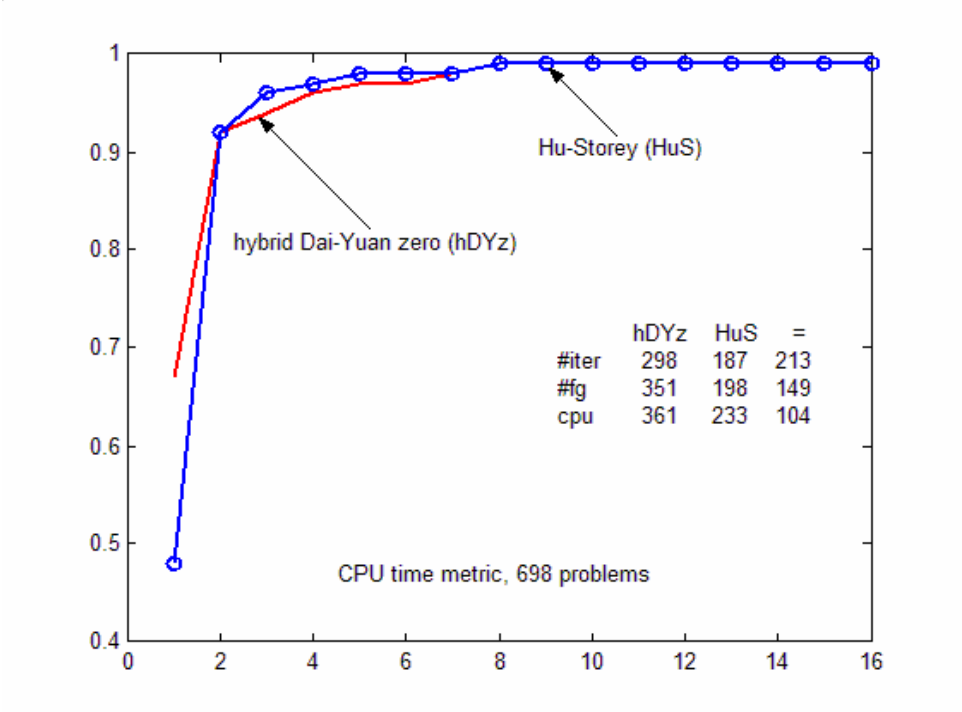

Fig. 3.3. Performance profile of Hybrid Dai-Yuan zero (hDYz) and Hu-Storey (HuS).

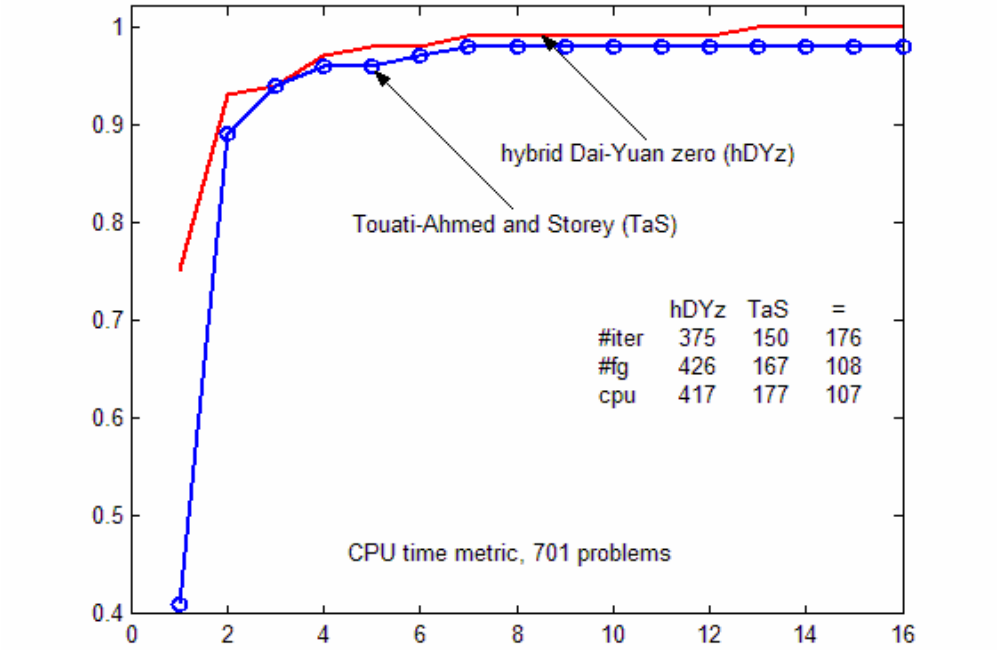

Fig. 3.4. Performance profile of Hybrid Dai-Yuan zero (hDYz) and Touati-Ahmed and Storey (TaS).

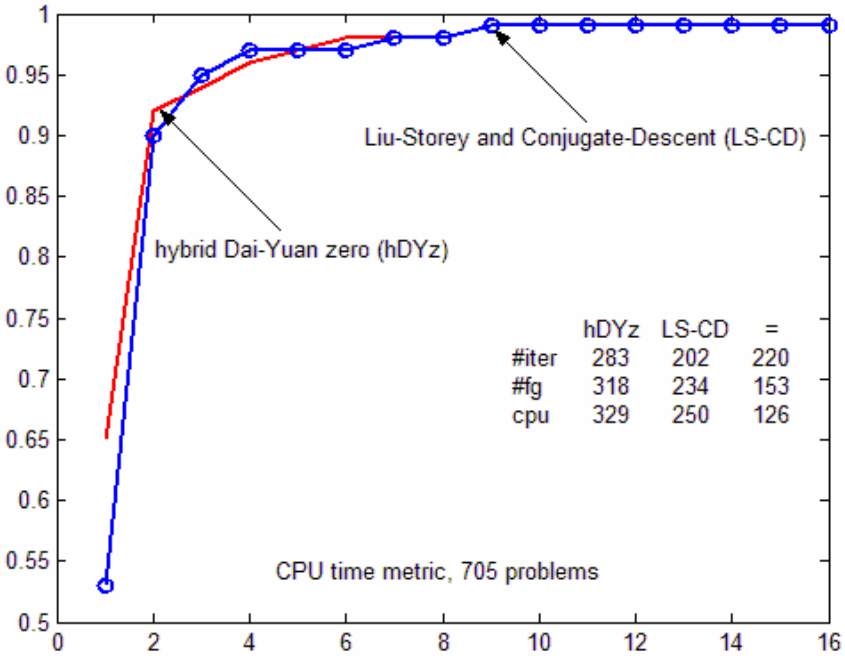

Fig. 3.5. Performance profile of Hybrid Dai-Yuan zero (hDYz) and Liu-Storey and Conjugate-Descent (LS-CD).

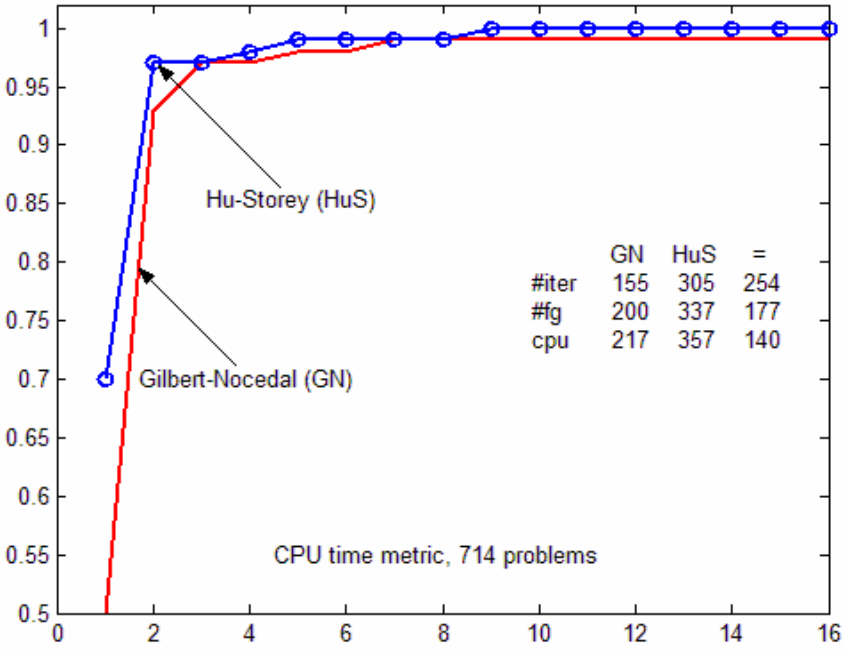

Fig. 3.6. Performance profile of Gilbert-Nocedal (GN) and Hu-Storey (HuS).

