Hindawi
Mathematical Problems in Engineering
Volume 2018, Article ID 4729318, 11 pages
https://doi.org/10.1155/2018/4729318

Research Article

A Conjugate Gradient Algorithm under Yuan-Wei-Lu Line Search Technique for Large-Scale Minimization Optimization Models

Xiangrong Li [1], Songhua Wang [2], Zhongzhou Jin [3], and Hongtruong Pham [4]

[1] *College of Mathematics and Information Science, Guangxi University, Nanning, Guangxi 530004, China*
[2] *School of Mathematics and Statistics, Baise University, Baise, Guangxi 533000, China*
[3] *Business School, Guangxi University, Nanning, Guangxi 530004, China*
[4] *Thai Nguyen University of Economics and Business Administration, Thai Nguyen, Vietnam*

Correspondence should be addressed to Xiangrong Li; xrli68@163.com

Received 17 October 2017; Accepted 27 November 2017; Published 15 January 2018

Academic Editor: Guillermo Cabrera-Guerrero

This paper gives a modified Hestenes and Stiefel (HS) conjugate gradient algorithm under the Yuan-Wei-Lu inexact line search technique for large-scale unconstrained optimization problems, where the proposed algorithm has the following properties: (1) the new search direction possesses not only a sufficient descent property but also a trust region feature; (2) the presented algorithm has global convergence for nonconvex functions; (3) the numerical experiment showed that the new algorithm is more effective than similar algorithms.

1. Introduction

Consider the minimization optimization models defined by

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

where the function $f : \Re^n \to \Re$ and $f \in C^2$. There exist many good algorithms for (1), such as the quasi-Newton methods [1] and the conjugate gradient methods [2–5], where the iterative formula of the conjugate gradient algorithm for (1) is designed by

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

where x_k is the kth iterative point, α_k is the steplength, and d_k is the so-called conjugate gradient search direction with

$$d_{k+1} = \begin{cases} -g(x_{k+1}) + \beta_k d_k, & \text{if } k \geq 1, \\ -g(x_{k+1}), & \text{if } k = 0, \end{cases} \tag{3}$$

where β_k is a scalar determined from different conjugate gradient formulas and the HS method [3] is one of the most well-known conjugate gradient methods, which is

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \tag{4}$$

where $y_k = g_{k+1} - g_k$, $g_{k+1} = \nabla f(x_{k+1})$ and $g_k = \nabla f(x_k)$. The HS method has good numerical results for (1); however, the convergent theory is not interesting especially for the nonconvex function. At present, there exist many good conjugate gradients (see [6–8], etc.). Yuan, Wei, and Lu [9] gave a modified weak Wolfe-Powell (we called it YWL) line search for steplength α_k designed by

$$f(x_k + \alpha_k d_k) \leq f_k + \delta \alpha_k g_k^T d_k + \alpha_k \min\left[-\delta_1 g_k^T d_k, \delta \frac{\alpha_k}{2} \|d_k\|^2\right], \tag{5}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k + \min\left[-\delta_1 g_k^T d_k, \delta \alpha_k \|d_k\|^2\right], \tag{6}$$

where $\delta \in (0, 1/2)$, $\delta_1 \in (0, \delta)$, $\sigma \in (\delta, 1)$, and $\|\cdot\|$ denotes the Euclidean norm. It is well known that there exist two open problems which are the global convergence of the normal BFGS method and the global convergence of the PRP method for nonconvex functions under the inexact line search technique, where the first problem is regarded as one of the most difficult one thousand mathematical problems of the 20th century [10]. Yuan et al. [9] partly solved these two open problems under the YWL technique, and the numerical performance shows that the YWL technique is more competitive than the normal weak Wolfe-Powell technique. Further study work can be found in their paper [11]. By (5), it is not difficult to see that the YWL conditions are equivalent to the weak Wolfe-Powell (WWP) conditions if $-\delta_1 g_k^T d_k < \delta(\alpha_k/2)\|d_k\|^2$ holds, which implies that the YWL technique includes the WWP technique in some sense. Motivated by the above observations, we will make a further study and propose a new algorithm for (1). The main features of this paper are as follows:

(i) A modified HS conjugate gradient formula is given, which has not only a sufficient descent property but also a trust region feature.

(ii) The global convergence of the given HS conjugate gradient algorithm for nonconvex functions is established.

(iii) Numerical results show that the new HS conjugate gradient algorithm under the YWL line search technique is better than the normal weak Wolfe-Powell technique.

This paper is organized as follows. In Section 2, a modified HS conjugate gradient algorithm is introduced. The global convergence of the given algorithm for nonconvex functions is established in Section 3 and numerical results are reported in Section 4.

2. Motivation and Algorithm

The nonlinear conjugate gradient algorithm is simple and has low memory requirement properties and is very effective for large-scale optimization problems, where the HS method is one of the most effective methods. However, the normal HS method has good numerical performance but fails in the convergence of nonconvex functions under the inexact line search technique. In order to overcome this shortcoming, a modified HS formula is defined by

$$d_{k+1} = \begin{cases} -g_{k+1} + \dfrac{g_{k+1}^T y_k d_k - d_k^T g_{k+1} y_k}{\psi_1 \|d_k\|^2 + 2\psi_2 \|d_k\| \|y_k\| + \|g_k\|^2 + \psi_3 \|y_k\|^2}, & \text{if } k \geq 1, \\ -g_{k+1}, & \text{if } k = 0, \end{cases} \tag{7}$$

where $y_k = g_{k+1} - g_k$ and ψ_1, ψ_2, and ψ_3 are positive constants. This formula is inspired by the idea of these two papers [6, 8]. In recent years, lots of scholars like to study the three-term conjugate gradient formula because of its good properties [7]. In the next section, we will prove that the new formula possesses not only a sufficient descent property but also a trust region feature. The sufficient descent property is good for the convergence and the trust region makes the convergence easy to prove. Now, we give the steps of the proposed algorithm as follows.

