Hindawi
Journal of Function Spaces
Volume 2021, Article ID 6693401, 10 pages
https://doi.org/10.1155/2021/6693401

Research Article

A Novel Value for the Parameter in the Dai-Liao-Type Conjugate Gradient Method

Branislav Ivanov,[1] Predrag S. Stanimirović,[2] Bilall I. Shaini,[3] Hijaz Ahmad,[4] and Miao-Kun Wang[5]

[1]*Technical Faculty in Bor, University of Belgrade, Vojske Jugoslavije 12, 19210 Bor, Serbia*
[2]*Faculty of Sciences and Mathematics, University of Niš, Višegradska 33, 18000 Niš, Serbia*
[3]*University of Tetovo, St. Ilinden, n.n., Tetovo, North Macedonia*
[4]*Department of Basic Sciences, University of Engineering and Technology Peshawar, Pakistan*
[5]*Department of Mathematics, Huzhou University, Huzhou 313000, China*

Correspondence should be addressed to Miao-Kun Wang; wmk000@126.com

Received 29 October 2020; Revised 28 November 2020; Accepted 15 January 2021; Published 30 January 2021

Academic Editor: Ioan Rasa

A new rule for calculating the parameter t involved in each iteration of the MHSDL (Dai-Liao) conjugate gradient (CG) method is presented. The new value of the parameter initiates a more efficient and robust variant of the Dai-Liao algorithm. Under proper conditions, theoretical analysis reveals that the proposed method in conjunction with backtracking line search is of global convergence. Numerical experiments are also presented, which confirm the influence of the new value of the parameter t on the behavior of the underlying CG optimization method. Numerical comparisons and the analysis of obtained results considering Dolan and Moré's performance profile show better performances of the novel method with respect to all three analyzed characteristics: number of iterative steps, number of function evaluations, and CPU time.

1. Introduction and Background Results

The topic of our research is solving the unconstrained nonlinear optimization problem

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

where the function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable and bounded below. Following the standard notation, $g_k = \nabla f(x_k)$ denotes the gradient, $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = g_k - g_{k-1}$. Using an extended conjugacy condition

$$d_k^{\mathrm{T}} y_{k-1} = -t g_k^{\mathrm{T}} s_{k-1}, \quad t > 0, \tag{2}$$

Dai and Liao in [1] proposed the conjugate gradient (CG) method

$$x_{k+1} = x_k + \alpha_k d_k, \tag{3}$$

where the step size α_k is a positive parameter, x_k is an already generated point, x_{k+1} is a new iterative point, and d_k is a suitable search direction. The search directions d_k are generated by the conceptual formula

$$d_k = \begin{cases} -g_0, & k = 0, \\ -g_k + \beta_k^{\mathrm{DL}} d_{k-1}, & k \geq 1, \end{cases} \tag{4}$$

where the conjugate gradient coefficient β_k^{DL} is defined by

$$\beta_k^{\mathrm{DL}} = Y(t) \coloneqq \frac{g_k^{\mathrm{T}} y_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}} - t \frac{g_k^{\mathrm{T}} s_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}}, \quad t > 0, \tag{5}$$

wherein $t > 0$ is a scalar.

Some well-known formulas for defining β_k have been created by modifying the conjugate gradient parameter β_k^{DL} [2–9]. One of them is denoted as β_k^{MHSDL} and defined

```
The backtracking line search.
**Require**: Nonlinear objective function $f(x)$, search direction $d_k$, previous point $x_k$, and real quantities $0 < \omega < 0.5$ and $\varphi \in (0, 1)$.
1: $\alpha = 1$.
2: While $f(x_k + \alpha d_k) > f(x_k) + \omega \alpha g_k^{\mathrm{T}} d_k$, do $\alpha := \alpha\varphi$.
3: Return $\alpha_k = \alpha$.
```

Algorithm 1:

in [7] by

$$\beta_k^{\mathrm{MHSDL}} = Y_1(t) := \frac{g_k^{\mathrm{T}} \widehat{y_{k-1}}}{d_{k-1}^{\mathrm{T}} y_{k-1}} - t \frac{g_k^{\mathrm{T}} s_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}}, \tag{6}$$

where $t > 0$ is a scalar as in (5) and $\widehat{y_{k-1}} = g_k - (\|g_k\| / \|g_{k-1}\|) g_{k-1}$.

The family of CG methods for nonlinear optimization has reached great popularity lately, thanks to the various benefits and advantages it possesses. The most important property is based on computationally efficient iterations arising from a simple CG rule. This property initiates the high efficiency of CG methods with respect to analogous methods for nonlinear optimization. Moreover, global convergence is ensured under suitable conditions. Finally, the application of various CG methods in solving image restoration problems has become an important research topic [10, 11].

Since the parameter t is important for the numerical behavior of Dai-Liao (DL) CG methods [12], one of the most important problems in the implementation of the DL class CG method is to determine a proper value $t > 0$ which will give desirable results. Many scientists have invested a lot of time and effort in the previous period to determine the best definition of the nonnegative parameter t in the DL class CG methods. So far, the research in finding the appropriate value of t has evolved in two directions. One group of methods is aimed at finding an appropriate fixed value for t [1, 2, 6–8], while methods from another group promote appropriate rules for computing values of t in each iteration, which ensure a satisfactory decrease of the objective. In our research, we will pay attention to the second research stream: find the parameter t whose values change through iterations so that the faster convergence is achieved. The value of the parameter t defined in the kth iteration will be denoted by $t(k) := t_k$.