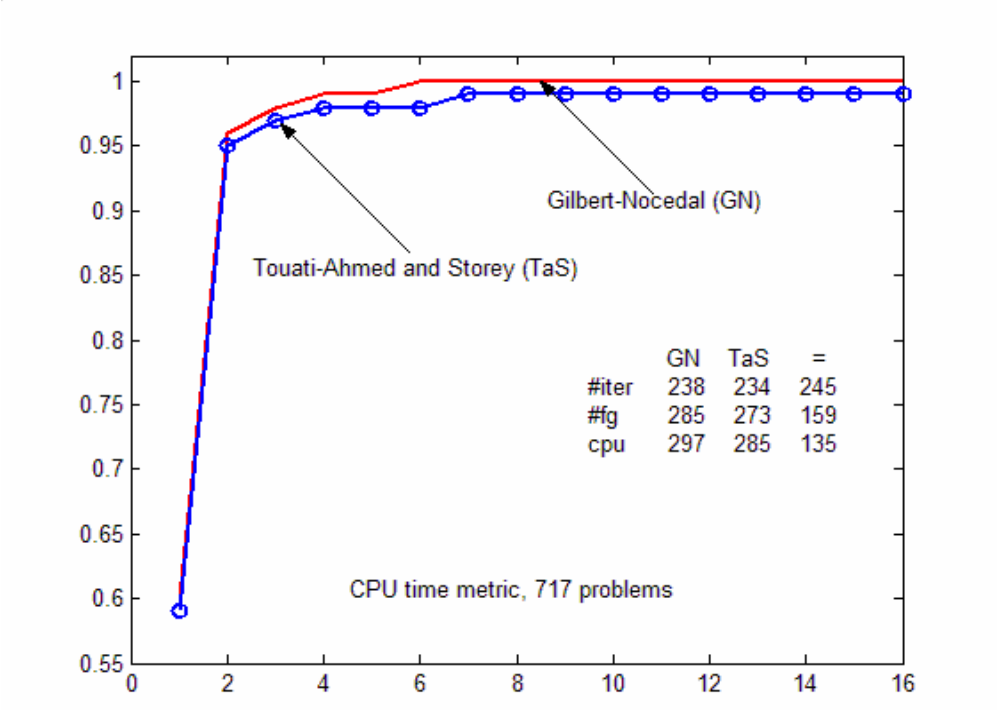

Fig. 3.7. Performance profile of Gilbert-Nocedal (GN) and Touati-Ahmed and Storey (TaS).

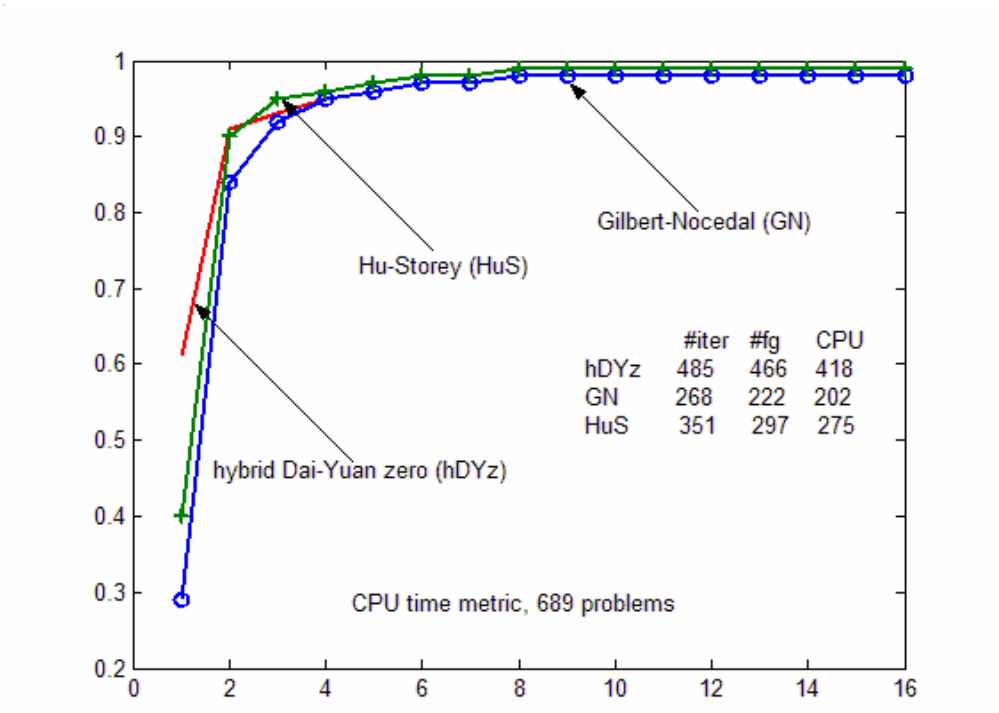

Fig. 3.8. Performance profile of hybrid Dai-Yuan zero (hDYz), Gilbert-Nocedal (GN) and Hu-Storey (HuS).

From these Tables we se that the hybrid Dai-Yuan zero (hDYz) is the best variant of hybrid conjugate gradient algorithms.

4. Performance Profiles of Scaled Conjugate Gradient Algorithms

The scaled conjugate gradient algorithms are defined by the recurrence:

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

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

where the parameter θ_k is the inverse of the Rayleigh quotient :

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

and $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$. The parameter β_k is computed as in Table 4.

Table 4. Scaled conjugate gradient algorithms.

Nr.	File	Formula	Name
1.	Z17	$\beta_k^{BM} = \frac{(\theta y_k - s_k)^T g_{k+1}}{y_k^T s_k}$	Birgin and Martinez, Scaled Perry
2.	Z18	$\beta_k^{BM+} = max\left\{0, \frac{\theta y_k^T g_{k+1}}{y_k^T s_k}\right\} - \frac{s_k^T g_{k+1}}{y_k^T s_k}$	Brigin and Martinez +
3.	Z19	$\beta_k^{sPRP} = \frac{\theta_{k+1} y_k^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k}$	Scaled Polak-Ribiere-Polyak (sPRP)
4.	Z20	$\beta_k^{sFR} = \frac{\theta_{k+1} g_{k+1}^T g_{k+1}}{\alpha_k \theta_k g_k^T g_k}$	Scaled Fletcher-Reeves (sFR)

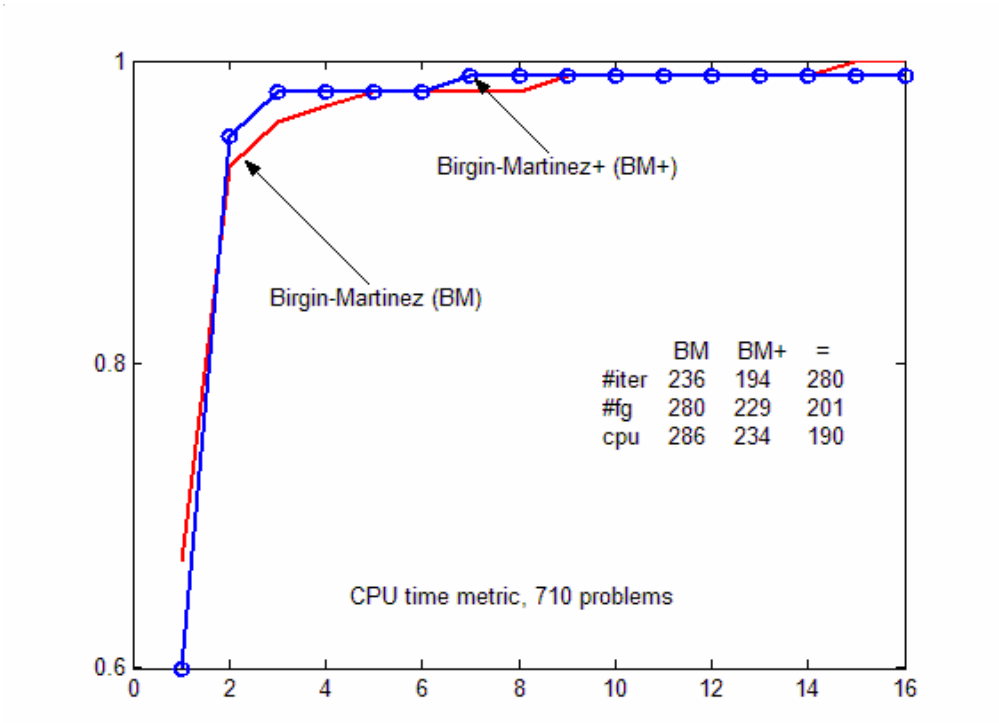

Fig. 4.1. Performance profile of Birgin-Martínez (BM) and Birgin-Martínez+ (BM+).

