Variants of the Dai-Liao conjugate gradient method for unconstrained optimization

Neculai Andrei[1]
Center for Advanced Modeling and Optimization,
Academy of Romanian Scientists,
54, Splaiul Independenței, Sector 5,
Bucharest, ROMANIA
E-mail: neculaiandrei70@gmail.com

Technical Report 3/2021
January 19, 2021

Abstract. This paper presents the performances of the Dai-Liao conjugate gradient algorithm subject to different values of the scalar t in the definition of the conjugate gradient parameter β_k^{DL}. Since the Dai-Liao algorithm is a modification of the Hestenes-Stiefel, the comparison between these algorithms are presented using a collection of 800 unconstrained optimization test functions.

1. Introduction

For solving the large-scale unconstrained optimization problems

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

where $f:\mathbb{R}^n \to \mathbb{R}$ is a continuous differentiable function and n is enough large the *conjugate gradient method* is recommended as one of the most efficiency. In this method the updating of the variables is as

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0,1,\ldots, \tag{2}$$

the scalar $\alpha_k > 0$ being the stepsize and d_k the search direction updated as

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad k = 0,1,\ldots, \tag{3}$$

In (2) for $k = 0$ x_0 is a known initial point around which the minimum point x^* of function f is sought and in (3) $d_0 = -g_0$. Here, $g_k = \nabla f(x_k)$, is the gradient of f evaluated in x_k.
Details on conjugate gradient methods for unconstrained optimization including both their convergence and numerical performances are given in [1].
The conjugate gradient method of Dai and Liao [2] is defined by (2) where the search direction is computed as in (3) with

[1] Academy of Romanian Scientists, Splaiul Independenţei nr. 54, Sector 5, Bucharest, Romania

$$\beta_k^{DL} = \frac{y_k^T g_{k+1}}{y_k^T d_k} - t\frac{s_k^T g_{k+1}}{y_k^T d_k}, \tag{4}$$

where $t > 0$ is a scalar. This method was introduced by Dai and Liao using the modified conjgacy condition. $y_k^T d_{k+1} = -t(s_k^T g_{k+1})$. Observe that the parameter β_k^{DL} in the Dai-Liao conjugate gradient method is a modification of the Hestenes-Stiefel conjugate gradient parameter

$$\beta_k^{HS} = \frac{y_k^T g_{k+1}}{y_k^T d_k}. \tag{5}$$

Plenty of methods for selection of the scalar t in parameter β_k^{DL} are known in literature (see [3-15]). Selection of the scalar t in β_k^{DL} is an open problem [3] and until now no optimal value of it is known. Therefore, a good idea is to see the performances of the Dai-Liao algorithm subject to different values of the scalar t suggested in literature. Since the Dai-Liao algorithm is a modification of the Hestenes-Stiefel one, comparisons among these algorithms are presented for solving a collection of 800 unconstrained optimization test problems with the number of variables in the range [1000, 10000].

2. The context of the numerical experiments

The numerical experiments have bee considered in the following context. All algorithms have been coded in double precision Fortran and compiled with f77 (default compiler settings) and run on an Intel Pentium 4, 1.8 GHz workstation. The test functions are from the UOP collection [16], which includes 80 functions. For each test function ten numerical experiments with the number of variables $n = 1000, 2000, ..., 10000$ have been considered, thus obtaining a number of 800 problems.

The algorithms compared in these numerical experiments find local solutions. Therefore, the comparisons of the 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 $i = 1, \ldots, 800,$ respectively. We say that, in the particular problem $i,$ the performance of ALG1 was better than the performance of ALG2 if

$$\left| f_i^{ALG1} - f_i^{ALG2} \right| < 10^{-3} \tag{6}$$

and if the number of iterations (#iter), or the number of function-gradient evaluations (#fg), 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 iterations are stopped if the inequality $\|g_k\|_\infty \le 10^{-6}$ is satisfied, where $\|.\|_\infty$ is the maximum absolute component of a vector. All algorithms implement the standard Wolfe line search,

$$f(x_k + \alpha_k d_k) \le f(x_k) + \rho \alpha_k d_k^T g_k,$$
$$\nabla f(x_k + \alpha_k d_k)^T d_k \ge \sigma d_k^T g_k,$$

where $\rho = 0.0001$ and $\sigma = 0.8.$ The maximum number of iterations was limited to 2000. To compare the performances of algorithms, the Dolan and Moré [17] performance profiles are used.

3. Numerical results and comparisons

1) In the first numerical experiment we compare the performances of the Hestenes-Stiefel algorithm versus the performances of the Dai-Liao in which the parameter t is computed as

$$t = 1 \tag{7}$$

Figure 1 presents the Dolan-Moré performances profiles of these algorithms, where DL1 is the Dai-Liao algorithm (4) with $t = 1$ and HS is the Hestenes-Stiefel algorithm given by (5).