In order to complete the presentation, we will restate the main principles proposed so far for computing t_k. Hager and Zhang in [13, 14] proposed the DL CG method (5), known as CG-DESCENT, where $t(k) \equiv t_{k1}$ is defined by

$$t(k) \equiv t_{k1} := 2 \frac{\|y_{k-1}\|^2}{y_{k-1}^{\mathrm{T}} s_{k-1}}. \tag{7}$$

Dai and Kou [15] suggested the conjugate gradient coefficient β_k^{DK} of the form

$$\beta_k^{\mathrm{DK}} = Y\left(\tau_k + \frac{\|y_{k-1}\|^2}{y_{k-1}^{\mathrm{T}} s_{k-1}} - \frac{y_{k-1}^{\mathrm{T}} s_{k-1}}{\|s_{k-1}\|^2}\right) = \frac{g_k^{\mathrm{T}} y_{k-1}}{y_{k-1}^{\mathrm{T}} d_{k-1}} - \left(\tau_k + \frac{\|y_{k-1}\|^2}{y_{k-1}^{\mathrm{T}} s_{k-1}} - \frac{y_{k-1}^{\mathrm{T}} s_{k-1}}{\|s_{k-1}\|^2}\right) \frac{g_k^{\mathrm{T}} s_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}}, \tag{8}$$

where τ_k is the scaling parameter arising from the self-scaling memoryless BFGS method. Clearly, the Dai and Kou (DK) method is a member of the DL class CG methods, which is determined by

$$t(k) \equiv t_{k2} := \tau_k + \frac{\|y_{k-1}\|^2}{y_{k-1}^{\mathrm{T}} s_{k-1}} - \frac{y_{k-1}^{\mathrm{T}} s_{k-1}}{\|s_{k-1}\|^2}. \tag{9}$$

The results given in [15] confirm that the DK iterations outperform many existing CG methods. Following the development of DL methods, Babaie-Kafaki and Ghanbari [16] defined two new ways to calculate the value of the parameter t in (5), as in the following two formulas:

$$\begin{aligned} t(k) &\equiv t_{k3} := \frac{s_{k-1}^{\mathrm{T}} y_{k-1}}{\|s_{k-1}\|^2} + \frac{\|y_{k-1}\|}{\|s_{k-1}\|}, \\ t(k) &\equiv t_{k4} := \frac{\|y_{k-1}\|}{\|s_{k-1}\|}. \end{aligned} \tag{10}$$

Andrei in [17] proposed the new rule for calculating t in order to define $Y(t)$ in (5) and defined a new variant of the DL class CG methods, denoted by DLE, with

$$t(k) \equiv t_{k5} := \frac{s_{k-1}^{\mathrm{T}} y_{k-1}}{\|s_{k-1}\|^2}. \tag{11}$$

Lotfi and Hosseini in [18] proposed the following rule for determining the parameter $t(k)$, using the expression

$$t(k) \equiv t_{k6} := \max \left\{ t_{k6}^{*}, \upsilon \frac{\|y_{k-1}\|^2}{s_{k-1}^{\mathrm{T}} y_{k-1}} \right\}, \tag{12}$$

```
Effective Dai-Liao (EDL) CG method.
Require: An initial point $x_0$ and quantities $0 < \varepsilon < 1$, $0 < \delta < 1$.
1: Assign $k = 0$ and $d_0 = -g_0$.
2: If
        $\|g_k\| \le \varepsilon$ and $((|f(x_{k+1}) - f(x_k)|)/(1 + |f(x_k)|)) \le \delta$,
    STOP;
    else go to Step 3.
3: Calculate $\alpha_k \in (0,1)$ using Algorithm 1 (backtracking line search).
4: Compute $x_{k+1} = x_k + \alpha_k d_k$.
5: Calculate $g_{k+1}$, $y_k = g_{k+1} - g_k$, $s_k = x_{k+1} - x_k$.
6: Compute $t_k^*$ by (16).
7: Calculate $\beta_{k+1}^{\mathrm{EDL}}$ by (18).
8: Compute $d_{k+1} = -g_{k+1} + \beta_{k+1}^{\mathrm{EDL}} d_k$.
9: Let $k := k + 1$, and go to Step 2.
```

Algorithm 2:

where

$$t_{k6}^* := \frac{(1 - h_k\|g_{k-1^r}\|)s_{k-1}^{\mathrm{T}}g_k + \left(g_k^{\mathrm{T}}y_{k-1}/y_{k-1}^{\mathrm{T}}s_{k-1}\right)h_k\|g_{k-1}\|^r\|s_{k-1}\|^2}{g_k^{\mathrm{T}}s_{k-1} + \left(g_k^{\mathrm{T}}s_{k-1}/s_{k-1}^{\mathrm{T}}y_{k-1}\right)h_k\|g_{k-1}\|^r\|s_{k-1}\|^2},$$
$$h_k = C + \max\left\{-\frac{s_{k-1}^{\mathrm{T}}y_{k-1}}{\|s_{k-1}\|^2}, 0\right\}\|g_{k-1}\|^{-r}, \tag{13}$$

and $v > 1/4$, C, and r are three positive constants.

On the basis of the above overview of the main CG methods and motivated by the strong theoretical properties and computational efficiency of modified Dai-Liao CG methods proposed by many researchers, we suggest a new way of calculating the value of the parameter $t(k)$. As a consequence, the corresponding CG method of DL type, termed as the *Effective Dai-Liao* (EDL) method, is proposed and its convergence is proven. Numerical testing and comparison with other known DL variants are presented in order to show the effectiveness of the introduced method. Analysis of generated numerical results exhibits that the proposed EDL method is efficient compared with other DL-type methods.

The global organization of sections is described as follows. Introduction, motivation, and a brief overview of the preliminary results are given in Section 1. A new rule for calculating the variable parameter $t(k)$ is proposed in Section 2. An effective algorithm and global convergence of the EDL method initiated by $t(k)$ are given in the same section. The new EDL method is tested in Section 3 on some unlimited optimization test problems and compared against some known variants of the DL class methods. Finally, concluding remarks are presented in the last concluding section.