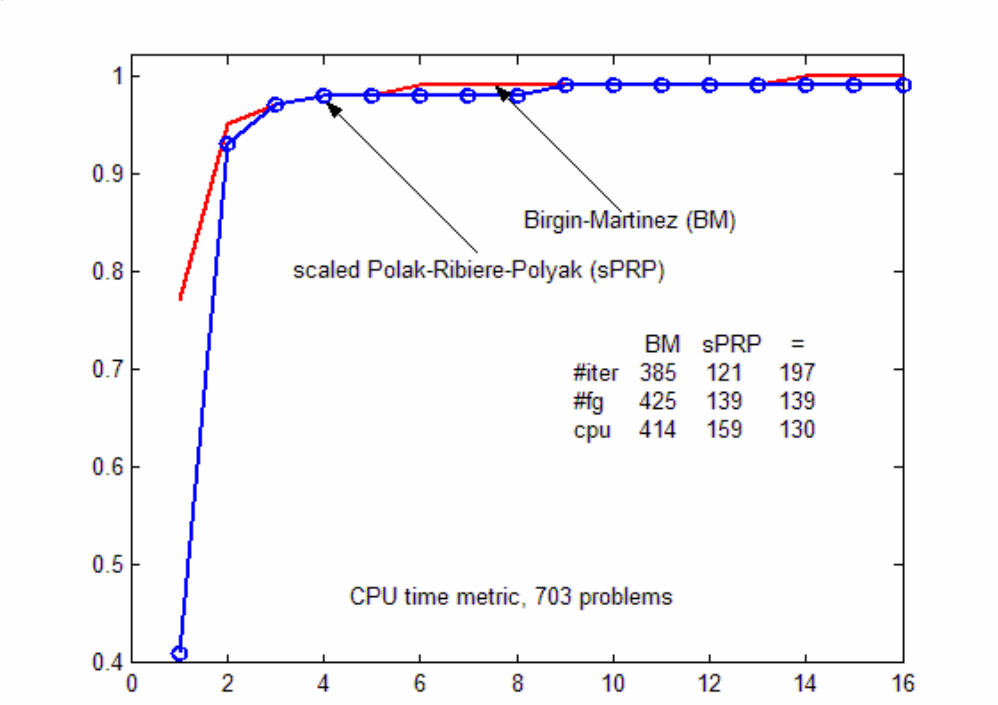

Fig. 4.2. Performance profile of Birgin-Martínez (BM) and scaled Polak-Ribiere-Polyak (sPRP).

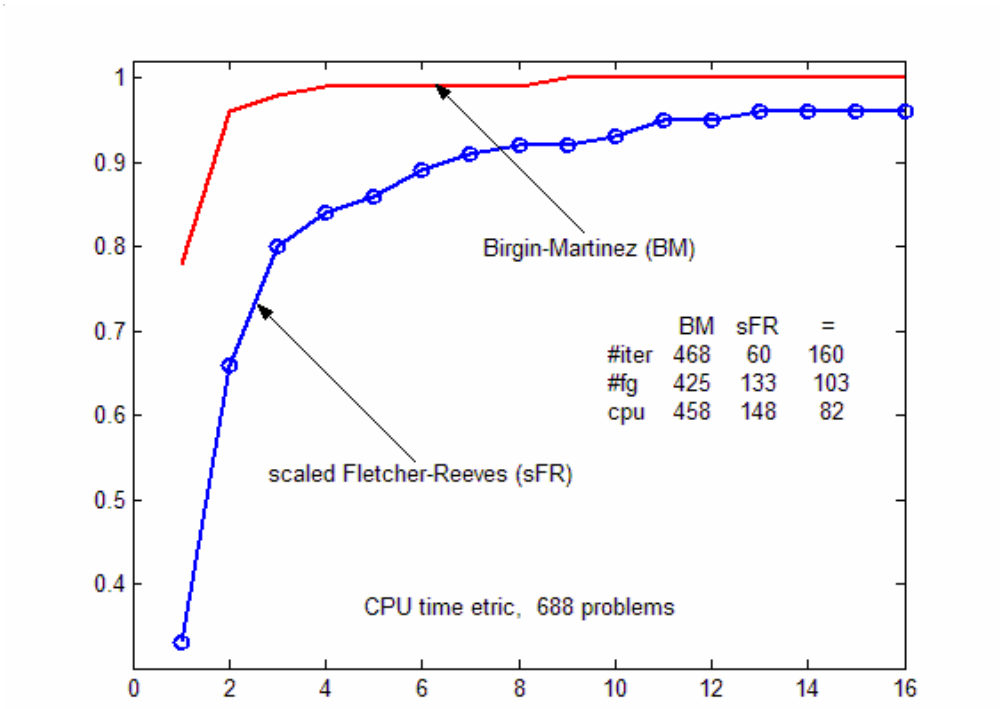

Fig. 4.3. Performance profile of Birgin-Martínez (BM) and scaled Fletcher-Reeves (sFR).

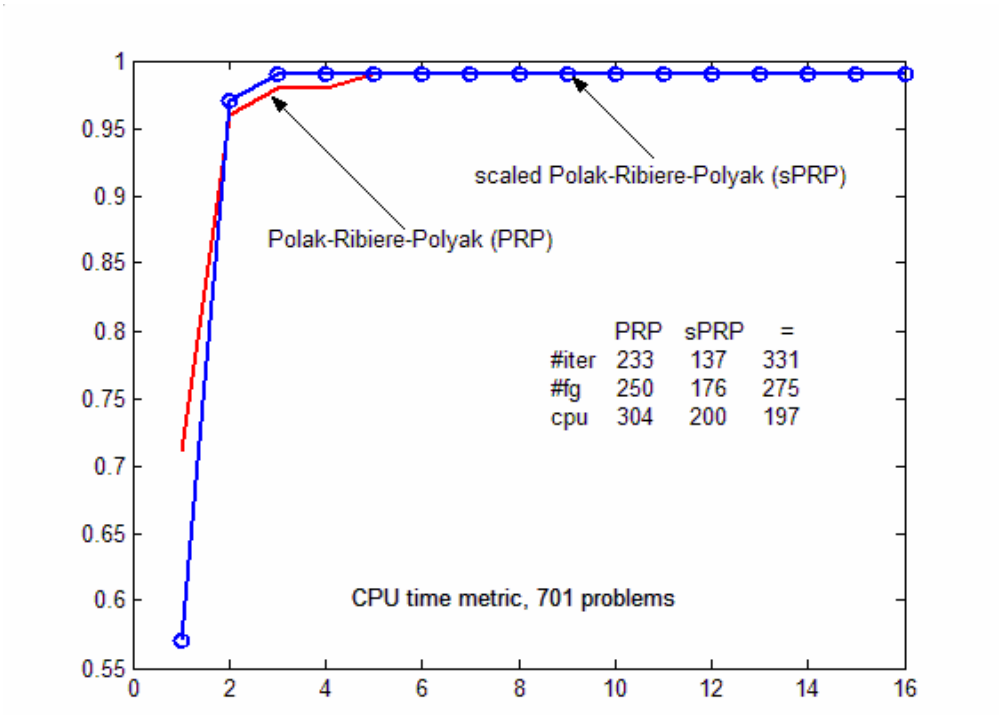

Fig. 4.4. Performance profile of Polak-Ribière-Polyak (PRP) and scaled Polak-Ribière-Polyak (sPRP).