Algorithm 1 (the modified three-term HS conjugate gradient algorithm (M-TT-HS-A)).

Step 1. $x_1 \in \mathfrak{R}^n$, $\epsilon \in (0, 1)$, $\delta \in (0, 1/2)$, $\delta_1 \in (0, \delta)$, $\sigma \in (\delta, 1)$, $\psi_1 > 0$, $\psi_2 > 0$, $\psi_3 > 0$. Let $k = 1$ and $d_1 = -g(x_1)$.

Step 2. If $\|g_k\| \leq \epsilon$ holds, stop.

Step 3. Find α_k by the YWL line search satisfying (5) and (6).

Step 4. Let $x_{k+1} = x_k + \alpha_k d_k$.

Step 5. Compute the direction d_{k+1} by (7).

Step 6. The algorithm stops if $\|g_{k+1}\| \leq \epsilon$.

Step 7. Let $k = k + 1$ and go to Step 3.

3. Sufficient Descent Property, Trust Region Feature, and Global Convergence

This section will prove some properties of Algorithm 1.

Lemma 2. *The search direction d_k is designed by (7); the following two relations hold:*

$$g_k^T d_k = -\|g_k\|^2, \tag{8}$$

$$\|d_k\| \leq \psi^* \|g_k\|, \tag{9}$$

where $\psi^* > 0$ is a constant.

Proof. If $k = 1$, it is easy to have (8) and (9). If $k \geq 1$, by formula (7), we have

$$\begin{aligned} g_{k+1}^T d_{k+1} &= g_{k+1}^T \Bigg[-g_{k+1} \\ &\quad + \frac{g_{k+1}^T y_k d_k - d_k^T g_{k+1} y_k}{\psi_1 \|d_k\|^2 + 2\psi_2 \|d_k\| \|y_k\| + \|g_k\|^2 + \psi_3 \|y_k\|^2} \Bigg] \\ &= -\|g_{k+1}\|^2 \\ &\quad + \frac{g(x_{k+1})^T y_k g_{k+1}^T d_k - d_k^T g(x_{k+1}) g_{k+1}^T y_k}{\psi_1 \|d_k\|^2 + 2\psi_2 \|d_k\| \|y_k\| + \|g_k\|^2 + \psi_3 \|y_k\|^2} \end{aligned}$$

$$= -\|g_{k+1}\|^2,$$

$$\begin{aligned}\|d_{k+1}\| &= \Bigg\| -g(x_{k+1}) \\ &+ \frac{g_{k+1}^T y_k d_k - d_k^T g_{k+1} y_k}{\psi_1 \|d_k\|^2 + 2\psi_2 \|d_k\| \|y_k\| + \|g_k\|^2 + \psi_3 \|y_k\|^2} \Bigg\| \\ &\le \|g_{k+1}\| \\ &+ \frac{\|g(x_{k+1})\| \|y_k\| \|d_k\| + \|d_k\| \|g(x_{k+1})\| \|y_k\|}{\psi_1 \|d_k\|^2 + 2\psi_2 \|d_k\| \|y_k\| + \|g_k\|^2 + \psi_3 \|y_k\|^2} \\ &\le \|g_{k+1}\| + \frac{2\|g(x_{k+1})\| \|y_k\| \|d_k\|}{2\psi_2 \|d_k\| \|y_k\|} \le \left(1 + \frac{1}{\psi_2}\right) \\ &\cdot \|g_{k+1}\|.\end{aligned} \tag{10}$$

Then, (8) holds as well as (9) by letting $\psi^* \in [1 + 1/\psi_2, +\infty)$. This completes the proof. □

Inequality (8) shows that the new formula has a sufficient descent property and inequality (9) proves that the new formula possesses a trust region feature. Both of these properties (8) and (9) are good theory characters and they play an important role in the global convergence of a conjugate gradient algorithm. The following global convergence theory will explain all this.

The following general assumptions are needed.

Assumption A. (i) The defined level set $L_0 = \{x \mid f(x) \le f(x_1)\}$ is bounded.

(ii) The objective function $f(x)$ is bounded below, twice continuously differentiable, and is Lipschitz continuous; namely, the following inequality is true:

$$\|g(x) - g(y)\| \le L\|x - y\|, \quad x, y \in \Re^n, \tag{11}$$

where $L > 0$ is the Lipschitz constant.

By Lemma 2 and Assumption A, similar to [9], it is not difficult to show that the YWL line search technique is reasonable and Algorithm 1 is well defined. Here, we do not state it anymore. Now, we prove the global convergence of Algorithm 1 for nonconvex functions.

Theorem 3. *Let Assumption A hold, and the iterate sequence $\{x_k, \alpha_k, d_k, g(x_k)\}$ is generated by M-TT-HS-A. Then, the relation*

$$\lim_{k\to\infty} \|g_k\| = 0 \tag{12}$$

is true.