2. A Modified Dai-Liao Method and Its Convergence

Popularity in defining new rules for calculating $t(k)$ is a guarantee that such an approach is effective and still insufficiently explored. The idea for defining a new parameter t_k^* comes from previously described rules for computing $t(k)$, particularly from the paper Li and Ruan [19] and from the idea which can be found in the paper Yuan et al. [11]. Further, analyzing the results from [1, 2, 6–8], we conclude that the scalar t was defined by a fixed value of 0.1 in related numerical experiments. Also, numerical experience related to the fixed valued $t = 1$ was reported in [1]. According to this experience, our intention is to define variable values $t(k)$ inside the interval $(0, 1)$.

To successfully define $t(k)$ with values belonging to the interval $(0, 1)$, let us start from the definition of the quantity L_k which was used in defining the direction d_k in [19]. The parameter L_k was defined by $L_k = s_{k-1}^{\mathrm{T}}s_{k-1}/s_{k-1}^{\mathrm{T}}y_{k-1}^* \in (0,1)$, $k \ge 0$, where

$$y_{k-1}^* = y_{k-1} + \left(\max\left\{0, -\frac{s_{k-1}^{\mathrm{T}}y_{k-1}}{\|s_{k-1}\|^2}\right\} + 1\right)s_{k-1}. \tag{14}$$

By putting y_{k-1}^* into L_k, the following can be obtained:

$$L_k = \frac{s_{k-1}^{\mathrm{T}}s_{k-1}}{s_{k-1}^{\mathrm{T}}\left(y_{k-1} + \left(\max\left\{0, -\left(s_{k-1}^{\mathrm{T}}y_{k-1}/\|s_{k-1}\|^2\right)\right\} + 1\right)s_{k-1}\right)}$$
$$= \frac{\|s_{k-1}\|^2}{s_{k-1}^{\mathrm{T}}y_{k-1} + \left(\max\left\{0, -\left(s_{k-1}^{\mathrm{T}}y_{k-1}/\|s_{k-1}\|^2\right)\right\} + 1\right)\|s_{k-1}\|^2}. \tag{15}$$

Further, with certain modifications and substitutions in the equation defining L_k, as well as using the function max, which chooses the maximum between the value of the expression $d_{k-1}^{\mathrm{T}}g_k$ and 1, we come to a new definition of the parameter $t(k)$. As described in advance imposed desired restrictions, the novel parameter t_k^* is defined by

$$t_k^* = \frac{\|g_k\|^2}{\max\left\{1, d_{k-1}^{\mathrm{T}}g_k\right\} + \left(\max\left\{0, \left(d_{k-1}^{\mathrm{T}}g_k/\|g_k\|^2\right)\right\} + 1\right)\|g_k\|^2}. \tag{16}$$

Table 1: Summary results of EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods with respect to NI.

Test function	MHSDL3	MHSDL4	MHSDL5	EDL	MHSDL6
Extended penalty	1466	2243	2231	1259	1371
Perturbed quadratic	1203710	754291	746557	305622	423037
Raydan 1	159055	110587	106586	55477	75154
Raydan 2	1636	441	441	70	209
Diagonal 1	116788	78844	73512	30978	20332
Diagonal 2	176983	270434	271595	515000	271295
Diagonal 3	150328	98647	104417	47155	37711
Hager	8666	5219	5157	3234	3625
Generalized tridiagonal 1	1862	1471	1485	639	877
Extended TET	1357	5954	5915	4030	2664
Diagonal 4	30693	19589	19332	8040	12012
Diagonal 5	1721	25120	25120	60	216
Extended Himmelblau	1777	8023	7946	1376	3682
Perturbed quadratic diagonal	2940970	2115659	2027128	1136414	1352704
Quadratic QF1	1270802	799192	786032	309509	325415
Extended quadratic penalty QP1	770	594	575	560	543
Extended quadratic penalty QP2	399671	240530	245254	96620	137799
Extended quadratic exponential EP1	462	606	606	513	526
Extended tridiagonal 2	3119	2176	2177	1132	1455
ARWHEAD (CUTE)	88824	69868	67413	40713	48669
ENGVAL1 (CUTE)	2323	1407	1415	552	820
INDEF (CUTE)	20	31	1080	23	36240
QUARTC (CUTE)	173913	262291	262291	524299	262181
Diagonal 6	1824	508	508	70	227
Generalized quartic	1208	1403	2846	1265	1154
Diagonal 7	3217	655	655	653	580
Diagonal 8	511	698	698	686	596
Full Hessian FH3	1456	5353	5350	2523	3176

It is easy to verify that t_k^* defined by (16) satisfies

$$0 < t_k^* \le \frac{\|g_k\|^2}{1 + (0+1)\|g_k\|^2} = \frac{\|g_k\|^2}{1 + \|g_k\|^2} < 1. \tag{17}$$

Accordingly, $t_k^* \in (0, 1)$, which was our initial intention. Clearly, greater values of $\|g_k\|$ lead to values $t_k^* \nearrow 1$. Further, since the trend $\|g_k\| \longrightarrow 0$ is expectable, we can expect smaller values $t_k^* \searrow 0$ in late iterations. Therefore, t_k^* is suitable for defining corresponding conjugate gradient coefficient $Y(t)$ or $Y_1(t)$ and further DL CG iterations (4).

Considering $t = t_k^*$ in (6), it is reasonable to propose a novel variant of the Dai-Liao CG parameter β_k^{EDL} which is subject to the following rule during the iterative process:

$$\beta_k^{\mathrm{EDL}} = Y_1(t_k^*) := \frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)\left|g_k^{\mathrm{T}} g_{k-1}\right|}{d_{k-1}^{\mathrm{T}} y_{k-1}} - t_k^* \frac{g_k^{\mathrm{T}} s_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}}. \tag{18}$$

Before the main algorithm, it is necessary to define the backtracking line search as one of the most popular and practical methods for computing the step length α_k in (3). The procedure for the backtracking line search proposed in [20] starts from the initial value $\alpha = 1$ and generates output values which ensure that the goal function decreases in each iteration. Consequently, it is appropriate to use Algorithm 1, restated from [21], in order to determine the primary step size α_k.

Algorithm 2 describes a computational framework for the EDL method.

It is necessary to examine the properties of the EDL method and prove its convergence.

