Barzilai and Borwein conjugate gradient method equipped with a non-monotone line search technique

Sajad Fathi Hafshejani SAJAD.FATHIHAFSHEJAN@ULETH.CA
Daya Gaur DAYA.GAUR@ULETH.CA
Shahadat Hossain SHAHADAT.HOSSAIN@ULETH.CA
Robert Benkoczi ROBERT.BENKOCZI@ULETH.CA
Department of Math and Computer Science University of Lethbridge, Lethbridge, AB, Canada.

Abstract

In this paper, we propose a new non-monotone conjugate gradient method for solving unconstrained nonlinear optimization problems. We first modify the non-monotone line search method by introducing a new trigonometric function to calculate the non-monotone parameter, which plays an essential role in the algorithm's efficiency. Then, we apply a convex combination of the Barzilai-Borwein method [6] for calculating the value of step size in each iteration. Under some suitable assumptions, we prove that the new algorithm has the global convergence property. The efficiency and effectiveness of the proposed method are determined in practice by applying the algorithm to some standard test problems and non-negative matrix factorization problems.

1. Introduction

In this paper, we are interested to solve the following unconstrained optimization problem:

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

in which $f:\mathbb{R}^n\rightarrow\mathbb{R}$ is a continuously differentiable function. There are various iterative approaches for solving (1) [20]. The Conjugate Gradient (CG) method is one such approach. The CG based methods do not need any second-order information of the objective function. For a given point $x_0\in\mathbb{R}^n$, the iterative formula describing the CG method is:

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

in which x_k is current iterate point, α_k is the step size, and d_k is the search direction determined by:

$$d_k = \begin{cases} -g_k & k=0, \\ -g_k + \beta_{k-1} d_{k-1} & k\geq 1, \end{cases} \tag{3}$$

where $g_k = \nabla f(x_k)$ is the gradient of the objective function in the current iteration. The conjugate gradient parameter is β_k, whose choice of different values leads to various CG methods. The most well-known of the CG methods are the Hestenes-Stiefel (HS) method [15], Fletcher-Reeves (FR) method [10], Conjugate Descent (CD) [11], and Polak-Ribiere-Polyak (PRP) [21].

There are various approaches to determining a suitable step size in each iteration such as Armijo line search, Goldstein line search, and Wolfe line search [20]. The Armijo line search finds the largest value for the step size in each iteration such that the following inequality holds:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \gamma \alpha_k g_k^T d_k \tag{4}$$

in which $\gamma \in (0, 1)$ is a constant parameter. Grippo et al. [12] introduced a non-monotone Armijo-type line search technique as another way to compute the step size. The incorporation of the non-monotone strategy into the gradient and projected gradient approaches, the conjugate gradient method, and the trust-region methods have led to significant improvements to these methods. Zhang and Hager [22] gave some conditions to improve the convergence rate of this strategy. Ahookhosh et al. [3] built on these results and investigated a new non-monotone condition:

$$f(x_k + \alpha_k d_k) \leq R_k + \gamma \alpha_k g_k^T d_k, \quad \text{where} \tag{5}$$

$$R_k = \eta_k f_{l_k} + (1 - \eta_k) f_k, \tag{6}$$

$$\eta_k \in [\eta_{\min}, \eta_{\max}],\ \eta_{\min} \in [0, 1), \eta_{\max} \in [\eta_{\min}, 1], \quad f_{l_k} = \max_{0 \leq j \leq m_k} \{f_{k-j}\},$$
$$m_0 = 0,\ \ 0 \leq m_k \leq \min\{m_{k-1} + 1, N\} \text{ for some } N \geq 0. \tag{7}$$

Note that η_k is known as the non-monotone parameter and plays an essential role in the algorithm's convergence. Although this new non-monotone strategy in [3] has some appealing properties, especially in functional performance, current algorithms based on this non-monotone strategy face the following challenges.

- The existing schemes for determining the parameter η_k may not reduce the value of the objective function significantly in initial iterations. To overcome this drawback, we propose a new scheme for choosing η_k based on the gradient behaviour of the objective function. This can reduce the total number of iterations.

- Many evaluations of the objective function are needed to find the step length α_k in step k. To make this step more efficient, we use an adaptive and composite step length procedure from [18] to determine the initial value of the step length in inner iterations.