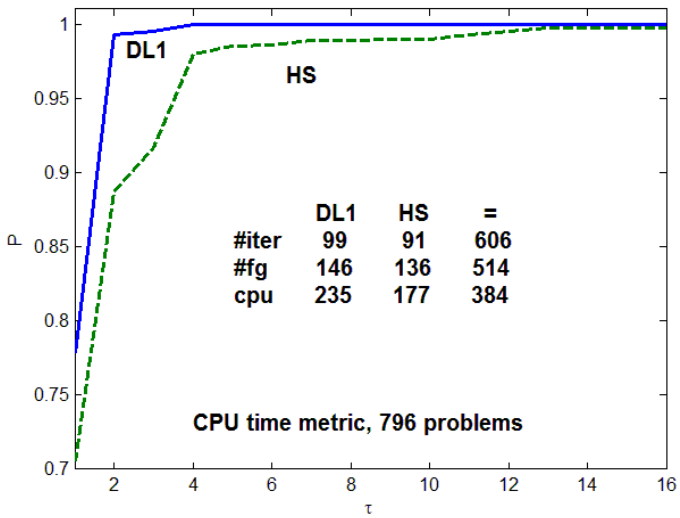

Fig. 1. Performances profiles of DL1 (Dai-Liao with $t = 1$) versus HS (Hestenes-Stiefel)

Observe that this simple modification of the Hestenes-Stiefel algorithm leads us to a conjuagte gradient algorithm more efficient. The tables inside the plot show the performances of the algorithms subject to the number of iterations (*#iter*), the number of function and its gradient evaluations (*#fg*) and subject to the CPU time metric (*cpu*) in seconds. When comparing DL1 versus HS (see Figure 1) subject to the number of iterations, we see that DL1 was better in 99 problems (i.e. it achieved the minimum number of iterations in 99 problems). HS was better in 91 problems and they achieved the same number of iterations in 606 problems, etc. Out of 800 problems considered in this numerical experiment, only for 796 problems does the criterion (6) hold. In the following numerical experiments let us see the performances of Dai-Liao algorithm with different values of the scalar t.

2) In the second set of numerical experiments the scalar t in (4) is computed as

$$t = \frac{y_k^T s_k}{\|s_k\|^2}. \tag{8}$$

Observe that (8) is exactly the value given by Oren and Luenberger [18]. Besides, it is worth mentioning that (8) is obtained by clustering the eigenvalues of the self-scaling BFGS matrix

$$H_{k+1} = \frac{1}{t}\left(I - \frac{s_k y_k^T + y_k s_k^T}{y_k^T s_k} \right) + \left(1 + \frac{1}{t} \frac{\|y_k\|^2}{y_k^T s_k} \right) \frac{s_k s_k^T}{y_k^T s_k}, \tag{9}$$

using the determinant of this matrix (see [1], pp. 291). Figure 2 presents the performances of these algorithms.

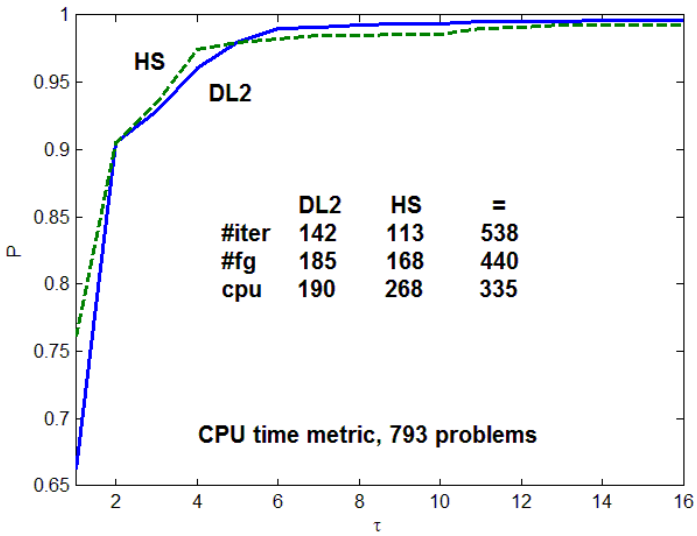

Fig. 2. Performances profiles of DL2 (Dai-Liao with t given by (8)) versus HS (Hestenes-Stiefel)

Figure 2 shows that subject to CPU time metric, HS algorithm is more efficient than DL2. On the other hand, DL2 is slightly more robust. Anyway, both these algorithms have similar performances. From Figure 2 we see that DL2 was fastest in solving 190 problems, but HS was fastest in 268 problems. Out of 800 problem only for 793 does the criterion (6) hold.

3) In the third set of numerical experiments we compare the Dai-Liao algorithm with

$$t = \frac{\|y_k\|^2}{y_k^T s_k} \tag{10}$$

versus the Hestenes-Stiefel algorithm. The value of t in (10) was suggested by Oren and Spedicato [19]. Figure 3 presents the performances of these algorithms. From Figure 3 we see that DL3 is more robust than HS.