Assumption 1.

(1) The level set $\mathscr{M} = \{x \in \mathbb{R}^n \mid f(x) \le f(x_0)\}$, defined upon the initial point x_0 of the iterative method (3), is bounded.

(2) The goal function f is continuous and differentiable in a neighborhood $\mathscr{P}$ of M with the Lipschitz

Table 2: Summary results of EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods with respect to NFE.

Test function	MHSDL3	MHSDL4	MHSDL5	EDL	MHSDL6
Extended penalty	54876	73764	73429	46820	49791
Perturbed quadratic	56691737	34287604	33885701	13168688	18486375
Raydan 1	5066739	3364983	3236335	1551846	2170553
Raydan 2	6554	1162	1162	159	428
Diagonal 1	5004640	3256274	3022015	1200086	744278
Diagonal 2	353976	540878	543200	1030010	542600
Diagonal 3	6339146	3998904	4229565	1798032	1400076
Hager	192474	107413	106534	59187	69735
Generalized tridiagonal 1	37429	27860	28138	10760	15177
Extended TET	19546	77422	76925	40340	29334
Diagonal 4	713120	425023	418666	155027	242443
Diagonal 5	6874	50460	50460	140	442
Extended Himmelblau	45972	192362	190524	26104	80854
Perturbed quadratic diagonal	135901222	94177165	90238441	48147512	57702654
Quadratic QF1	55972697	33836473	33243711	12316721	12853424
Extended quadratic penalty QP1	17016	12882	12565	11116	10544
Extended quadratic penalty QP2	13015888	7454686	7584960	2743358	4030601
Extended quadratic exponential EP1	14914	18463	18463	14132	15133
Extended tridiagonal 2	36450	22564	22379	9687	12920
ARWHEAD (CUTE)	4296028	3305257	3182138	1846606	2230650
ENGVAL1 (CUTE)	40462	22432	22898	8209	12858
INDEF (CUTE)	1808	2182	5995	2060	104962
QUARTC (CUTE)	347926	524662	524662	1048648	524422
Diagonal 6	7394	1416	1408	159	468
Generalized quartic	14364	21842	48770	16695	14103
Diagonal 7	6454	6838	6838	3891	4521
Diagonal 8	6098	6938	6938	4161	5494
Full Hessian FH3	60792	212799	212701	89890	114962

continuous gradient g. This assumption implies the existence of a positive constant $L > 0$ satisfying

$$\|g(u) - g(v)\| \le L\|u - v\|, \quad \forall u, v \in \mathscr{P}. \tag{19}$$

Assumption 1 initiates the existence of positive constants D and γ satisfying

$$\begin{aligned} \|u - v\| &\le D, \quad \forall u, v \in \mathscr{P}, \\ \|g(u)\| &\le \gamma, \quad \forall u \in \mathscr{P}. \end{aligned} \tag{20}$$

The conditions from Assumption 1 are assumed. In view of the uniform convexity of f, there is a constant $\theta > 0$ that satisfies

$$(g(u) - g(v))^{\mathrm{T}}(u - v) \ge \theta\|u - v\|^2, \quad \text{for all } u, v \in \mathscr{M}, \tag{21}$$

or equivalently,

$$f(u) \ge f(v) + g(v)^{\mathrm{T}}(u - v) + \frac{\theta}{2}\|u - v\|^2, \quad \text{for all } u, v \in \mathscr{M}. \tag{22}$$

It follows from (21) and (22) that

$$s_{k-1}^{\mathrm{T}} y_{k-1} \ge \theta\|s_{k-1}\|^2, \tag{23}$$

$$f(x_{k-1}) - f(x_k) \ge -g(x_k)^{\mathrm{T}} s_{k-1} + \frac{\theta}{2}\|s_{k-1}\|^2. \tag{24}$$

By (19) and (23), one concludes

$$\theta\|s_{k-1}\|^2 \le s_{k-1}^{\mathrm{T}} y_{k-1} \le L\|s_{k-1}\|^2, \tag{25}$$

where the inequality implies $\theta \le L$.

The inequality (25) initiates

$$s_{k-1}^{\mathrm{T}} y_{k-1} = \alpha_{k-1} d_{k-1}^{\mathrm{T}} y_{k-1} > 0. \tag{26}$$

Table 3: Summary results of EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods with respect to CPU time (sec).

Test function	MHSDL3	MHSDL4	MHSDL5	EDL	MHSDL6
Extended penalty	29.75	34.11	31.42	18.30	24.27
Perturbed quadratic	40532.66	24358.20	24947.84	8335.80	13225.80
Raydan 1	3054.67	1904.48	1692.06	690.91	1184.86
Raydan 2	6.77	1.58	1.66	0.31	0.77
Diagonal 1	7834.03	5106.41	4592.28	1476.89	486.09
Diagonal 2	885.13	1428.05	1447.02	2352.11	1513.50
Diagonal 3	13614.27	8416.77	9064.30	3132.02	1916.30
Hager	586.63	325.75	333.41	142.06	198.13
Generalized tridiagonal 1	66.14	35.59	34.42	15.19	21.63
Extended TET	20.50	78.34	82.94	41.23	31.45
Diagonal 4	134.53	77.86	87.88	30.41	55.34
Diagonal 5	18.06	134.73	121.09	0.56	1.84
Extended Himmelblau	11.13	44.47	44.36	6.19	18.30
Perturbed quadratic diagonal	91655.55	58226.16	60920.06	32179.38	36383.83
Quadratic QF1	62610.50	31552.48	28679.91	8832.11	8465.34
Extended quadratic penalty QP1	7.56	7.25	6.98	4.98	4.94
Extended quadratic penalty QP2	3814.16	2128.86	2288.55	671.52	1204.72
Extended quadratic exponential EP1	9.11	10.23	8.55	8.00	8.02
Extended tridiagonal 2	11.13	8.83	6.95	4.08	5.25
ARWHEAD (CUTE)	2709.42	2336.92	2369.28	1266.80	1689.80
ENGVAL1 (CUTE)	19.47	11.33	11.81	4.03	6.70
INDEF (CUTE)	2.44	2.89	10.70	1.92	774.34
QUARTC (CUTE)	3106.56	4818.58	4808.70	7138.72	4735.39
Diagonal 6	6.75	1.92	2.03	0.38	1.34
Generalized quartic	7.16	11.53	21.05	7.53	9.78
Diagonal 7	5.98	8.20	8.28	4.56	6.25
Diagonal 8	6.17	8.20	8.08	4.72	7.69
Full Hessian FH3	30.08	66.45	79.48	35.77	43.42