Proof. By (5), (8), and (9), we obtain

$$\begin{aligned} f(x_k + \alpha_k d_k) &\le f(x_k) + \delta\alpha_k g_k^T d_k \\ &\quad + \alpha_k \min\left[-\delta_1 g_k^T d_k, \delta\frac{\alpha_k}{2}\|d_k\|^2\right] \\ &\le f(x_k) + \delta\alpha_k g_k^T d_k - \alpha_k \delta_1 g_k^T d_k \\ &\le f(x_k) + \alpha_k(\delta - \delta_1) g_k^T d_k \\ &\le f(x_k) - \alpha_k(\delta - \delta_1)\|g_k\|^2. \end{aligned} \tag{13}$$

Summing these inequalities for $k = 0$ to ∞ and using Assumption A (ii) generate

$$\sum_{k=0}^{\infty} (\gamma_1 - \gamma)\alpha_k(\delta - \delta_1)\|g_k\|^2 \le f(x_1) - f_\infty < +\infty. \tag{14}$$

Inequality (14) implies that

$$\lim_{k\to\infty} \alpha_k \|g_k\|^2 = 0 \tag{15}$$

is true. By (6) and (8) again, we get

$$\begin{aligned} g(x_k + \alpha_k d_k)^T d_k &\ge \sigma g_k^T d_k \\ &\quad + \min\left[-\delta_1 g_k^T d_k, \delta\alpha_k \|d_k\|^2\right] \\ &\ge \sigma g_k^T d_k. \end{aligned} \tag{16}$$

Thus, the inequality

$$\begin{aligned} -(\sigma - 1)\|g_k\|^2 &\le (\sigma - 1) g_k^T d_k \\ &\le [g(x_k + \alpha_k d_k) - g(x_k)]^T d_k \\ &\le \|g(x_k + \alpha_k d_k) - g(x_k)\| \|d_k\| \\ &\le \alpha_k L \|d_k\|^2 \end{aligned} \tag{17}$$

holds, where the first inequality follows (8) and the last inequality follows (11). Then, we have

$$\alpha_k \ge \frac{(1 - \sigma)\|g_k\|^2}{L\|d_k\|^2} \ge \frac{(1 - \sigma)\|g_k\|^2}{L(\psi^*)^2\|g_k\|^2} = \frac{(1 - \sigma)}{L(\psi^*)^2}. \tag{18}$$

By (15) and (18), we have

$$\lim_{k\to\infty} \|g_k\|^2 = 0. \tag{19}$$

Therefore, we get (12) and the proof is complete. □

4. Numerical Results Performance

This section will give numerical results of Algorithm 1 and the similar algorithms for comparing them. We will give another two algorithms for comparison; they are listed as follows.

Algorithm 2 (the normal three-term formula [8] under the YWL technique).

Step 1. $x_1 \in \mathfrak{R}^n$, $\epsilon \in (0,1)$, $\delta \in (0,1/2)$, $\delta_1 \in (0,\delta)$, $\sigma \in (\delta,1)$, $\psi_1 > 0$, $\psi_2 > 0$, $\psi_3 > 0$. Let $k = 1$ and $d_1 = -g(x_1)$.

Step 2. If $\|g_k\| \le \epsilon$ holds, stop.

Step 3. Find α_k by the YWL line search satisfying (5) and (6).

Step 4. Let $x_{k+1} = x_k + \alpha_k d_k$.

Step 5. Compute the direction d_{k+1} by

$$d_{k+1} = \begin{cases} -g_{k+1} + \dfrac{g_{k+1}^T y_k d_k - d_k^T g_{k+1} y_k}{\|g_k\|^2}, & \text{if } k \ge 1, \\ -g_{k+1}, & \text{if } k = 0. \end{cases} \quad (20)$$

Step 6. The algorithm stops if $\|g_{k+1}\| \le \epsilon$.

Step 7. Let $k = k+1$ and go to Step 3.

Algorithm 3 (the normal three-term formula [8] under the WWP technique).

Step 1. $x_1 \in \mathfrak{R}^n$, $\epsilon \in (0,1)$, $\delta \in (0,1/2)$, $\sigma \in (\delta,1)$. Let $k = 1$ and $d_1 = -g(x_1)$.

Step 2. If $\|g_k\| \le \epsilon$ holds, stop.

Step 3. Find α_k by the WWP line search satisfying

$$\begin{gathered} f(x_k + \alpha_k d_k) \le f_k + \delta \alpha_k g_k^T d_k, \\ g(x_k + \alpha_k d_k)^T d_k \ge \sigma g_k^T d_k. \end{gathered} \quad (21)$$

Step 4. Let $x_{k+1} = x_k + \alpha_k d_k$.

Step 5. Compute the direction d_{k+1} by (20).

Step 6. The algorithm stops if $\|g_{k+1}\| \le \epsilon$.

Step 7. Let $k = k+1$ and go to Step 3.

4.1. Problems and Experiment. The following are some notes.

Test Problems. These problems and the related initial points are listed in Table 1; the detailed problems can be found in Andrei [12], and some papers also use these problems [13].

Experiments. Codes are run on Intel(R) Xeon(R) CPU, E5507 @2.27 GHz, and 6.00 GB memory and Windows 7 operation system and written by MATLAB R2009a.

Parameters. $\delta = 0.1$, $\delta_1 = 0.05$, $\sigma = 0.9$, and $\psi_1 = \psi_2 = \psi_3 = 0.001$.

Dimension. Large-scale dimensions $n = 3000, 6000, 12000$, and 30000.

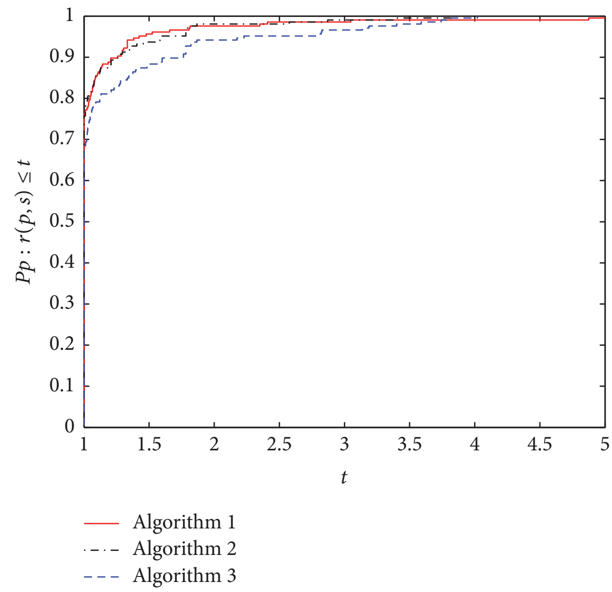

Figure 1: NI performance of these methods.