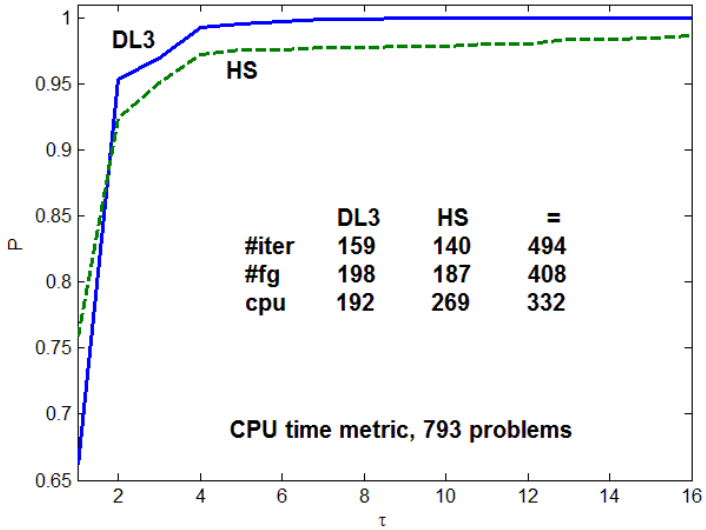

Fig. 3. Performances profiles of DL3 (Dai-Liao with t given by (10)) versus HS (Hestenes-Stiefel)

4) In the forth set of numerical experiments in the Dai-Liao algorithm consider

$$t = \min\left\{1, \frac{y_k^T s_k}{\|s_k\|^2}\right\}. \tag{11}$$

This value has been suggested by Al-Baali [20]. Figure 4 shows the performances of Dai-Liao algorithm with t given by (11) versus those of Hestenes-Stiefel.

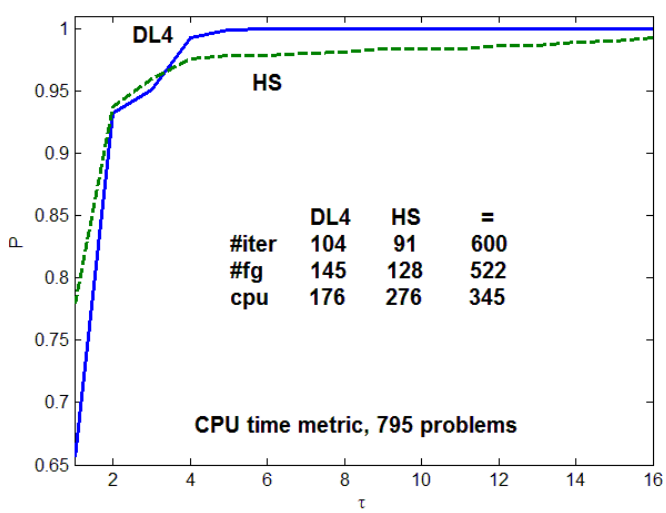

Fig. 4. Performances profiles of DL4 (Dai-Liao with t given by (11)) versus HS (Hestenes-Stiefel)

5) In this set of numerical experiments let us consider

$$t = \min\left\{1, \frac{\|y_k\|^2}{y_k^T s_k}\right\} \tag{12}$$

in the Dai-Liao conjugate gradient parameter (4). This value of t also has been suggested by AlBaali [20]. Figure 5 shows the performances of this algorithm versus Hestenes-Stifel.

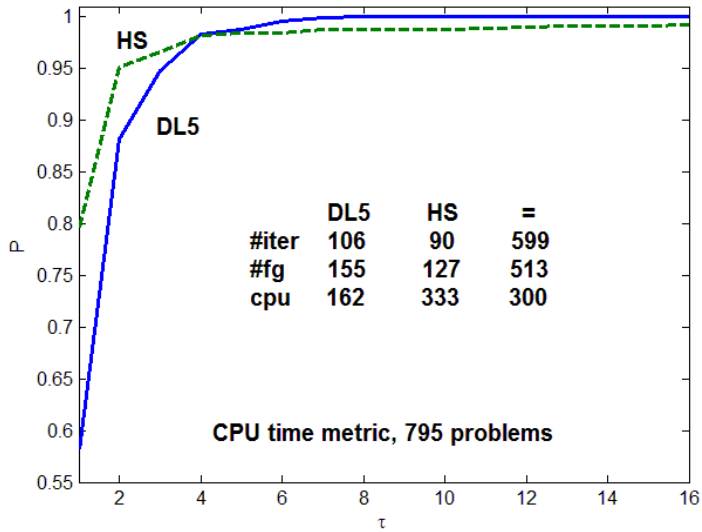

Fig. 5. Performances profiles of DL5 (Dai-Liao with t given by (12)) versus HS (Hestenes-Stiefel)

6) Now, let us consider in (4)

$$t = \left(2 - \frac{\|y_k\|^2 \|s_k\|^2}{(y_k^T s_k)^2}\right) \frac{y_k^T s_k}{\|s_k\|^2}. \tag{13}$$

This value of t is obtained by clustering the eigenvalues of the self-scaling memoryless BFGS matrix (9) using the trace of this matrix (se [1], pp.293). Figure 6 shows the performance profiles of these algorithms.