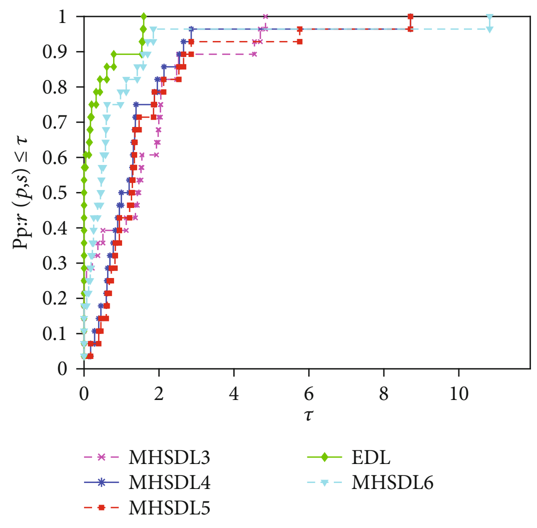

Figure 1: NI performance profile for EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods.

Taking into account $\alpha_{k-1} > 0$ and the last inequality, we conclude

$$d_{k-1}^{\mathrm{T}} y_{k-1} > 0. \tag{27}$$

Lemma 2. *[22, 23]. Let Assumption 1 be accomplished and the points $\{x_k\}$ be generated by the method (3)–(4). Then, it holds*

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < +\infty. \tag{28}$$

Lemma 3. *Consider the proposed Dai-Liao CG method, including (3), (4), and (18). If the search procedure guarantees (27), for all $k \geq 0$, then the next inequality holds*

$$g_k^T d_k \leq -c\|g_k\|^2, \tag{29}$$

for some $0 \leq c \leq 1$.

Proof. The inequality (29) will be verified by induction. In the initial situation $k = 0$, one obtains $g_0^{\mathrm{T}} d_0 = -\|g_0\|^2$. Since $c \leq 1$,

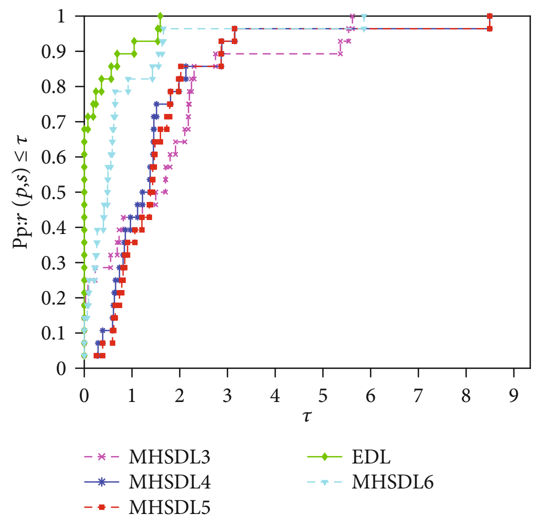

Figure 2: NFE performance profile for EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods.

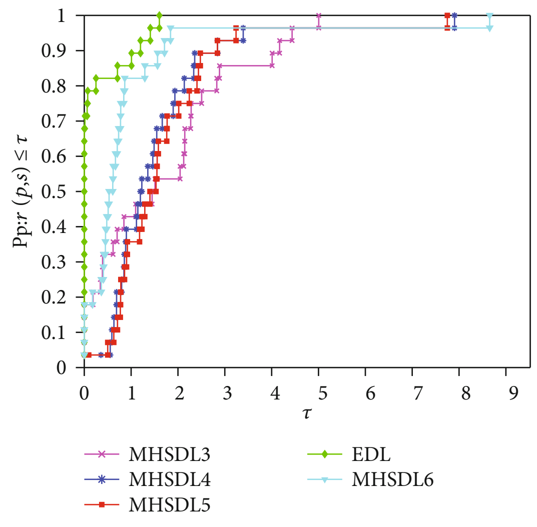

Figure 3: CPU performance profile for EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods.

obviously (29) is satisfied in the basic case. Suppose that (29) is valid for some $k \geq 1$. Taking the inner product of both the left- and right-hand sides in (4) with the vector g_k^T, the following can be obtained:

$$\begin{aligned} g_k^T d_k &= -\|g_k\|^2 + \beta_k^{EDL} g_k^T d_{k-1} \\ &= -\|g_k\|^2 + \left(\frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} - t_k^* \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right) g_k^T d_{k-1} \\ &= -\|g_k\|^2 + \frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} - t_k^* \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} \\ &= -\|g_k\|^2 + \frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} - t_k^* \frac{\alpha_{k-1} (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}}. \end{aligned} \tag{30}$$

Using (17) in common with (27) and $\alpha_{k-1} > 0$, we conclude

$$t_k^* \frac{\alpha_{k-1} (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}} > 0. \tag{31}$$

Now from (30), (31), and

$$0 \leq \beta_k^{MHS} = \frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} \leq \frac{\|g_k\|^2}{\lambda \mid g_k^T d_{k-1} \mid}, \quad \lambda \geq 1, \tag{32}$$

it follows that

$$\begin{aligned} g_k^T d_k &\leq -\|g_k\|^2 + \frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} \\ &\leq -\|g_k\|^2 + \frac{\|g_k\|^2}{\lambda |g_k^T d_{k-1}|} |g_k^T d_{k-1}| = -\left(1 - \frac{1}{\lambda}\right) \|g_k\|^2. \end{aligned} \tag{33}$$

In view of $\lambda \geq 1$, the inequality (29) is satisfied for $c = (1 - (1/\lambda))$ in (33) and arbitrary $k \geq 0$.