Stop Rules. Himmeblau stop rule: if $|f(x_k)| > 1e-5$, let stop1 $= |f(x_k) - f(x_{k+1})|/|f(x_k)|$; otherwise, set stop1 $= |f(x_k) - f(x_{k+1})|$. If stop1 $< 1e-5$ or $\|g(x_k)\| < 1e-6$ holds, the program stops.

Other Cases. The line search technique accepts α_k if the searching number is more than 6 and the algorithm will stop if the total iteration number is larger than 800.

The numerical results are listed in Table 2, where

"Number" is the tested problems number;

"Dim." is the problems dimension;

"NI" is the total iteration number;

"CPU" is the system CPU time in seconds;

"NFG" is the total number of functions and gradients.

4.2. Results and Discussion. We use the tool of Dolan and Moré [14] to analyze the efficiency of the three given algorithms. Figures 1 and 2 show that the performance of Algorithm 1 is the best and that Algorithm 1 has the best robust property among those three methods and Algorithm 2 is better than Algorithm 3, which shows that the given formula (7) is competitive to the normal three-term conjugate gradient formula (20) and the YWL line search technique is more effective than the norm WWP technique, and all of these conclusions are coincident with the results of [9]. Algorithm 1 in Figure 3 is competitive to the other two algorithms and it has the best robust property. It is not difficult to see that Figure 3 shows that Algorithm 1 is not so good and we think the reason is formula (7) or the YWL technique since more information is needed and hence more CPU time is necessary.

Table 1: Test problems.

Number	Problem	x_0
(1)	Extended Freudenstein and Roth Function	$[0.5, -2, \ldots, 0.5, -2]$
(2)	Extended Trigonometric Function	$[0.2, 0.2, \ldots, 0.2]$
(3)	Extended Rosenbrock Function	$[-1.2, 1, -1.2, 1, \ldots, -1.2, 1]$
(4)	Extended Beale Function	$[1, 0.8, \ldots, 1, 0.8]$
(5)	Raydan 1 Function	$[1, 1, \ldots, 1]$
(6)	Raydan 2 Function	$[1, 1, \ldots, 1]$
(7)	Diagonal 1 Function	$[1/n, 1/n, \ldots, 1/n]$
(8)	Diagonal 3 Function	$[1, 1, \ldots, 1]$
(9)	Hager Function	$[1, 1, \ldots, 1]$
(10)	Generalized Tridiagonal 1 Function	$[2, 2, \ldots, 2]$
(11)	Extended Tridiagonal 1 Function	$[2, 2, \ldots, 2]$
(12)	Extended Three Exponential Terms Function	$[0.1, 0.1, \ldots, 0.1]$
(13)	Diagonal 4 Function	$[1, 1, \ldots, 1, 1]$
(14)	Diagonal 5 Function	$[1.1, 1.1, \ldots, 1.1]$
(15)	Extended Himmelblau Function	$[1, 1, \ldots, 1]$
(16)	Generalized PSC1 Function	$[3, 0.1, \ldots, 3, 0.1]$
(17)	Extended PSC1 Function	$[3, 0.1, \ldots, 3, 0.1]$
(18)	Extended Block Diagonal BD1 Function	$[0.1, 0.1, \ldots, 0.1]$
(19)	Extended Maratos Function	$[1.1, 0.1, \ldots, 1.1, 0.1]$
(20)	Extended Cliff Function	$[0, -1, \ldots, 0, -1]$
(21)	Extended Wood Function	$[-3, -1, -3, -1, \ldots, -3, -1]$
(22)	Extended Quadratic Penalty QP1 Function	$[1, 1, \ldots, 1]$
(23)	Extended Quadratic Penalty QP2 Function	$[1, 1, \ldots, 1]$
(24)	A Quadratic Function QF2 Function	$[0.5, 0.5, \ldots, 0.5]$
(25)	Extended EP1 Function	$[1.5, 1.5, \ldots, 1.5]$
(26)	Extended Tridiagonal-2 Function	$[1, 1, \ldots, 1]$
(27)	BDQRTIC Function (CUTE)	$[1, 1, \ldots, 1]$
(28)	ARWHEAD Function (CUTE)	$[1, 1, \ldots, 1]$
(29)	NONDIA (Shanno-78) Function (CUTE)	$[-1, -1, \ldots, -1]$
(30)	DQDRTIC Function (CUTEr)	$[3, 3, 3, \ldots, 3]$
(31)	EG2 Function (CUTE)	$[1, 1, 1, \ldots, 1]$
(32)	DIXMAANA Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(33)	DIXMAANB Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(34)	DIXMAANC Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(35)	Partial Perturbed Quadratic Function	$[0.5, 0.5, \ldots, 0.5]$
(36)	Broyden Tridiagonal Function	$[-1, -1, \ldots, -1]$
(37)	EDENSCH Function (CUTE)	$[0, 0, \ldots, 0]$
(38)	LIARWHD Function (CUTEr)	$[4, 4, \ldots, 4]$
(39)	DIAGONAL 6 Function	$[1, 1, \ldots, 1]$
(40)	DIXON3DQ Function (CUTE)	$[-1, -1, \ldots, -1]$
(41)	DIXMAANF Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(42)	DIXMAANG Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(43)	DIXMAANH Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(44)	DIXMAANI Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(45)	DIXMAANJ Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(46)	DIXMAANK Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(47)	DIXMAANL Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(48)	DIXMAAND Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(49)	ENGVAL1 Function (CUTE)	$[2, 2, 2, \ldots, 2]$
(50)	Extended DENSCHNB Function (CUTE)	$[1, 1, \ldots, 1]$
(51)	SINQUAD Function (CUTE)	$[0.1, 0.1, \ldots, 0.1]$
(52)	Partial Perturbed Quadratic PPQ2 Function	$[0.5, 0.5, \ldots, 0.5]$

Table 2: Numerical results.