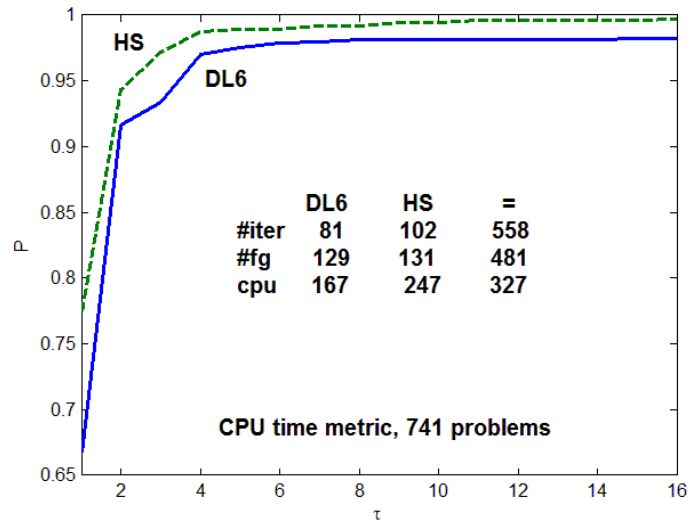

Fig. 6. Performances profiles of DL6 (Dai-Liao with t given by (13)) versus HS (Hestenes-Stiefel)

7) In this set of numerical experiments consider

$$t = \frac{n-2}{n-1} + \frac{1}{n-1}\frac{\|y_k\|^2\|s_k\|^2}{(y_k^T s_k)^2}. \tag{14}$$

This value of t is obtained by minimizing the measure function $\varphi(H_{k+1})$ introduced by Byrd and Nocedal [21], where H_{k+1} is given by (9) (see [1], pp.296). Figure 7 presents the performance profiles of these algorithms.

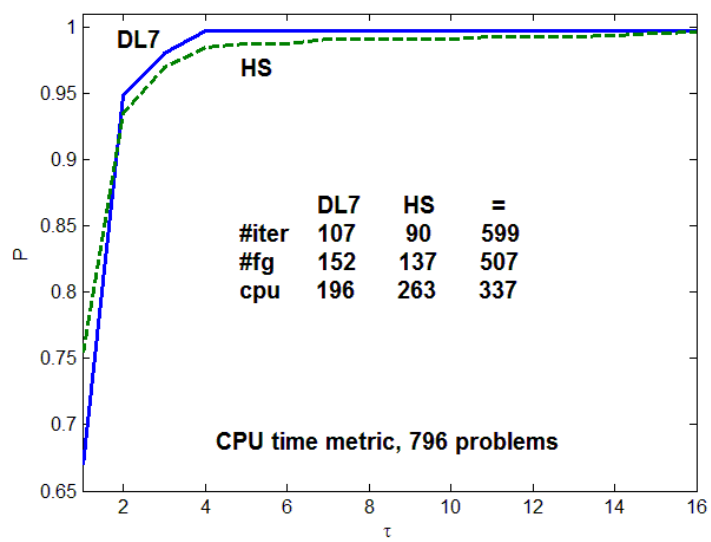

Fig. 7. Performances profiles of DL7 (Dai-Liao with t given by (14)) versus HS (Hestenes-Stiefel)

8) In this set of numerical experiments consider the value of the scalar t given by Aminifard and Babaie-Kafaki (see [8]):

$$t = \begin{cases} \max\{t^-, t^+, \theta \dfrac{\|y_k\|^2}{y_k^T s_k}\}, & a_k \neq 0,\ s_k^T g_{k+1} \neq 0, \\ \theta \dfrac{\|y_k\|^2}{y_k^T s_k}, & \text{otherwise}, \end{cases} \tag{15}$$

where

$$a_k = \frac{y_k^T s_k}{\|s_k\|^2} - \frac{y_k^T g_{k+1}}{s_k^T g_{k+1}},$$

$$t^{\pm} = \frac{\|y_k\|^2 - a_k^2 \|s_k\|^2 - \dfrac{(y_k^T s_k)^2}{\|s_k\|^2}}{2a_k \|s_k\|^2} \pm \frac{\sqrt{\left(\|y_k\|^2 - a_k^2 \|s_k\|^2 - \dfrac{(y_k^T s_k)^2}{\|s_k\|^2}\right)^2 + 4a_k \|s_k\|^2 \|y_k{}^2\|}}{2a_k \|s_k\|^2},$$

$$\theta = 0.26.$$

Figure 8 shows the performances of Dai-Liao algorithm with t given by (15) versus the Hestenes-Stiefel algorithm.