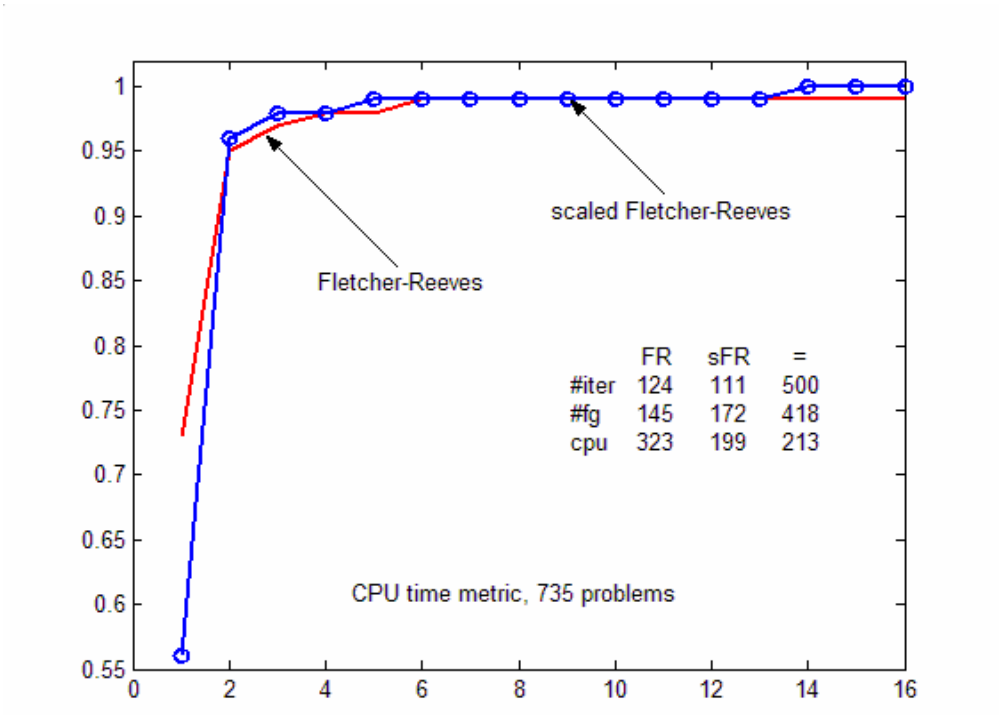

Fig. 4.5. Performance profile of Fletcher-Reeves (FR) and scaled Fletcher-Reeves (sFR).

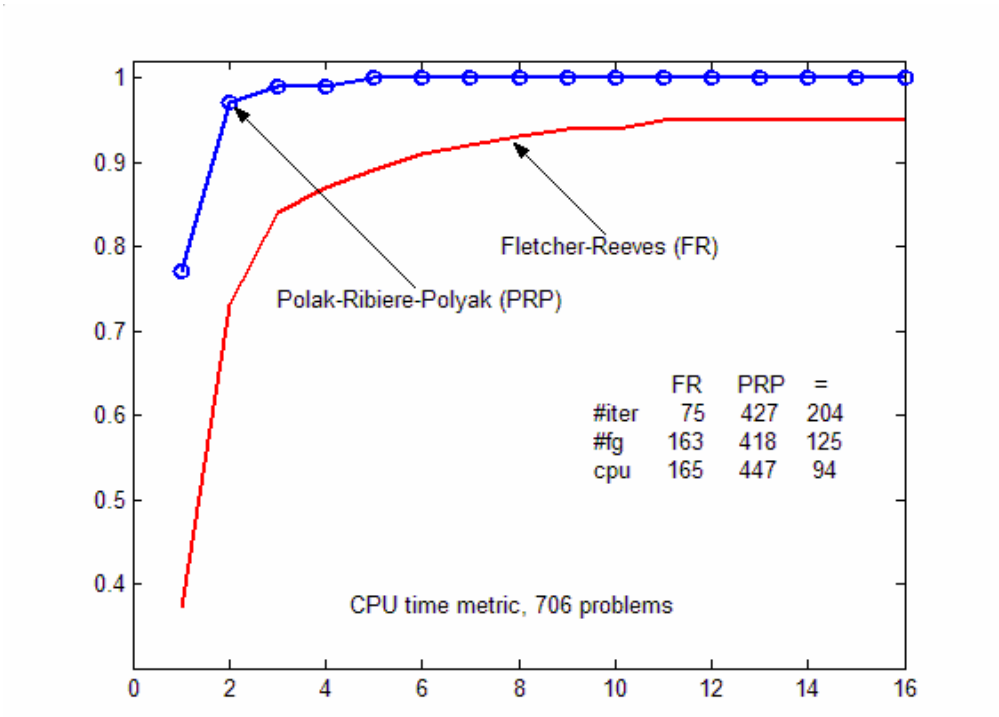

Fig. 4.6. Performance profile of Fletcher-Reeves (FR) and Polak-Ribière-Polyak (PRP).

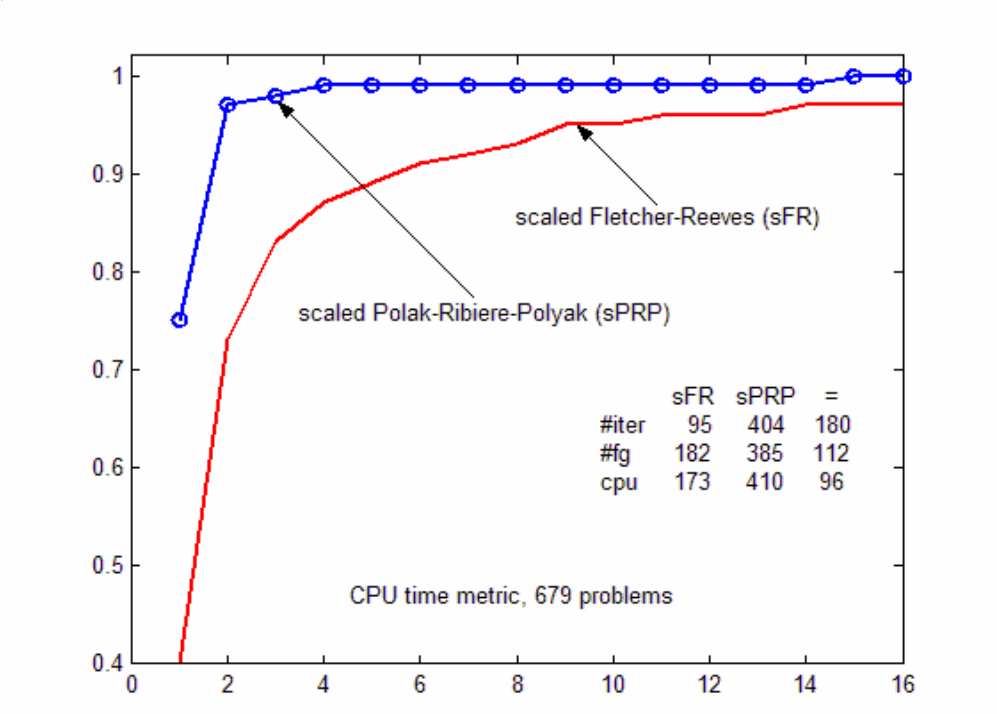

Fig. 4.7. Performance profile of scaled Fletcher-Reeves (sFR) and scaled Polak-Ribière-Polyak (sPRP).

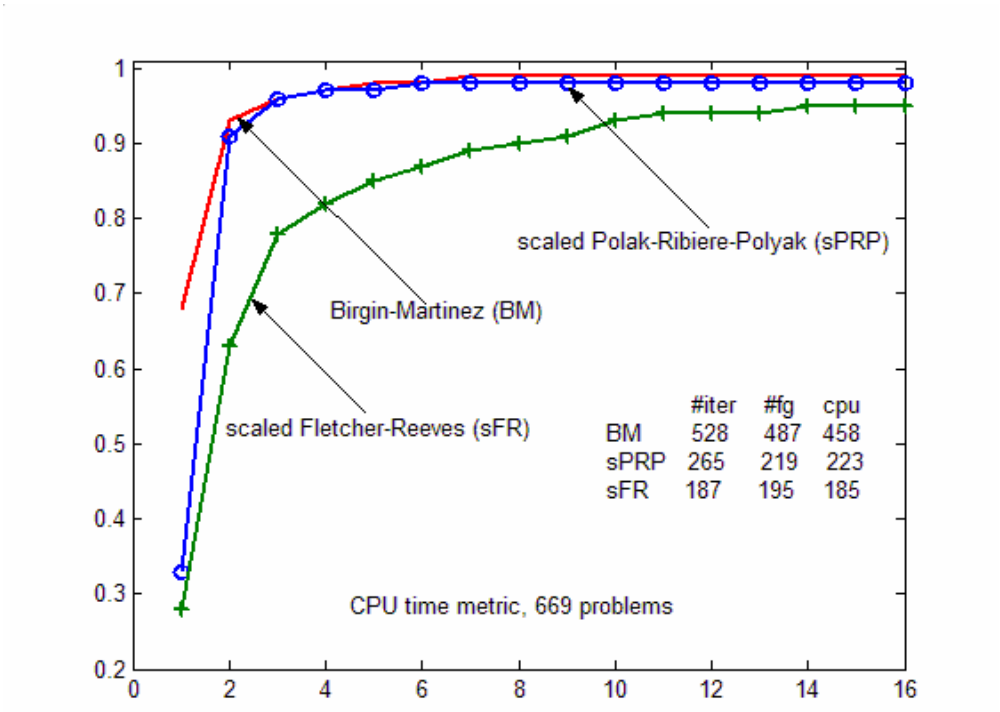

Fig. 4.8. Performance profile of BM, sPRP and sFR.