The global convergence of the proposed EDL method is confirmed by Theorem 4.

Theorem 4. *Let Assumption 1 be true and f be uniformly convex. Then, the sequence $\{x_k\}$ generated by (3), (4), and (18) fulfills*

$$\liminf_{k \to \infty} \|g_k\| = 0. \tag{34}$$

Proof. Suppose the opposite, i.e., (34) is not true. This implies the existence of a constant $c_1 > 0$ such that

$$\|g_k\| \geq c_1, \quad \text{for all } k. \tag{35}$$

Squaring both sides of (4) implies

$$\|d_k\|^2 = \|g_k\|^2 - 2\beta_k^{EDL} g_k^T d_{k-1} + \left(\beta_k^{EDL}\right)^2 \|d_{k-1}\|^2. \tag{36}$$

Taking into account (18), we can get

$$\begin{aligned} -2\beta_k^{EDL} g_k^T d_{k-1} &= -2 \left(\frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} - t_k^* \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right) g_k^T d_{k-1} \\ &= -2 \left(\frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)|g_k^T g_{k-1}|}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} - t_k^* \frac{\alpha_{k-1} (g_k^T d_{k-1})^2}{d_{k-1}^T y_{k-1}} \right). \end{aligned} \tag{37}$$

Now from (31) and (32), it follows that

$$-2\beta_k^{\mathrm{EDL}} g_k^{\mathrm{T}} d_{k-1} \le 2\left|\frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)\left|g_k^{\mathrm{T}} g_{k-1}\right|}{d_{k-1}^{\mathrm{T}} y_{k-1}}\right|\left|g_k^{\mathrm{T}} d_{k-1}\right| \le 2\frac{\|g_k\|^2}{\lambda\left|g_k^{\mathrm{T}} d_{k-1}\right|}\left|g_k^{\mathrm{T}} d_{k-1}\right| = 2\frac{\|g_k\|^2}{\lambda}. \tag{38}$$

Now, an application of (18) initiates

$$\begin{aligned}
\beta_k^{\mathrm{EDL}} &= \frac{\|g_k\|^2 - (\|g_k\|/\|g_{k-1}\|)\left|g_k^{\mathrm{T}} g_{k-1}\right| - t_k^* g_k^{\mathrm{T}} s_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}} \\
&\le \left|\frac{g_k^{\mathrm{T}} g_k - (\|g_k\|/\|g_{k-1}\|)\left|g_k^{\mathrm{T}} g_{k-1}\right| - t_k^* g_k^{\mathrm{T}} s_{k-1}}{d_{k-1}^{\mathrm{T}} y_{k-1}}\right| \\
&\le \frac{\left|g_k^{\mathrm{T}}(g_k - (\|g_k\|/\|g_{k-1}\|) g_{k-1} - t_k^* s_{k-1})\right|}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&= \frac{\left|g_k^{\mathrm{T}}(g_k - g_{k-1} + g_{k-1} - (\|g_k\|/\|g_{k-1}\|) g_{k-1} - t_k^* s_{k-1})\right|}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&\le \frac{\|g_k\|(\|g_k - g_{k-1}\| + \|g_{k-1}(1 - (\|g_k\|/\|g_{k-1}\|))\| + t_k^*\|s_{k-1}\|)}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&= \frac{\|g_k\|(\|g_k - g_{k-1}\| + |1 - (\|g_k\|/\|g_{k-1}\|)|\|g_{k-1}\| + t_k^*\|s_{k-1}\|)}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&= \frac{\|g_k\|(\|g_k - g_{k-1}\| + |\|g_{k-1}\| - \|g_k\|| + t_k^*\|s_{k-1}\|)}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&\le \frac{\|g_k\|(\|g_k - g_{k-1}\| + \|g_{k-1} - g_k\| + t_k^*\|s_{k-1}\|)}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&= \frac{\|g_k\|(2\|g_k - g_{k-1}\| + t_k^*\|s_{k-1}\|)}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \le \frac{\|g_k\|(2L\|s_{k-1}\| + t_k^*\|s_{k-1}\|)}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&= \frac{(2L + t_k^*)\|g_k\|\|s_{k-1}\|}{\theta\alpha_{k-1}\|d_{k-1}\|^2} = \frac{(2L + t_k^*)\|g_k\|\alpha_{k-1}\|d_{k-1}\|}{\theta\alpha_{k-1}\|d_{k-1}\|^2} \\
&= \frac{(2L + t_k^*)\|g_k\|}{\theta\|d_{k-1}\|}.
\end{aligned} \tag{39}$$

Using $t_k^* \in (0, 1)$ and (38) and (39) in (36), we obtain

$$\begin{aligned}
\|d_k\|^2 &\le \|g_k\|^2 + 2\frac{\|g_k\|^2}{\lambda} + \frac{(2L + t_k^*)^2\|g_k\|^2}{\theta^2\|d_{k-1}\|^2}\|d_{k-1}\|^2 \\
&= \|g_k\|^2 + 2\frac{\|g_k\|^2}{\lambda} + \frac{(2L + t_k^*)^2}{\theta^2}\|g_k\|^2 \\
&= \left(1 + \frac{2}{\lambda} + \frac{(2L + t_k^*)^2}{\theta^2}\right)\|g_k\|^2 = \left(\frac{\lambda + 2}{\lambda} + \frac{(2L + t_k^*)^2}{\theta^2}\right)\|g_k\|^2 \\
&= \frac{(\lambda + 2)\theta^2 + \lambda(2L + t_k^*)^2}{\lambda\theta^2}\|g_k\|^2.
\end{aligned} \tag{40}$$