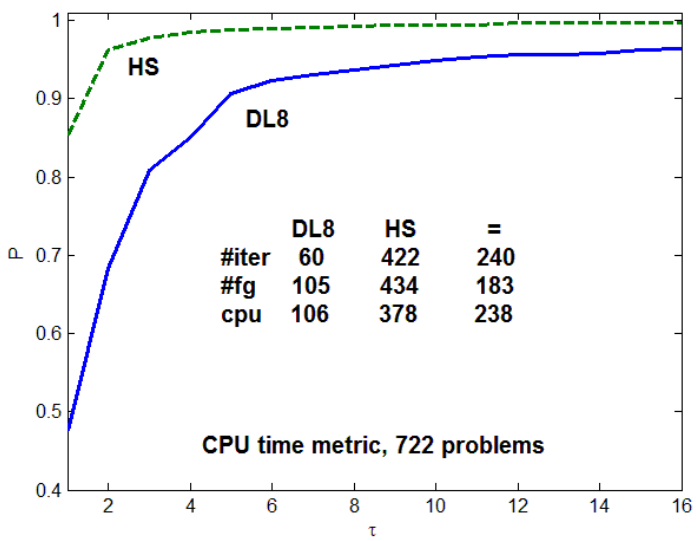

Fig. 8. Performances profiles of DL8 (Dai-Liao with t given by (15)) versus HS (Hestenes-Stiefel)

9) Another set of numerical experiments takes the scalar t as

$$t = \frac{y_k^T s_k}{\|s_k\|^2} + \frac{\|y_k\|}{\|s_k\|}, \tag{16}$$

introduced by Babaie-Kafaki & Ghanbari [6]. Figure 9 shows the performance profiles of these algorithms.

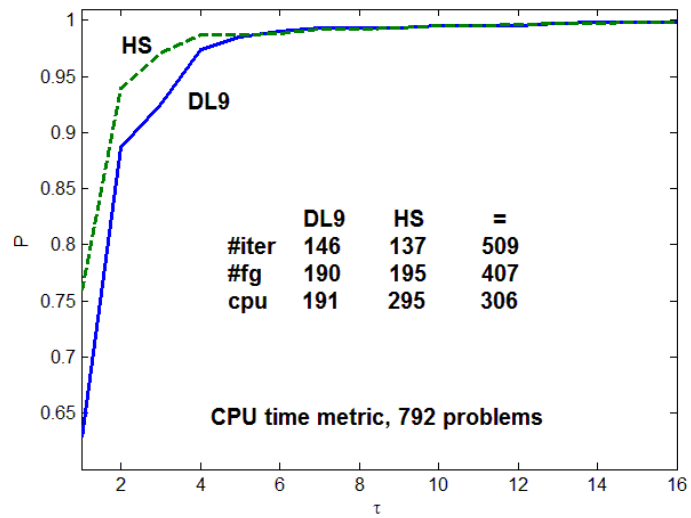

Fig. 9. Performances profiles of DL9 (Dai-Liao with t given by (16)) versus HS (Hestenes-Stiefel)

10) In this set of numerical experiment

$$t = \frac{\|y_k\|}{\|s_k\|}, \tag{17}$$

introduced by Babaie-Kafaki & Ghanbari [6]. Figure 10 shows the performance profiles of these algorithms.

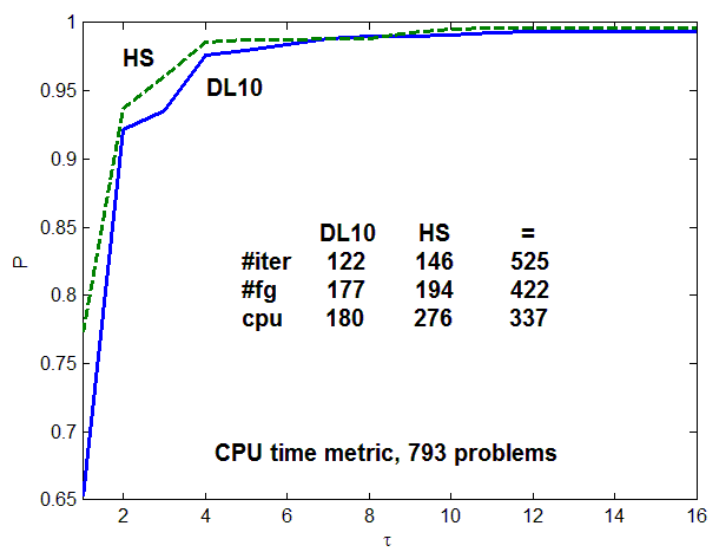

Fig. 10. Performances profiles of DL10 (Dai-Liao with t given by (17)) versus HS (Hestenes-Stiefel)

11) In this set of numerical experiments let us consider

$$t = 2\frac{\|y_k\|^2}{y_k^T s_k}, \tag{18}$$

which correspond to the CG-DESCENT algorithm by Hager and Zhang [22]. In Figure 11 we present the performance profiles of these algorithms.