The Figures above give the computational evidence that Birgin-Martínez (BM) conjugate gradient algorithm is the top performer in this class. The Polak-Ribière-Polyak (PRP) algorithm is way more competitive than Fletcher-Reeves (FR), both in original and scaled variants.

5. Performance Profiles of Parametric Conjugate Gradient Algorithms

As we have already seen in the above Figures (Fig. 2.8, Fig. 3.8 and Fig. 4.8) the PRP+, hDYz and BM are the best conjugate gradient algorithms, subject to the CPU time metric. The hybrid and parametric conjugate gradient methods have been introduced in order to combine the good properties of these methods and to exploit the attractive features of them. The parametric family of conjugate gradient methods was mainly designed to integrate the conjugate gradient algorithms in the same manner as the quasi-Newton methods have been combined together by introducing parameters.

In CGALL only two parametric conjugate gradient algorithms are implemented as in Table 5. We selected only the Dai and Liao parametric conjugate gradient method since it has only one parameter t, which can by very easy modified. However we implemented in CGALL the DL and DL+ algorithms with $t = 1$. Different values for t give different computational results, and for general functions it is very difficult to predict an advantageous value for t.

Table 5. Parametric conjugate gradient algorithms.

1.	Z8	$\beta_k^{DL} = \frac{g_{k+1}^T (y_k - t s_k)}{y_k^T s_k}$	Dai and Liao (DL)
2.	Z9	$\beta_k^{DL+} = max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\} - t\frac{s_k^T g_{k+1}}{y_k^T s_k}$	Dai and Liao + (DL+)

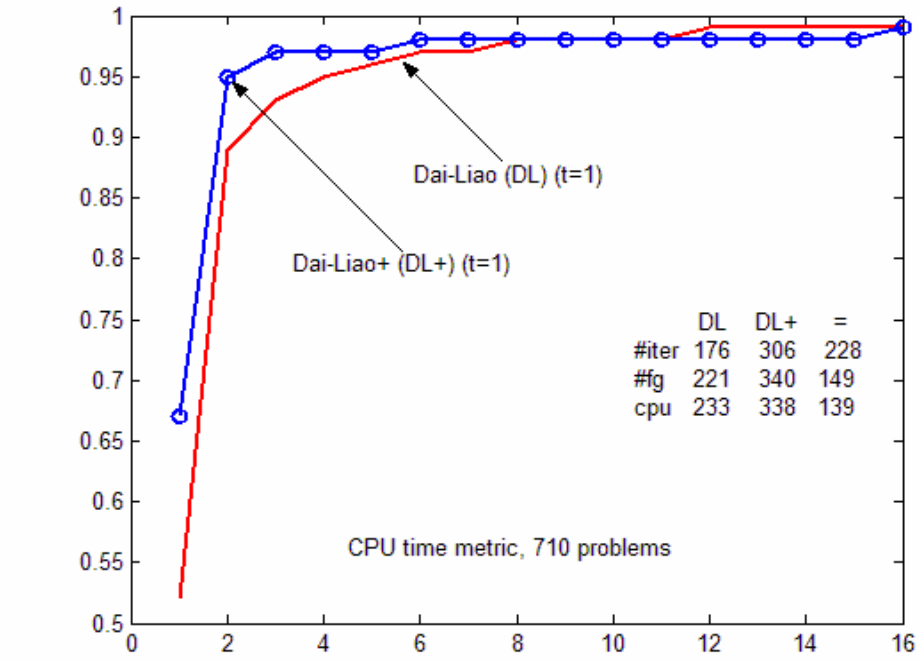

Fig. 5.1. Performance profile of Dai-Liao (DL) (t=1) and Dai-Liao+ (DL+) (t=1).

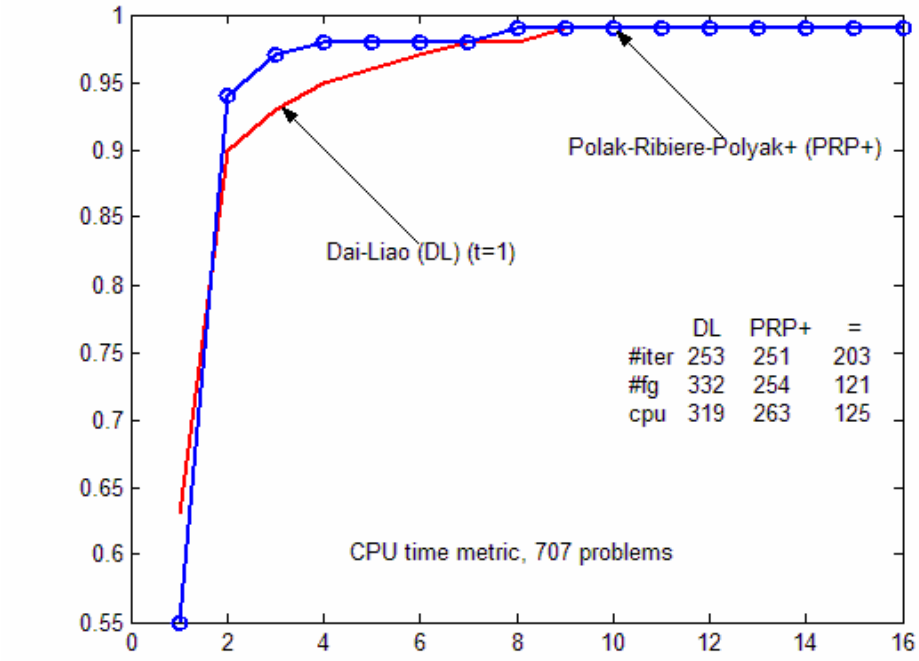

Fig. 5.2. Performance profile of Dai-Liao (DL) (t=1) and Polak-Ribière-Polyak+ (PRP+).

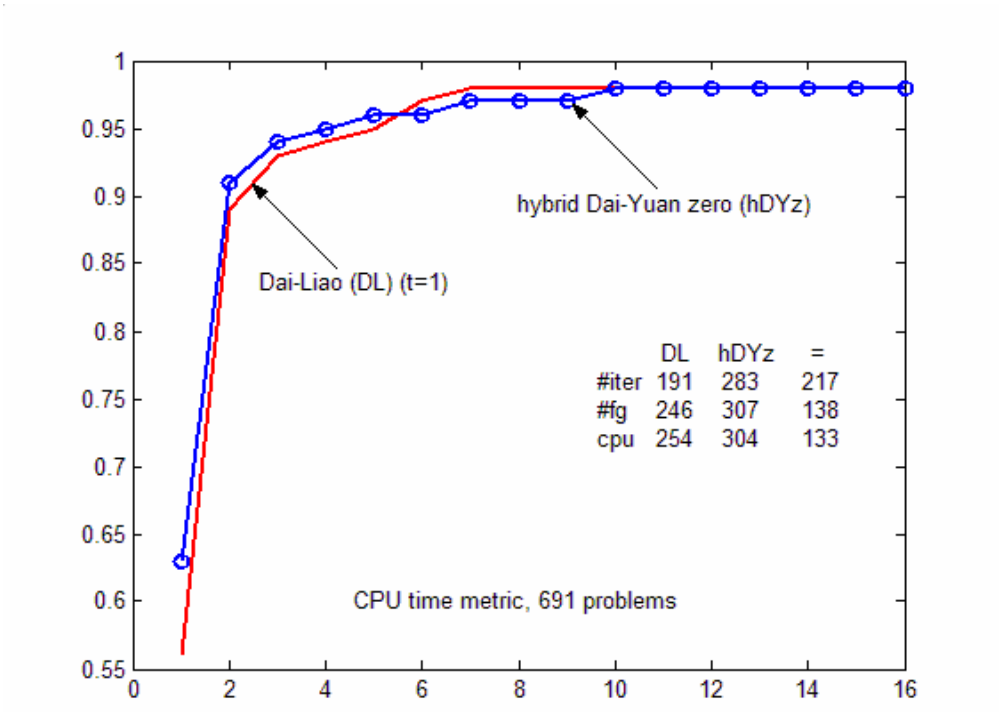

Fig. 5.3. Performance profile of Dai-Liao (DL) (t=1) and hybrid Dai-Yuan zero (hDYz).