Number	Dim.	Algorithm 1			Algorithm 2			Algorithm 3		
		NI	NFG	CPU	NI	NFG	CPU	NI	NFG	CPU
(1)	3000	15	38	0.093601	20	56	0.140401	20	56	0.124801
	6000	24	66	0.358802	21	59	0.280802	21	59	0.249602
	12000	22	59	0.577204	17	46	0.405603	17	46	0.405603
	30000	22	59	1.357209	17	46	0.998406	17	46	1.029607
(2)	3000	74	163	0.327602	74	163	0.343202	83	181	0.358802
	6000	81	176	0.702005	81	176	0.686404	81	176	0.655204
	12000	83	182	1.388409	83	182	1.341609	83	182	1.357209
	30000	88	194	3.572423	88	194	3.588023	89	196	3.478822
(3)	3000	88	262	0.124801	99	283	0.124801	73	216	0.093601
	6000	83	277	0.234002	112	329	0.265202	112	329	0.249602
	12000	78	272	0.390002	16	66	0.078001	16	66	0.093601
	30000	49	119	0.483603	46	182	0.561604	46	182	0.468003
(4)	3000	32	87	0.109201	21	57	0.093601	21	57	0.0624
	6000	24	65	0.171601	27	74	0.187201	27	74	0.187201
	12000	23	62	0.312002	23	63	0.265202	26	69	0.343202
	30000	24	67	0.748805	29	78	0.858005	25	66	0.717605
(5)	3000	23	51	0.0624	23	51	0.0312	23	51	0.0624
	6000	23	51	0.093601	23	51	0.093601	23	51	0.0624
	12000	23	51	0.171601	23	51	0.156001	23	51	0.124801
	30000	23	51	0.390002	23	51	0.358802	23	51	0.312002
(6)	3000	12	26	0.0156	12	26	0.0156	12	26	0.0156
	6000	12	26	0.0468	12	26	0.0468	12	26	0.0312
	12000	12	26	0.093601	12	26	0.0624	12	26	0.078
	30000	12	26	0.171601	12	26	0.171601	12	26	0.156001
(7)	3000	2	9	0.0312	2	9	0.0312	2	9	0.0312
	6000	2	9	0	2	9	0	2	9	0
	12000	2	9	0	2	9	0.0312	2	9	0.0312
	30000	2	9	0.0312	2	9	0.0468	2	9	0.0468
(8)	3000	17	36	0.0624	17	36	0.0468	17	36	0.0624
	6000	19	40	0.093601	19	40	0.109201	19	40	0.078
	12000	19	40	0.156001	19	40	0.156001	19	40	0.171601
	30000	19	40	0.405603	19	40	0.405603	19	40	0.374402
(9)	3000	24	113	0.078001	24	113	0.078	24	113	0.078
	6000	22	52	0.093601	21	50	0.109201	22	52	0.109201
	12000	2	9	0.0156	2	9	0.0156	2	9	0.0312
	30000	2	9	0.0624	2	9	0.0468	2	9	0.078
(10)	3000	6	15	0.655204	6	15	0.592804	6	15	0.592804
	6000	6	15	2.043613	6	15	1.981213	6	15	1.996813
	12000	6	15	7.207246	6	15	7.503648	6	15	7.394447
	30000	3	8	21.590538	3	8	21.528138	3	8	21.481338
(11)	3000	34	81	1.762811	35	86	1.653611	35	86	1.63801
	6000	47	114	6.520842	34	77	4.383628	36	88	4.74243
	12000	36	88	14.820095	37	93	15.241298	37	93	15.100897
	30000	43	102	92.492993	39	100	86.050152	39	100	85.83175
(12)	3000	14	30	0.0468	14	30	0.0624	14	30	0.0468
	6000	21	44	0.093601	21	44	0.109201	21	44	0.078001
	12000	24	50	0.202801	24	50	0.218401	24	50	0.218401
	30000	24	50	0.592804	24	50	0.483603	24	50	0.468003
(13)	3000	4	13	0.0312	3	10	0.0156	3	10	0
	6000	4	13	0	3	10	0	3	10	0
	12000	4	13	0.0312	3	10	0.0312	3	10	0
	30000	4	13	0.0468	3	10	0.0312	3	10	0.0312

Table 2: Continued.