- The third issue is the global convergence for the non-monotone CG method. Most exiting CG methods use the Wolfe condition, which plays a vital role in establishing the global convergence of various CG methods [19]. Wolfe line search is more expensive than the Armijo line search strategy. We define a suitable conjugate gradient parameter so that the scheme proposed here satisfies the global convergence property.

By combining the outlined strategies, we propose a modification to the non-monotone line search method. Then, we incorporate this approach into the CG method and introduce a new non-monotone CG algorithm. We prove that our proposed algorithm has global convergence. Finally, we compare our algorithm and eight other algorithms on standard tests and non-negative matrix factorization instances. We report several criteria, such as the number of objective function evaluations, the number of gradient evaluations, the number of iterations, and the CPU time, to compare the performance of the algorithms.

2. An improved non-monotone line search algorithm

In this section, we discuss our strategies for choosing the parameters that define our algorithm: the non-monotone parameter η, the step size α, and the conjugate gradient parameter β.

2.1. A new scheme of choosing η_k

Recall that the non-monotone line search strategy is determined by equation (5) in step k. Parameter η_k is used in the non-monotone term (6) and determines how aggressively we descend towards solutions with smaller cost. There are two common approaches for calculating η_k. The scheme proposed by Ahookhosh et al. [3] has been used in most of the existing non-monotone algorithms [1, 2, 9]. This strategy is described by recurrence $\eta_k = \frac{1}{3}\eta_0(-\frac{1}{2})^k + \frac{2}{3}\eta_0$ where $\eta_0 = 0.15$ and $\lim_{k\to\infty} \eta_k = 0.1$. The other scheme proposed by Amini et al. [4], which depends on the behaviour of gradient is given by:

$$\eta_0 = 0.95, \quad \eta_k = \begin{cases} \frac{2}{3}\eta_{k-1} + 0.01, & \text{if } \|g_k\|_\infty \le 10^{-3}; \\ \max\{0.99\eta_{k-1}, 0.5\}, & \text{otherwise.} \end{cases} \tag{8}$$

To illustrate the behaviour of these two schemes, we solve the problem $f(x) = (x_0 - 5)^2 + \sum_{i=1}^{40}(x_i - 1)^2$ for $x \in \mathbb{R}^{41}$. The values of parameter η_k corresponding to the two schemes are displayed in Fig. 1 (Left). As shown in Fig. 1, for the scheme proposed by Ahookhosh et al. [3],

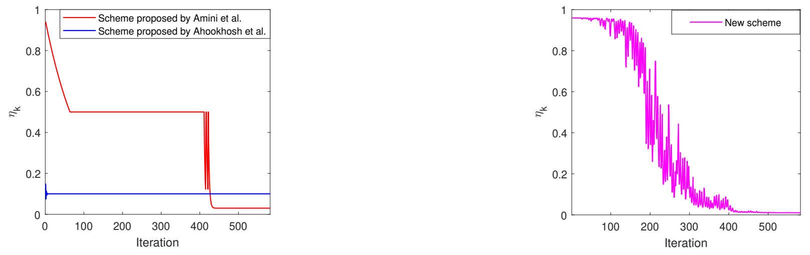

Figure 1: (Left): Values of η_k proposed in [3] and [4], (Right): Values of η_k for the new scheme.

η_k is close to 0.1 after only a few iterations. Notice that η_k in each iteration does not have any connection with the behaviour of the objective function. Thus this scheme is not effective. In addition, there are two issues with the scheme introduced by Amini et al. in [4]. One problem indicated by Fig. 1 is that that η_k decreases relatively quickly for the first 65 iterations. Since the algorithm requires many iterations to solve the test problem, ideally η_k should be close to 1 for these initial iterations. The second issue is that the value of η_k remains constant for a large number of iterations and it is not affected by the behaviour of the objective function.