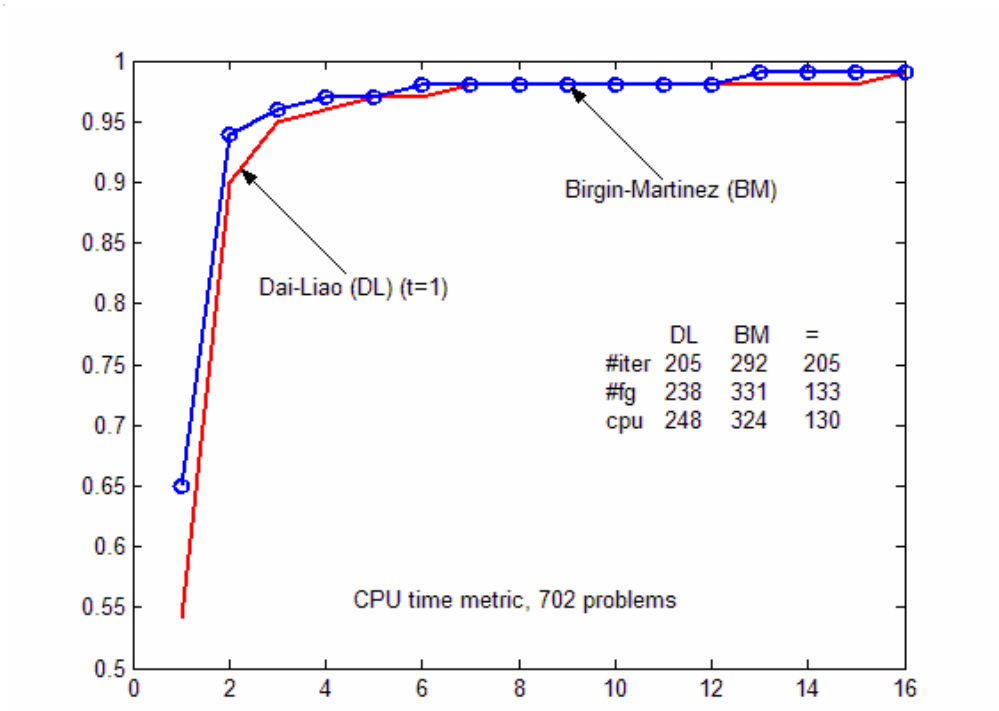

Fig. 5.4. Performance profile of Dai-Liao (DL) (t=1) and Birgin-Martínez (BM).

6. Performance Profiles of Modified Conjugate Gradient Algorithms

In this section we present some conjugate gradient algorithms obtained by modification of classical conjugate gradient algorithms in order to satisfy the conjugacy condition as well as the sufficient descent condition. These algorithms are described in Table 6.

Table 6. Modified conjugate gradient algorithms.

1.	Z10	$\beta_k^{ASDC} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k} - \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{(y_k^T s_k)^2}$	Andrei Sufficient Descent Condition. Please, see the paper for AML, AML5382[2]. ($y_k^T \bar{Q}_{k+1} = 0$)
2.	Z21	$\beta_k^A = \frac{1}{y_k^T s_k}\left(y_k^T g_{k+1} - \frac{(y_k^T y_k)(s_k^T g_{k+1})}{g_k^T g_k} \right)$	Andrei (*Sufficient descent condition from PRP*) Please, see Remark 8.3.3 and formula (8.3.130) in the book: Neculai Andrei, "*Criticism of the Unconstrained Optimization Algorithms Reasoning*".
3.	Z22	$\beta_k^{ACGSD} = \frac{y_k^T g_{k+1}}{y_k^T s_k} - \frac{(y_k^T g_{k+1})(s_k^T g_{k+1})}{(y_k^T s_k)^2}$	Andrei (*Sufficient descent condition from DY*) (ACGSD) Please see paper for SIOPT, Manuscris #067836[3]. ($y_k^T \bar{Q}_{k+1} g_{k+1} = 0$)
4.	Z23	$\beta_k^{ACGSDz} = max\left\{0, \frac{y_k^T g_{k+1}}{y_k^T s_k}\right\}\left(1 - \frac{s_k^T g_{k+1}}{y_k^T s_k}\right)$	Andrei (*Sufficient descent condition from DY zero*) (ACGSDz)

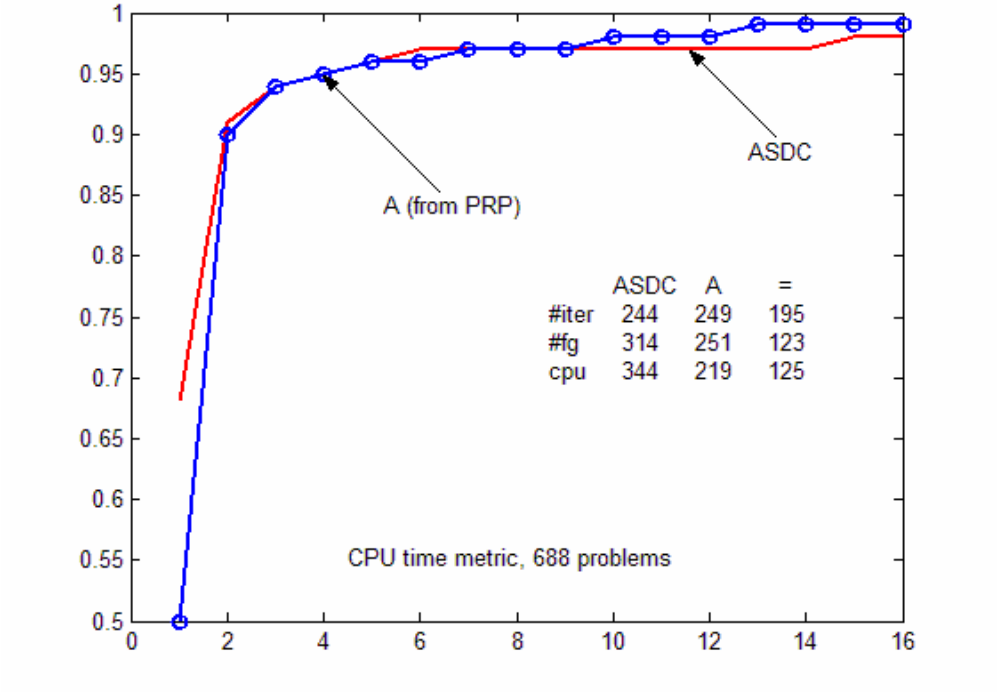

Fig. 6.1. Performance profile of ASDC and A (from PRP).

[2] Neculai Andrei, *Dai-Yuan conjugate gradient algorithm with sufficient descent and conjugacy conditions for unconstrained optimization.* Submitted AML, paper No. 5382, November 20, 2006.

[3] Neculai Andrei, *Another nonlinear conjugate gradient algorithm with conjugacy and sufficient descent conditions for unconstrained optimization.* Submitted SIOPT, paper No. 067836, December 22, 2006.