Number	Dim.	Algorithm 1			Algorithm 2			Algorithm 3		
		NI	NFG	CPU	NI	NFG	CPU	NI	NFG	CPU
(14)	3000	3	9	0.0312	3	9	0.0312	3	9	0
	6000	3	9	0.0312	3	9	0.0312	3	9	0.0312
	12000	3	9	0.0624	3	9	0.0624	3	9	0.0312
	30000	3	9	0.078001	3	9	0.078	3	9	0.078001
(15)	3000	9	48	0.0468	9	48	0.0312	9	48	0.0312
	6000	128	292	0.312002	42	108	0.093601	42	108	0.078
	12000	24	114	0.140401	29	139	0.156001	29	139	0.156001
	30000	13	39	0.156001	16	45	0.171601	16	45	0.156001
(16)	3000	33	76	0.109201	33	76	0.093601	42	90	0.093601
	6000	32	73	0.187201	32	73	0.156001	41	88	0.187201
	12000	32	73	0.312002	32	73	0.312002	43	95	0.390003
	30000	33	78	0.811205	33	78	0.780005	37	86	0.889206
(17)	3000	9	44	0.124801	9	44	0.156001	9	43	0.124801
	6000	9	44	0.280802	9	44	0.265202	9	44	0.249602
	12000	9	44	0.499203	9	44	0.514803	9	44	0.514803
	30000	9	44	1.279208	9	44	1.232408	9	44	1.232408
(18)	3000	11	88	0.0624	11	87	0.0312	11	86	0.0312
	6000	11	88	0.093601	11	83	0.093601	11	83	0.093601
	12000	47	207	0.436803	20	115	0.202801	20	115	0.187201
	30000	39	158	0.873606	89	248	1.63801	31	134	0.733205
(19)	3000	28	56	0.0312	28	56	0.0468	28	56	0.0312
	6000	38	76	0.078	38	76	0.093601	38	76	0.0624
	12000	159	504	0.717605	170	594	0.733205	66	162	0.249602
	30000	7	16	0.078	7	16	0.0468	7	16	0.078
(20)	3000	97	212	0.234002	97	212	0.218401	97	212	0.202801
	6000	106	230	0.452403	106	230	0.421203	106	230	0.390003
	12000	93	206	0.733205	93	206	0.702005	93	206	0.686404
	30000	87	194	1.700411	87	194	1.54441	87	194	1.52881
(21)	3000	36	90	0.0468	36	90	0.0468	36	90	0.078
	6000	37	94	0.124801	37	94	0.093601	37	94	0.093601
	12000	34	88	0.202801	33	86	0.156001	33	86	0.156001
	30000	33	83	0.436803	36	92	0.405603	36	92	0.421203
(22)	3000	42	92	0.0624	42	92	0.078	42	92	0.078
	6000	41	90	0.109201	41	90	0.124801	41	90	0.093601
	12000	41	90	0.234002	41	90	0.187201	41	90	0.202801
	30000	45	98	0.514803	45	98	0.499203	45	98	0.483603
(23)	3000	42	88	0.109201	42	88	0.078001	42	88	0.078
	6000	71	146	0.296402	71	146	0.296402	71	146	0.280802
	12000	37	80	0.312002	37	80	0.265202	37	80	0.280802
	30000	52	110	1.029607	52	110	0.904806	52	110	0.920406
(24)	3000	3	7	0	3	7	0	3	7	0
	6000	3	7	0.0312	3	7	0.0312	3	7	0
	12000	2	5	0	2	5	0	2	5	0
	30000	2	5	0.0312	2	5	0.0312	2	5	0.0624
(25)	3000	4	8	0	4	8	0	4	8	0
	6000	5	10	0.0156	5	10	0.0156	5	10	0.0312
	12000	7	14	0.0312	7	14	0.0312	7	14	0.0624
	30000	10	20	0.140401	10	20	0.124801	10	20	0.124801
(26)	3000	12	24	0.0312	12	24	0.0468	12	24	0.0468
	6000	16	32	0.078001	16	32	0.0468	16	32	0.0468
	12000	21	42	0.124801	21	42	0.109201	21	42	0.109201
	30000	30	62	0.421203	30	62	0.405603	31	62	0.405603

Table 2: Continued.

Number	Dim.	Algorithm 1			Algorithm 2			Algorithm 3		
		NI	NFG	CPU	NI	NFG	CPU	NI	NFG	CPU
(27)	3000	18	57	1.341609	14	48	1.029607	14	48	0.998406
	6000	5	16	1.060807	8	28	1.778411	8	28	1.778411
	12000	5	16	3.494422	8	28	6.084039	8	28	6.068439
	30000	5	16	21.138135	17	55	77.9693	17	55	77.9537
(28)	3000	9	23	0.0312	9	26	0.0312	9	26	0.0156
	6000	12	30	0.0312	12	30	0.0468	12	30	0.0312
	12000	10	25	0.078	10	28	0.078001	10	28	0.0468
	30000	12	29	0.140401	11	30	0.109201	11	30	0.124801
(29)	3000	26	52	0.0468	26	52	0.0312	26	52	0.0312
	6000	27	54	0.078001	27	54	0.0468	27	54	0.0468
	12000	29	58	0.124801	29	58	0.109201	29	58	0.093601
	30000	23	46	0.249602	23	46	0.171601	23	46	0.202801
(30)	3000	49	119	0.0624	27	68	0.0624	27	68	0.0468
	6000	37	92	0.109201	26	65	0.0468	26	65	0.078
	12000	29	75	0.124801	28	69	0.124801	28	69	0.093601
	30000	32	77	0.296402	27	68	0.234002	27	68	0.280802
(31)	3000	4	21	0	4	21	0	4	21	0
	6000	4	21	0.0312	4	21	0.0156	4	21	0.0312
	12000	4	21	0.0312	4	21	0.0312	4	21	0.0468
	30000	4	21	0.093601	4	21	0.109201	4	21	0.109201
(32)	3000	22	48	0.296402	22	48	0.296402	22	48	0.265202
	6000	23	50	0.577204	23	50	0.561604	23	50	0.608404
	12000	24	52	1.201208	24	52	1.185608	24	52	1.185608
	30000	25	54	3.026419	25	54	3.04202	25	54	3.010819
(33)	3000	38	80	0.483603	38	80	0.483603	36	76	0.436803
	6000	39	82	0.951606	39	82	0.967206	37	78	0.889206
	12000	41	86	1.950013	41	86	1.965613	38	80	1.856412
	30000	42	88	5.054432	42	88	5.070032	39	82	4.69563
(34)	3000	66	136	0.842405	66	136	0.795605	66	136	0.826805
	6000	69	142	1.669211	69	142	1.669211	69	142	1.63801
	12000	72	148	3.478822	72	148	3.447622	72	148	3.478822
	30000	75	154	9.001258	75	154	8.970057	75	154	8.876457
(35)	3000	85	175	26.036567	86	177	26.364169	86	177	26.379769
	6000	45	110	57.79837	84	180	103.631464	84	180	103.693865
	12000	58	146	296.932303	35	98	186.514796	35	98	186.421195
	30000	75	201	2473.645457	96	250	3169.191515	96	250	3169.331916
(36)	3000	33	76	1.279208	48	106	1.794012	49	105	1.809612
	6000	47	106	7.020045	50	112	7.254046	50	112	7.254047
	12000	49	101	26.410969	50	103	26.785372	50	103	26.769772
	30000	40	95	154.050987	41	97	155.096194	41	97	154.940193
(37)	3000	23	48	0.124801	23	48	0.124801	23	48	0.124801
	6000	23	48	0.265202	23	48	0.265202	23	48	0.265202
	12000	23	48	0.499203	23	48	0.499203	23	48	0.483603
	30000	23	48	1.232408	23	48	1.216808	23	48	1.170007
(38)	3000	12	36	0.0468	19	59	0.0468	19	59	0.0624
	6000	47	150	0.156001	26	76	0.093601	26	76	0.0624
	12000	68	178	0.374402	13	41	0.0624	13	41	0.093601
	30000	11	30	0.140401	42	99	0.468003	42	99	0.452403
(39)	3000	21	44	0.483603	21	44	0.514803	21	44	0.483603
	6000	22	46	1.950013	22	46	1.934412	22	46	1.887612
	12000	23	48	7.441248	23	48	7.441248	23	48	7.425648
	30000	24	50	51.979533	24	50	51.121528	24	50	51.027927