Next, dividing both sides of (40) by $\|g_k\|^4$ and using (35), it can be concluded that

$$\begin{aligned}
\frac{\|d_k\|^2}{\|g_k\|^4} &\le \frac{(\lambda + 2)\theta^2 + \lambda(2L + t_k^*)^2}{\lambda\theta^2} \cdot \frac{1}{c_1^2}, \\
\frac{\|g_k\|^4}{\|d_k\|^2} &\ge \frac{\lambda\theta^2 \cdot c_1^2}{(\lambda + 2)\theta^2 + \lambda(2L + t_k^*)^2}.
\end{aligned} \tag{41}$$

The inequalities in (41) imply

$$\sum_{k=0}^{\infty}\frac{\|g_k\|^4}{\|d_k\|^2} \ge \sum_{k=0}^{\infty}\frac{\lambda\theta^2 \cdot c_1^2}{(\lambda + 2)\theta^2 + \lambda(2L + t_k^*)^2} = \infty. \tag{42}$$

Therefore, $\|g_k\| \ge c_1$ causes a contradiction with Lemma 2.

3. Numerical Experiments

The implementation of the EDL method is based on Algorithm 2. This section is intended to analyze and compare the numerical results obtained by the EDL method and four variants of the MHSDL class methods (6). These variants are defined by $t \equiv t_{k3}$, $t \equiv t_{k4}$, $t \equiv t_{k5}$, and $t \equiv t_{k6}$ and denoted, respectively, as MHSDL3, MHSDL4, MHSDL5, and MHSDL6. The obtained results are not compared with the values t_{k1} and t_{k2}, because in [16], the authors have already shown that t_{k3} and t_{k4} initiate better numerical performances compared to t_{k1} and t_{k2}.

The codes used in the testing experiments for the above methods are written in MATLAB R2017a and executed on the Intel Core i3 2.0 GHz workstation with the Windows 10 operating system. Three important criteria are analyzed in each individual test case: number of iterations (NI), number of function evaluations (NFE), and processor time (CPU).

The numerical experiment is performed using 28 test functions presented in [24], where much of the problems are taken over from the CUTEr collection [25]. All methods used in the testing of an arbitrary objective function start from the same initialization x_0. Each function is tested 10 times with gradually increasing dimensions $n = 100$, 500, 1000, 3000, 5000, 7000, 8000, 10000, 15000, and 20000.

The uniform terminating criteria for each of the five considered algorithms (EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6) are

$$\begin{aligned}
\|g_k\| &\le \varepsilon, \\
\frac{|f(x_{k+1}) - f(x_k)|}{1 + |f(x_k)|} &\le \delta,
\end{aligned} \tag{43}$$

where $\varepsilon = 10^{-6}$ and $\delta = 10^{-16}$. The backtracking line search is based on the parameters $\omega = 0.0001$ and $\varphi = 0.8$ for all five algorithms. Specific parameters used only in the MHSDL6 method are defined as $C = 1$, $\upsilon = 0.26$, and $r = r_k = \upsilon\|g_{k-1}\|$.

Summary numerical results for EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods, executed on 28 test functions, are arranged in Tables 1–3. Tables 1–3 show the numerical outcomes corresponding to all three criteria (NI, NFE, and CPU) for the EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods.

We utilized the performance profile given in [26] to compare numerical results for three criteria (NI, NFE, and CPU) generated by five methods (EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6). The upper curve of the selected performance profile corresponds to the method that shows the best performance.

Figures 1–3 plot the performance profiles for the numerical values included in Tables 1–3, respectively. Figure 1 presents the performance profiles of the NI criterion generated by the EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods. In this figure, it is noticeable that EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods solved all tested functions, wherein the EDL method shows the best performances in 57.14% of test functions compared with MHSDL3 (25.00%), MHSDL4 (0.00%), MHSDL5 (0.00%), and MHSDL6 (17.86%). From Figure 1, it is observable that the graph of the EDL method comes first to the top, which means that the EDL outperforms other considered methods with respect to the NI.

Figure 2 presents the performance profiles of the NFE of the EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods. It is observable that EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 generated solutions to all tested cases, and the EDL method is the best in 67.86% of the functions compared with MHSDL3 (17.86%), MHSDL4 (0.00%), MHSDL5 (0.00%), and MHSDL6 (14.28%). From Figure 2, it is observed that the EDL graph first comes to the top, which confirms that the EDL is the winner with respect to the NFE.

Figure 3 contains graphs of the performance profiles corresponding to the CPU time of the EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 methods. It is obvious that EDL, MHSDL3, MHSDL4, MHSDL5, and MHSDL6 solved all tested functions. Further analysis gives that the EDL method is the winner in 67.86% of the test cases compared with MHSDL3 (17.86%), MHSDL4 (0.00%), MHSDL5 (0.00%), and MHSDL6 (14.28%). Figure 3 demonstrates that the graph of the EDL method first comes to level 1, which indicates its superiority with respect to the CPU time.

From the previous analysis of the results shown in Tables 1–3 and Figures 1–3, it can be concluded that the EDL method produces superlative results in terms of all three basic metrics: NI, NFE, and CPU.

4. Conclusion

A novel rule which determines the value $t(k)$ of the parameter t in each iteration of the Dai-Liao-type CG method is presented. The proposed expression for defining $t(k)$ is denoted by t_k^*. Considering $t = t_k^*$ in (6), a novel variant of the Dai-Liao CG parameter β_k^{EDL} is defined and a novel Effective Dai-Liao (EDL) conjugate gradient method is proposed. The convergence of the EDL method is investigated, and the global convergence on a class of uniformly convex functions is established. By numerical testing, we have shown that there is a significant influence of the scalar size of t_k^* on the convergence speed of the EDL method. Numerical comparisons on large-scale unconstrained optimization test functions of different structures and complexities confirm the computational efficiency of the algorithm EDL and its superiority over the previously known DL CG variants, such as MHSDL3, MHSDL4, MHSDL5, and MHSDL6. During the testing, we tracked the number of iterations (NI), number of function evaluations (NFE), and spanned processor time (CPU) performances for each function and each method. Analysis of the obtained performance profiles introduced by Dolan and Moré revealed that the EDL method is the most efficient.