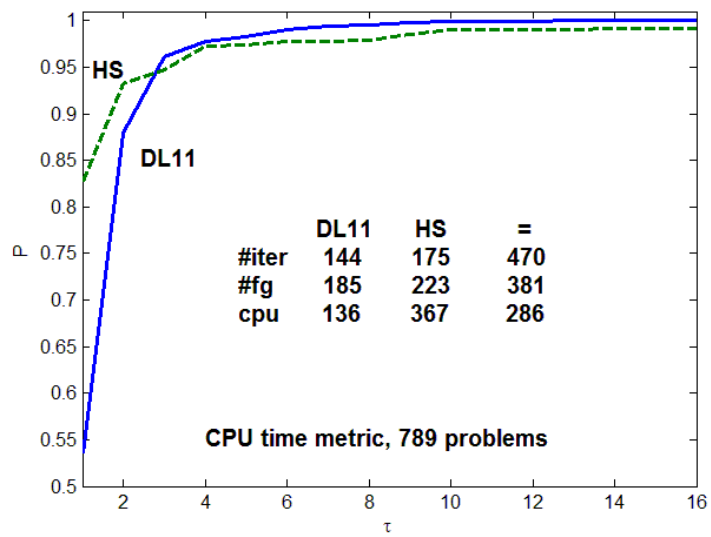

Fig. 11. Performances profiles of DL11 (Dai-Liao with t given by (18)) versus HS (Hestenes-Stiefel)

12) Now, consider

$$t = \frac{\|y_k\|^2}{y_k^T s_k}, \tag{19}$$

introduced by Dai and Kou [23]. Figure 12 presents the profiles of these algorithms.

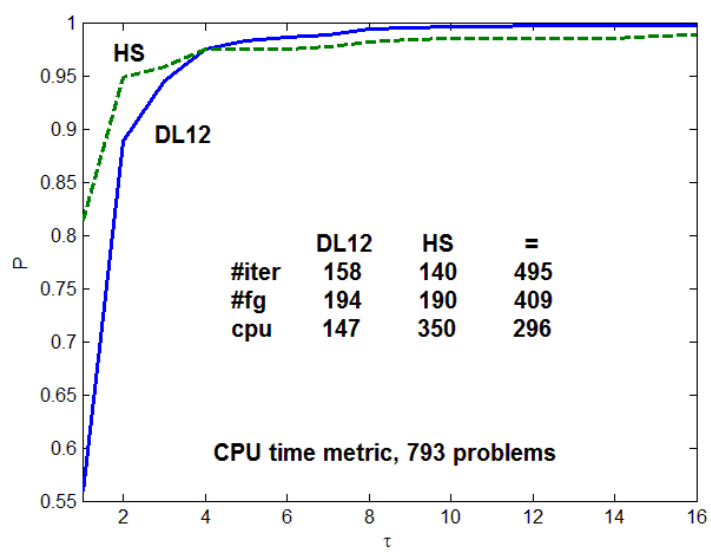

Fig. 12. Performances profiles of DL12 (Dai-Liao with t given by (19)) versus HS (Hestenes-Stiefel)

13) For $t = 0.1$ in (4), Figure 13 shows the performances of these algorithms.

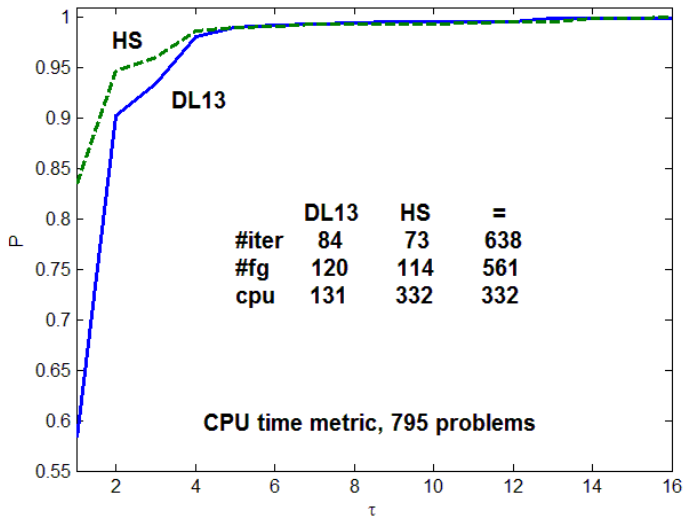

Fig. 13. Performances profiles of DL13 (Dai-Liao with $t = 0.1$) versus HS (Hestenes-Stiefel)