Table 2: Continued.

Number	Dim.	Algorithm 1			Algorithm 2			Algorithm 3		
		NI	NFG	CPU	NI	NFG	CPU	NI	NFG	CPU
(40)	3000	449	906	0.468003	800	1619	0.826805	800	1615	0.780005
	6000	449	906	0.748805	800	1619	1.294808	800	1615	1.279208
	12000	449	906	1.263608	800	1619	2.371215	800	1615	2.308815
	30000	449	906	3.07322	800	1619	5.288434	800	1615	5.163633
(41)	3000	93	192	1.170007	94	194	1.201208	164	339	2.028013
	6000	73	152	1.825212	73	152	1.794012	204	419	5.038832
	12000	80	166	3.931225	80	166	3.915625	251	513	12.246079
	30000	90	186	11.029271	90	186	10.93567	323	657	39.07825
(42)	3000	154	318	1.996813	150	310	1.887612	118	242	1.51321
	6000	151	308	3.775224	151	323	3.822024	134	274	3.307221
	12000	188	386	9.40686	202	414	9.937264	156	318	7.675249
	30000	215	453	26.972573	239	492	29.390588	198	402	24.164555
(43)	3000	96	205	1.248008	71	158	0.889206	65	150	0.826805
	6000	65	151	1.669211	58	129	1.48201	81	172	1.996813
	12000	81	177	4.102826	85	185	4.212027	100	210	4.929632
	30000	89	193	11.169672	90	195	11.122871	91	205	11.310072
(44)	3000	142	294	1.856412	137	284	1.731611	128	267	1.63801
	6000	159	328	4.024826	153	316	3.837625	162	339	4.056026
	12000	178	366	9.001258	170	350	8.455254	254	531	12.682881
	30000	212	434	26.379769	199	408	24.554557	800	1612	99.060635
(45)	3000	93	192	1.170007	94	194	1.185608	164	339	2.074813
	6000	73	152	1.840812	73	152	1.825212	206	423	5.101233
	12000	80	166	3.962425	80	166	3.900025	255	521	12.44888
	30000	90	186	11.013671	89	184	10.85767	333	677	40.404259
(46)	3000	72	163	0.951606	73	176	0.982806	99	210	1.279208
	6000	119	250	2.979619	115	242	2.839218	123	258	3.026419
	12000	138	288	6.848444	147	306	7.238446	137	286	6.739243
	30000	174	360	21.574938	172	356	21.091335	170	352	20.794933
(47)	3000	302	613	3.946825	283	575	3.634823	800	1606	10.654868
	6000	398	805	10.124465	368	745	9.31326	800	1606	20.950934
	12000	359	727	18.189717	402	813	20.139729	800	1606	41.621067
	30000	437	883	54.912352	437	883	54.366348	800	1606	103.584664
(48)	3000	26	58	0.343202	26	58	0.327602	27	60	0.327602
	6000	27	60	0.686404	27	60	0.670804	27	60	0.639604
	12000	27	60	1.341609	27	60	1.326009	27	60	1.326009
	30000	27	60	3.354022	27	60	3.291621	28	62	3.416422
(49)	3000	41	86	3.276021	41	86	3.244821	42	88	3.322821
	6000	40	84	10.670468	40	84	10.654868	41	86	10.90447
	12000	40	84	37.48704	40	84	37.799042	42	88	39.655454
	30000	42	88	250.459606	42	88	249.367599	44	92	261.020873
(50)	3000	35	72	0.0624	35	72	0.0624	35	72	0.0468
	6000	36	74	0.093601	36	74	0.109201	36	74	0.078
	12000	38	78	0.171601	38	78	0.156001	38	78	0.156001
	30000	39	80	0.452403	39	80	0.390002	39	80	0.421203
(51)	3000	28	97	1.918812	51	183	3.432022	51	183	3.447622
	6000	47	164	10.608068	56	218	13.104084	56	218	13.088484
	12000	42	135	30.872598	37	160	31.964605	37	160	31.886604
(52)	3000	569	1582	193.176038	800	2260	272.783349	800	2260	272.627348
	6000	800	2184	1074.254086	800	2200	1077.140105	800	2200	1074.83129
	30000	728	2025	24653.74284	784	2183	26511.62115	784	2183	26511.71475