We are convinced that the obtained results will be a motivation for further research in defining new values of the parameter t_k in the Dai-Liao CG methods. Future research would include research in finding some more efficient rules to calculate the parameter t_k during the iterative process. We hope that our proposal of the new expression for defining the parameter t will initiate further research in that direction. It is evident that finding novel approaches in defining different values of t and the conjugate gradient parameter β_k is an inexhaustible topic for scientific research, and our approach is only one possible direction in this research.

Data Availability

Data will be provided on request to the first author.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgments

The research was supported by the National Natural Science Foundation of China (Grant Nos. 11971142, 11871202, 61673169, 11701176, 11626101, and 11601485).

References

[1] Y. -H. Dai and L. -Z. Liao, “New conjugacy conditions and related nonlinear conjugate gradient methods,” *Applied Mathematics and Optimization*, vol. 43, no. 1, pp. 87–101, 2001.

[2] Y. Cheng, Q. Mou, X. Pan, and S. Yao, “A sufficient descent conjugate gradient method and its global convergence,” *Optimization Methods and Software*, vol. 31, no. 3, pp. 577–590, 2016.

[3] I. E. Livieris and P. Pintelas, “A descent Dai-Liao conjugate gradient method based on a modified secant equation and its global convergence,” *ISRN Computational Mathematics*, vol. 2012, Article ID 435495, 8 pages, 2012.

[4] M. R. Peyghami, H. Ahmadzadeh, and A. Fazli, “A new class of efficient and globally convergent conjugate gradient methods in the Dai-Liao family,” *Optimization Methods and Software*, vol. 30, no. 4, pp. 843–863, 2015.

[5] H. Yabe and M. Takano, “Global convergence properties of nonlinear conjugate gradient methods with modified secant condition,” *Computational Optimization and Applications*, vol. 28, no. 2, pp. 203–225, 2004.

[6] S. Yao and B. Qin, "A hybrid of DL and WYL nonlinear conjugate gradient methods," *Abstract and Applied Analysis*, vol. 2014, Article ID 279891, 9 pages, 2014.

[7] S. Yao, X. Lu, and Z. Wei, "A conjugate gradient method with global convergence for large-scale unconstrained optimization problems," *Journal of Applied Mathematics*, vol. 2013, Article ID 730454, 9 pages, 2013.

[8] Y. Zheng and B. Zheng, "Two new Dai-Liao-type conjugate gradient methods for unconstrained optimization problems," *Journal of Optimization Theory and Applications*, vol. 175, no. 2, pp. 502–509, 2017.

[9] W. Zhou and L. Zhang, "A nonlinear conjugate gradient method based on the MBFGS secant condition," *Optimization Methods and Software*, vol. 21, no. 5, pp. 707–714, 2006.

[10] W. Hu, J. Wu, and G. Yuan, "Some modified Hestenes-Stiefel conjugate gradient algorithms with application in image restoration," *Applied Numerical Mathematics*, vol. 158, pp. 360–376, 2020.

[11] G. Yuan, T. Li, and W. Hu, "A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems," *Applied Numerical Mathematics*, vol. 147, pp. 129–141, 2020.

[12] N. Andrei, "Open problems in nonlinear conjugate gradient algorithms for unconstrained optimization," *Bulletin of the Malaysian Mathematical Sciences Society*, vol. 34, no. 2, pp. 319–330, 2011.

[13] W. W. Hager and H. Zhang, "A new conjugate gradient method with guaranteed descent and an efficient line search," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, 2005.

[14] W. W. Hager and H. Zhang, "Algorithm 851," *ACM Transactions on Mathematical Software*, vol. 32, no. 1, pp. 113–137, 2006.

[15] Y. -H. Dai and C. -X. Kou, "A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 296–320, 2013.

[16] S. Babaie-Kafaki and R. Ghanbari, "The Dai-Liao nonlinear conjugate gradient method with optimal parameter choices," *European Journal of Operational Research*, vol. 234, no. 3, pp. 625–630, 2014.

[17] N. Andrei, "A Dai-Liao conjugate gradient algorithm with clustering of eigenvalues," *Numerical Algorithms*, vol. 77, no. 4, pp. 1273–1282, 2018.

[18] M. Lotfi and S. M. Hosseini, "An efficient Dai-Liao type conjugate gradient method by reformulating the CG parameter in the search direction equation," *Journal of Computational and Applied Mathematics*, vol. 371, article 112708, 2020.

[19] X. Li and Q. Ruan, "A modified PRP conjugate gradient algorithm with trust region for optimization problems," *Numerical Functional Analysis and Optimization*, vol. 32, no. 5, pp. 496–506, 2011.

[20] N. Andrei, "An acceleration of gradient descent algorithm with backtracking for unconstrained optimization," *Numerical Algorithms*, vol. 42, no. 1, pp. 63–73, 2006.

[21] P. S. Stanimirovic and M. B. Miladinovic, "Accelerated gradient descent methods with line search," *Numerical Algorithms*, vol. 54, no. 4, pp. 503–520, 2010.

[22] W. Cheng, "A two-term PRP-based descent method," *Numerical Functional Analysis and Optimization*, vol. 28, no. 11–12, pp. 1217–1230, 2007.

[23] G. Zoutendijk, "Nonlinear programming, computational methods," in *Integer and Nonlinear Programming, North-Holland*, J. Abadie, Ed., pp. 37–86, North-Holland, Amsterdam, 1970.

[24] N. Andrei, "An unconstrained optimization test functions collection," *Advanced Modeling and Optimization*, vol. 10, no. 1, pp. 147–161, 2008.

[25] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, "CUTE: constrained and unconstrained testing environments," *ACM Transactions on Mathematical Software*, vol. 21, no. 1, pp. 123–160, 1995.

[26] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.