14) For $t = 0.5$ in (4), Figure 14 shows the performances of these algorithms

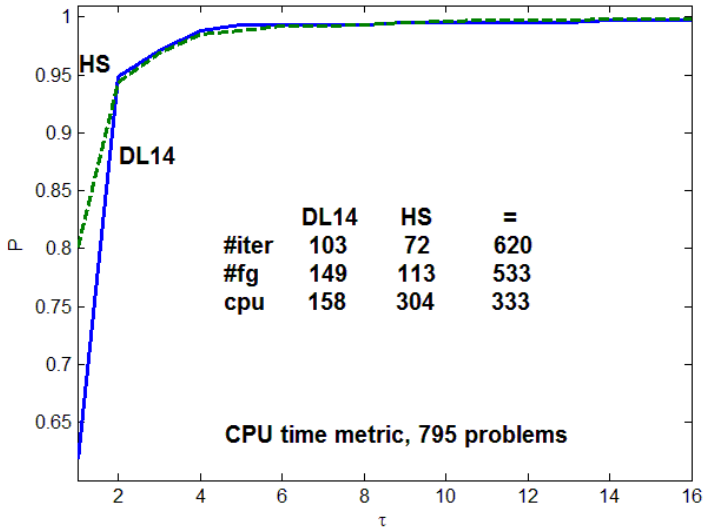

Fig. 14. Performances profiles of DL14 (Dai-Liao with $t = 0.5$) versus HS (Hestenes-Stiefel)

15) For $t = 0.9$ in (4), Figure 15 shows the performances of these algorithms

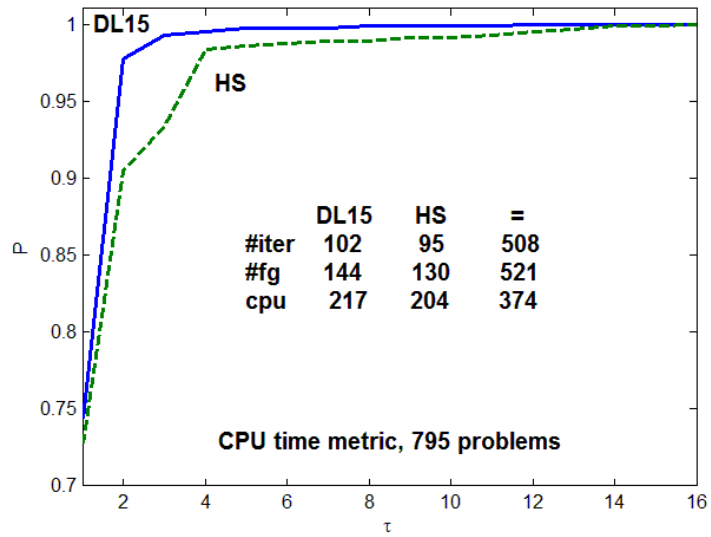

Fig. 15. Performances profiles of DL15 (Dai-Liao with $t = 0.9$) versus HS (Hestenes-Stiefel)

16) Consider

$$t = \frac{\|s_k\|^2}{y_k^T s_k} \tag{20}$$

in (4). Figure 16 shows the performances of these algorithms.

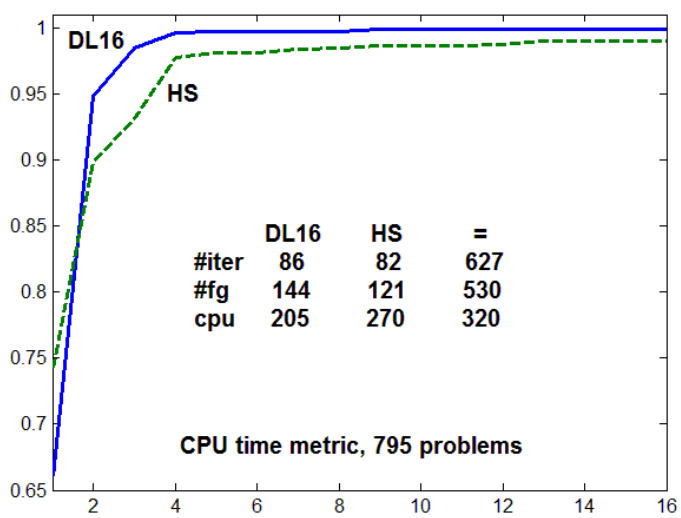

Fig. 16. Performances profiles of DL16 (Dai-Liao with t given by (20)) versus HS (Hestenes-Stiefel)

17) Now consider in (4)

$$t = \frac{s_k^T g_{k+1}}{y_k^T s_k}. \tag{21}$$

Figure 17 shows the performance profiles of these algorithms.

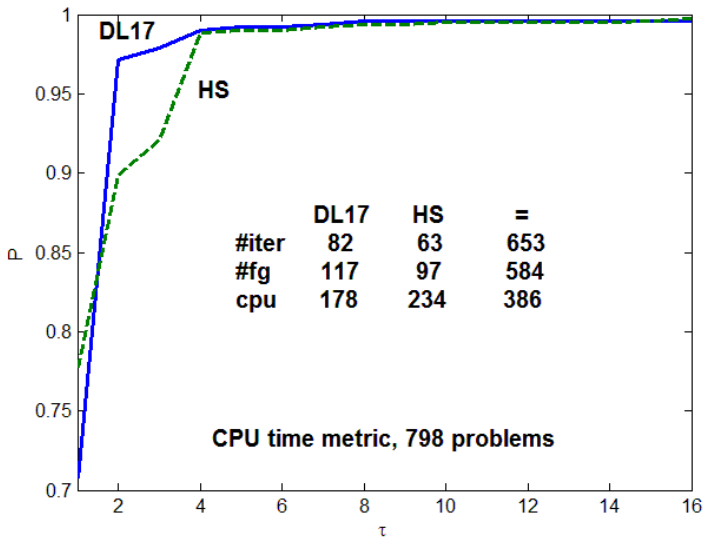

Fig. 17. Performances profiles of DL17 (Dai-Liao with t given by (21)) versus HS (Hestenes-Stiefel)