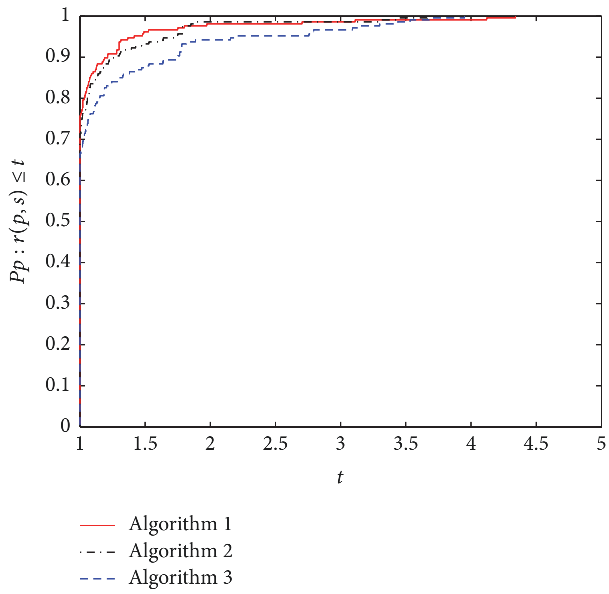

Figure 2: NFG performance of these methods.

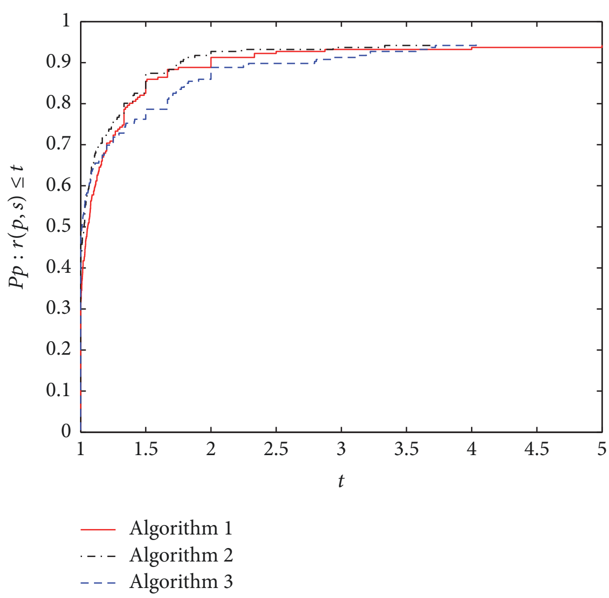

Figure 3: CPU time performance of these methods.

5. Conclusions

This paper proposes a modified HS three-term conjugate gradient algorithm for large-scale optimization problems and the given algorithm has some good features.

(1) The modified HS three-term conjugate gradient possesses a trust region property, which makes the global convergence of the general functions easy to get. However, the normal HS formula including many other conjugate gradient formulas does not have this feature, which may be the crucial point for the global convergence of the general functions.

(2) The largest dimension of the test problems is 30000 variables and the numerical results show that the presented algorithm is competitive to other similar methods. More experiments will be done to prove the performance of the proposed algorithm in the future.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Mr. Xiangrong Li and Dr. Songhua Wang wrote this paper in English. Dr. Zhongzhou Jin and Dr. Hongtruong Pham carried out the experiment. All the authors read and approved the final manuscript.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grants nos. 11661009 and 11261006), the Guangxi Science Fund for Distinguished Young Scholars (no. 2015GXNSFGA139001), the Guangxi Natural Science Key Fund (no. 2017GXNSFDA198046), and the Basic Ability Promotion Project of Guangxi Young and Middle-Aged Teachers (no. 2017KY0019).

References

[1] G. Yuan and Z. Wei, "Convergence analysis of a modified BFGS method on convex minimizations," *Computational optimization and applications*, vol. 47, no. 2, pp. 237–255, 2010.

[2] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, pp. 149–154, 1964.

[3] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436 (1953), 1952.

[4] E. Polak and G. Ribière, "Note sur la convergence de méthodes de directions conjuguées," *Revue Française d'Informatique et de Recherche Opérationnelle*, vol. 3, no. 16, pp. 35–43, 1969.

[5] B. T. Polyak, "The conjugate gradient method in extremal problems," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 4, pp. 94–112, 1969.

[6] G. Yuan, Z. Meng, and Y. Li, "A modified Hestenes and Stiefel conjugate gradient algorithm for large-scale nonsmooth minimizations and nonlinear equations," *Journal of Optimization Theory and Applications*, vol. 168, no. 1, pp. 129–152, 2016.

[7] G. Yuan and M. Zhang, "A three-terms Polak-Ribière-Polyak conjugate gradient algorithm for large-scale nonlinear equations," *Journal of Computational and Applied Mathematics*, vol. 286, pp. 186–195, 2015.

[8] L. Zhang, W. Zhou, and D.-H. Li, "A descent modified Polak-Ribière-Polyak conjugate method and its global convergence," *IMA Journal of Numerical Analysis (IMAJNA)*, vol. 26, no. 4, pp. 629–640, 2006.

[9] G. Yuan, Z. Wei, and X. Lu, "Global convergence of BFGS and PRP methods under a modified weak Wolfe-Powell line search," *Applied Mathematical Modelling: Simulation and Computation for Engineering and Environmental Systems*, vol. 47, pp. 811–825, 2017.

[10] A. S. Nemirovsky and D. B. Yudin, *Problem complexity and method efficiency in optimization*, John Wiley & Sons, Inc., NY, New York, USA, 1983.

[11] G. Yuan, Z. Sheng, B. Wang, W. Hu, and C. Li, "The global convergence of a modified BFGS method for nonconvex functions," *Journal of Computational and Applied Mathematics*, vol. 327, pp. 274–294, 2018.

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

[13] Y. Wu, "A modified three-term PRP conjugate gradient algorithm for optimization models," *Journal of Inequalities and Applications*, Paper No. 97, 14 pages, 2017.

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