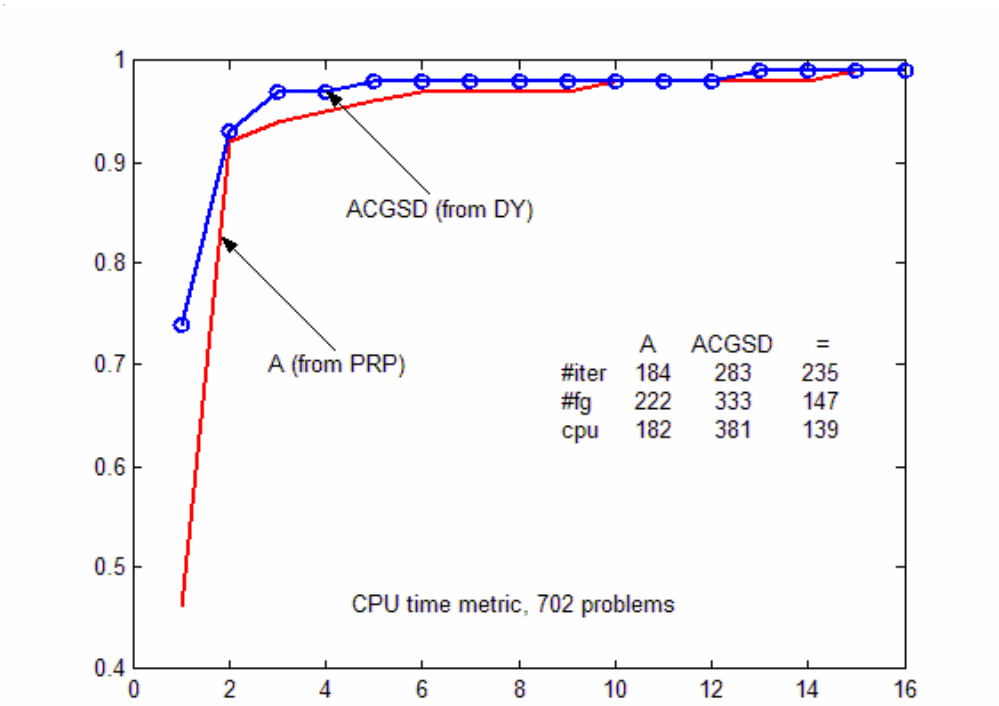

Fig. 6.2. Performance profile of A (from PRP) and ACGSD (from DY).

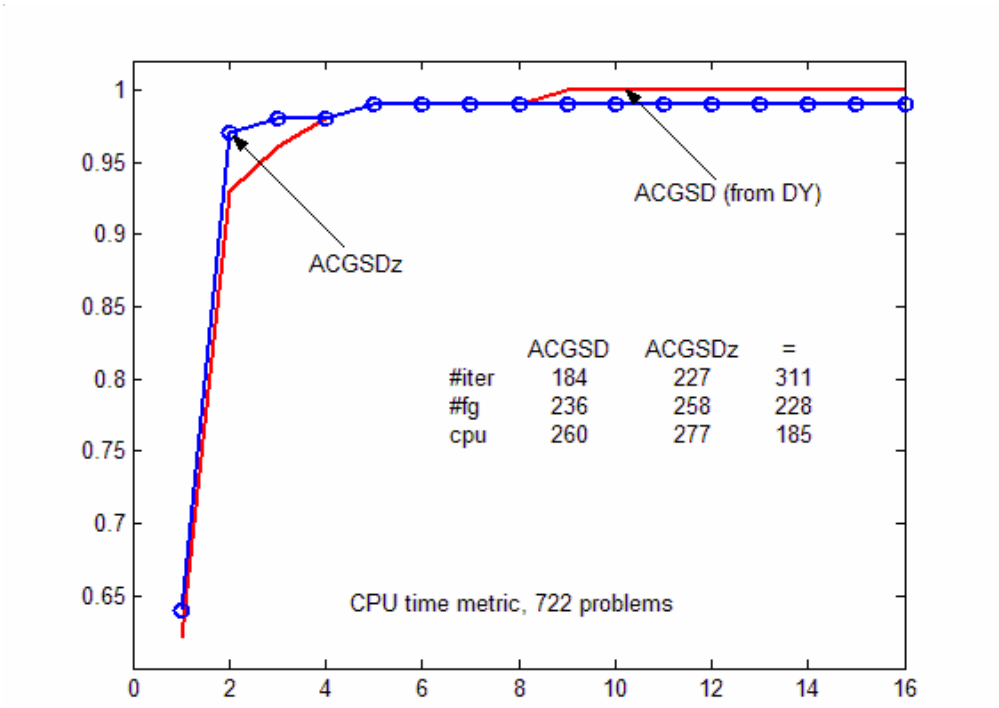

Fig. 6.3. Performance profile of ACGSD (from DY) and ACGSDz.

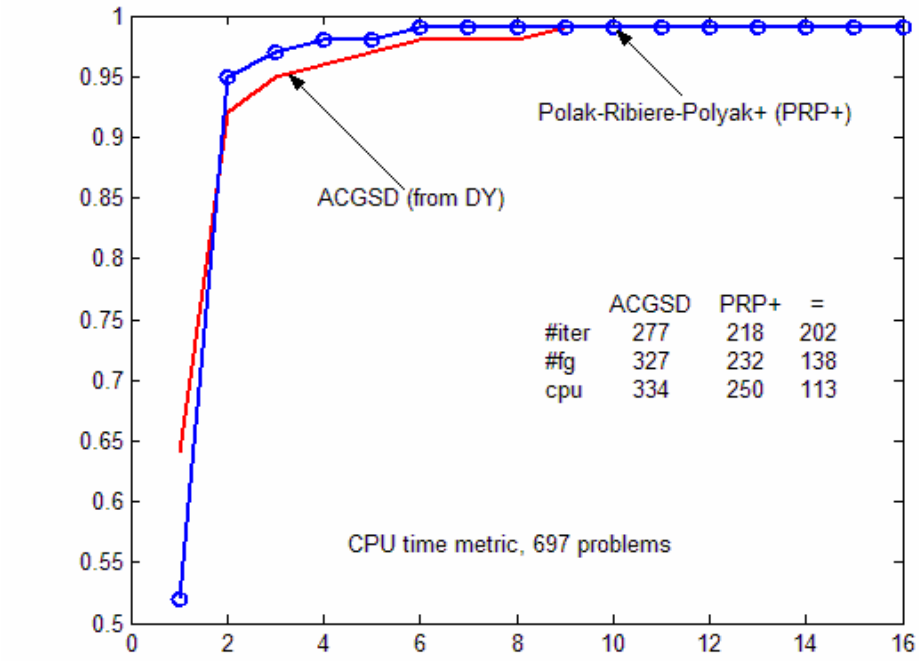

Fig. 6.4. Performance profile of ACGSD (from DY) and Polak-Ribiere-Polak+ (PRP+).

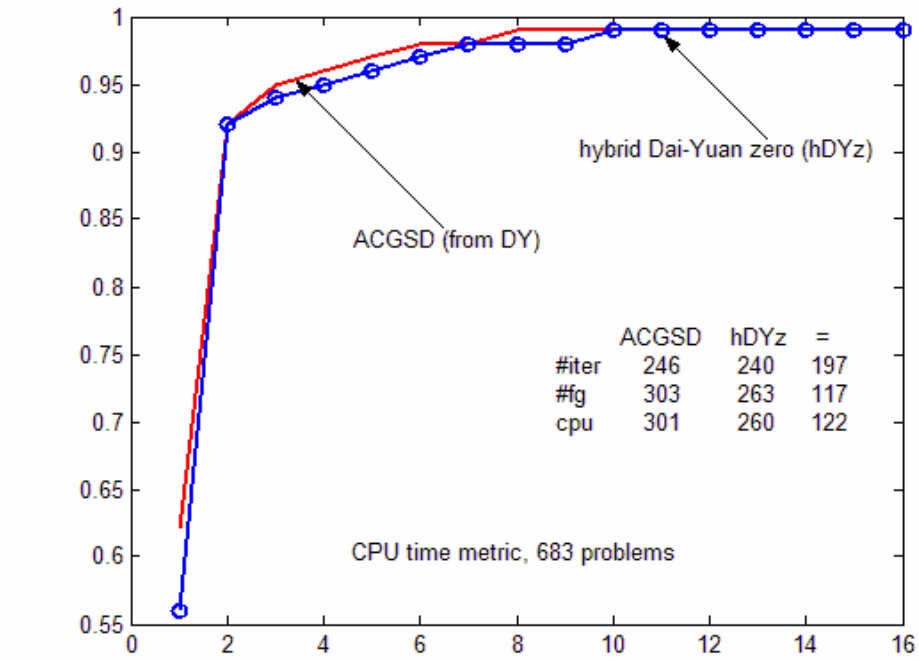

Fig. 6.5. Performance profile of ACGSD (from DY) and Hybrid Dai-Yuan zero (hDYz).