References

[1] Andrei, N., (2020). *Nonlinear Conjugate Gradient Methods for Unconstrained Optimization*, Springer Optimization and Its Applications Vol. 158, Springer Nature, 2020.

[2] Dai, Y.H., & Liao, L.Z., (2001). New conjugate conditions and related nonlinear conjugate gradient methods, *Applied Mathematics & Optimization, 43*, 87-101.

[3] Andrei, N., (2011). Open problems in conjugate gradient algorithms for unconstrained optimization. *Bulletin of the Malaysian Mathematical Sciences Society, 34*(2), 319-330.

[4] Andrei, N., (2017). A Dai–Liao conjugate gradient algorithm with clustering of eigenvalues. Numer Algorithms, 77:1273–1282.

[5] Andrei, N., (2018). A Dai-Liao conjugate gradient algorithm with clustering the eigenvalues. *Numerical Algorithms, 77*(4), 1273-1282.

[6] Babaie-Kafaki, S., & Ghanbari, R., (2014). The Dai-Liao nonlinear conjugate gradient method with optimal parameter choices. *European Journal of Operational Research, 234*, 625-630.

[7] Fatemi, M., (2016). An optimal parameter for Dai-Liao family of conjugate gradient methods. *Journal of Optimization Theory and Applications, 169*, 587-605

[8] Aminifard, Z., & Babaie-Kafaki, S., (2018). An optimal parameter choice for the Dai–Liao family of conjugate gradient methods by avoiding a direction of the maximum magnification by the search direction matrix. *4OR*, https://doi.org/10.1007/s10288-018-0387-1

[9] Babaie-Kafaki, S., & Ghanbari, R., (2014). A descent family of Dai–Liao conjugate gradient methods. *Optim. Methods Softw 29*(3):583–591

[10] Babaie-Kafaki, S., & Ghanbari, R., (2015). Two optimal Dai–Liao conjugate gradient methods. *Optimization*. *64*(11):2277–2287

[11] Babaie-Kafaki, S., & Ghanbari, R., (2017a). A class of adaptive Dai–Liao conjugate gradient methods based on the scaled memoryless BFGS update. *4OR 15*(1):85–92

[12] Babaie-Kafaki, S., & Ghanbari, R., (2017b). A class of descent four-term extension of the Dai–Liao conjugate gradient method based on the scaled memoryless BFGS update. *J Ind Manag Optim 3*(2):649–658

[13] Babaie-Kafaki, S., & Ghanbari, R., (2017c). Two adaptive Dai–Liao nonlinear conjugate gradient methods. *Iran J. Sci Technol Trans Sci 42*:1505–1509.

[14] Livieris, I.E., & Pintelas, P., (2012). A descent Dai–Liao conjugate gradient method based on a modified secant equation and its global convergence. *ISRN Comput Math 2012:8* Article ID 435495

[15] Peyghami, M.R., Ahmadzadeh, H., & Fazli, A., (2015). A new class of efficient and globally convergent conjugate gradient methods in the Dai–Liao family. *Optim Methods Softw 30*(4):843–863

[16] Andrei, N., (2018). *UOP - A collection of 80 unconstrained optimization test problems*. (Technical Report No. 7/2018, November 17, Research Institute for Informatics, Bucharest, Romania).

[17] Dolan, E.D., & Moré, J.J., (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming, 91*, 201-213.

[18] Oren, S.S., & Luenberger, D.G., (1974). Self-scaling variable metric (SSVM) algorithms, part I: criteria and sufficient conditions for scaling a class of algorithms. *Management Science, 20*, 845-862.

[19] Oren, S.S., & Spedicato, E., (1976). Optimal conditioning of self-scaling variable metric algorithm. *Mathematical Programming, 10*, 70-90.

[20] Al-Baali, M., (1998). Numerical experience with a class of self-scaling quasi-Newton algorithms, *Journal of Optimization Theory and Applications, 96*, 533-553.

[21] Byrd, R.H., & Nocedal, J., (1989). A tool for the analysis of quasi-Newton methods with application to unconstrained minimization. *SIAM Journal on Numerical Analysis, 26*, 727-739.

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

[23] Dai, Y.H., & Kou, C.X., (2013). A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search. *SIAM Journal on Optimization, 23*(1) 296-320.