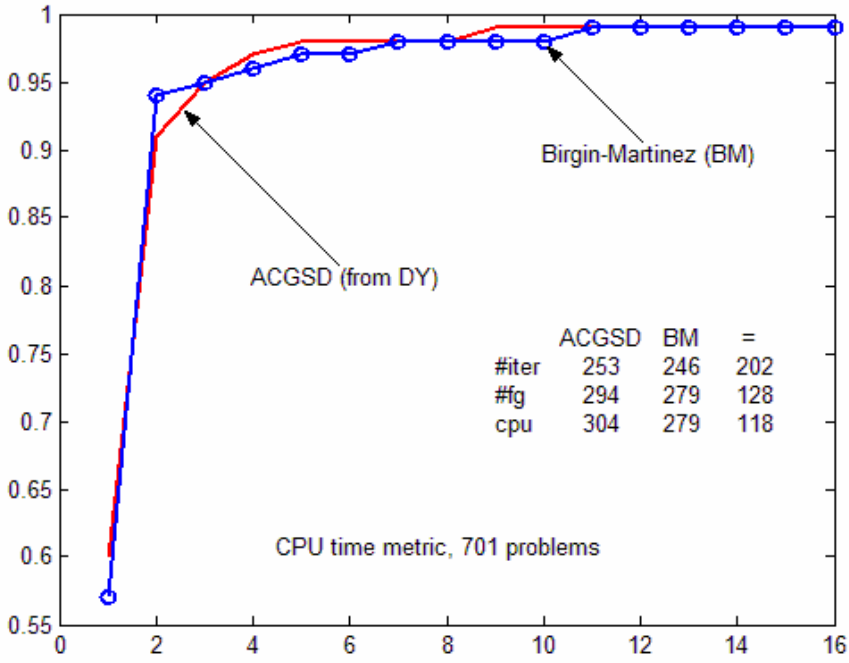

Fig. 6.6. Performance profile of ACGSD (from DY) and Birgin-Martínez (BM).

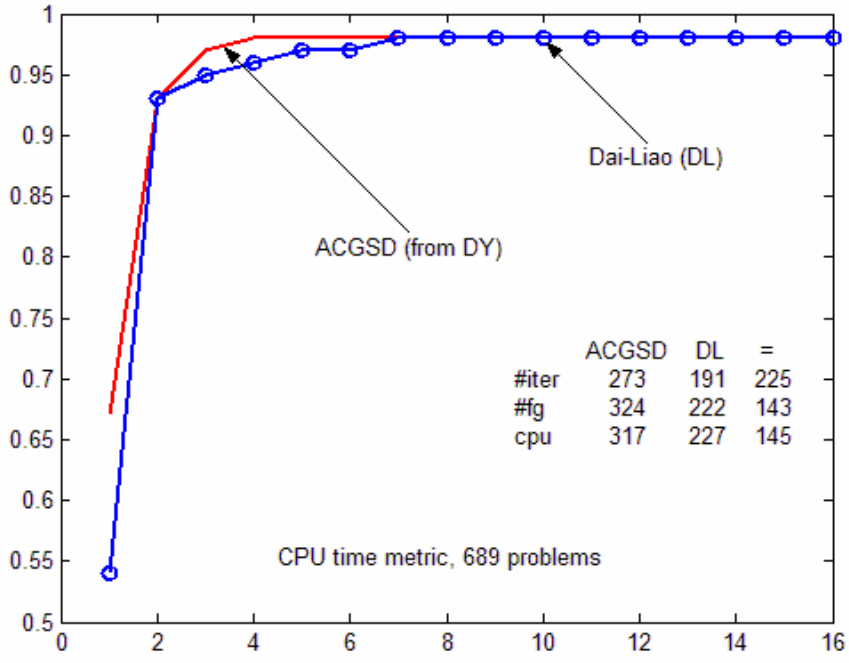

Fig. 6.7. Performance profile of ACGSD (from DY) and Dai-Liao (DL).

7. Conclusion

We have presented the performance profile of 23 conjugate gradient algorithms implemented in CGALL package. All algorithms are implemented in the same manner, and use the Wolfe line search conditions and the same stopping criterion $\|g_k\|_\infty \leq 10^{-6}$. From the above Figures it follows that the best variants of conjugate gradient algorithms are PRP+, hDYz, BM (θ_{k+1} spectral), DL and ACGSD (from DY).

Figure 7.1 shows the performance profile of the most competitive conjugate gradient algorithms implemented in CGALL package. We see that ACGSD (from DY) is the most robust conjugate gradient algorithm, at least for this set of test functions.

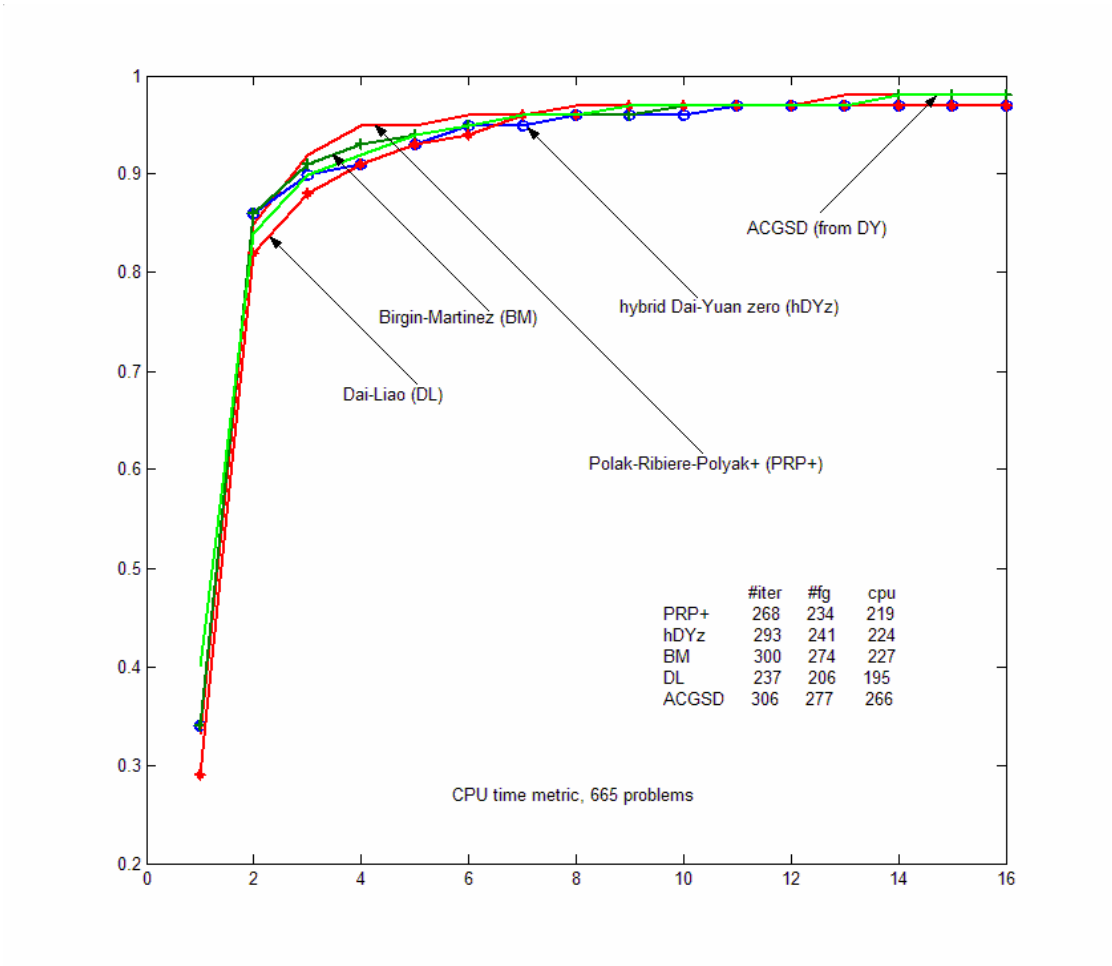

Fig. 7.1. Performance profile of PRP+, hDYz, BM, DL, ACGSD (from DY).

March 1, 2007