To avoid theses challenges, we propose an adaptive strategy for calculating the value of η_k:

$$\eta_k = 0.95 \sin\left(\frac{\pi\|g_k\|}{1 + 2\|g_k\|}\right) + 0.01. \tag{9}$$

When x_k is far away from the minimizer, we can reasonably assume that $\|g_k\|$ is large. Thus the value of η_k defined by (9) is close to 1. This allows the algorithm to explore a broader set of solutions in the initial iterations, providing a chance to reduce the value of the objective function more significantly. On the other hand, when x_k is close to the optimal solution, $\|g_k\|$ is small, then the value of η_k is close to zero. Thus, the step length is small so that the new point stays in the neighbourhood of the optimal point. We believe that the new scheme better serves the objectives of the non-monotone strategy. We plot the evolution of η_k defined by Eq. (9) in Fig. 1 (right), using the gradient from the optimization problem mentioned above. As mentioned earlier, we need strong non-monotonicity for the first few iterations. This does not mean that the value of η_k must be exactly one but that it is close to one. To achieve this, we multiply function $\sin()$ by 0.95. In addition, for the final iterations where the value of gradient $\|g_k\|$ approaches zero, we need weak non-monotonicity, i.e. the value of parameter η_k should be close to zero but not exactly zero. In this case, we add a constant term such as 0.01 to the expression.

2.2. New schemes for choosing α_k

We utilize a convex combination of the Barzilai-Borwein (BB) step sizes to calculate an appropriate α_k in each outer iteration [18]. We calculate α_k, using the following equation:

$$\alpha_k^{\text{CBB}} = \mu_k \alpha_k^{(1)} + (1-\mu_k)\alpha_k^{(2)}, \quad \text{where} \tag{10}$$

$$\alpha_k^{(1)} = \frac{s_k^T s_k}{s_k^T y_k}, \quad \alpha_k^{(2)} = \frac{s_k^T y_k}{y_k^T y_k}, \quad s_k := x_k - x_{k-1}, \quad y_k := g_k - g_{k-1};$$

$$\mu_k = \frac{K_2}{K_1+K_2} \quad K_1 = \|\alpha_k^{(1)} y_k - s_k\|^2, \quad K_2 = \|(\alpha_k^{(2)})^{-1} s_k - y_k\|^2.$$

2.3. Conjugate gradient parameter

We propose the following expression for the new conjugate gradient parameter:

$$\beta_k = \omega \frac{\|g_k\|}{\|d_{k-1}\|}, \quad \omega \in (0,1). \tag{11}$$

The complete algorithm is listed in Appendix 5 (Algorithm 1). The next lemma proves a key property for β_k which is very important in proving the algorithm's convergence. The proofs appear in Appendix 5.

Lemma 1 *Let d_k be the search direction used in iteration k of Algorithm 1 and let c be a suitable constant $c > 0$. Then,*

$$d_k^T g_k \leq -c\|g_k\|^2.$$

2.4. Algorithm

In this section, we describe the new non-monotone conjugate gradient algorithm. Algorithm 1 consists of two loops, an inner and an outer loop. The inner loop computes the value of the step size only, using the non-monotone line search strategy from Eq. (5). The outer loop computes a new point $x_{k+1} = x_k + \alpha_k d_k$, the conjugate gradient parameter β_k using Eq. (11), and the search direction d_{k+1} using Eq. (3). Finally, the algorithm calculates the non-monotone line search parameter using Eq. (9). This process is repeated until a proper solution is obtained.

The following assumptions are used to analyze the convergence properties of Algorithm 1.

H1 The level set $\mathcal{L}(x_0) = \{x | f(x) \leq f(x_0), \quad x \in \mathbb{R}^n\}$ is bounded set.

H2 The gradient of objective function is Lipschitz continuous over an open convex set C containing $\mathcal{L}(x_0)$. That is:

$$\|g(x) - g(y)\| \leq L\|x-y\|, \qquad \forall\, x, y \in C.$$

We prove the following Theorem about the global convergence of Algorithm 1, the proof of which follows from the lemmas presented in this section. Please see the appendix for the proofs.

Algorithm 1: An improved non-monotone conjugate gradient method

```
Data: $x_0 \in \mathbb{R}^n,\ \epsilon > 0,\ \rho \in (0,1), \gamma \in (0,1), N > 0, 0 < \beta_1 < 1, \omega \in (0,1)$, and $\alpha_0 = 1$
      Set $k = 0$
while $\|g_k\| \geq \varepsilon$ do
    Set $\alpha = \alpha_0$
    Set: $x_{k+1} = x_k + \alpha d_k$
    while the non-monotone line search condition (5) is not satisfied do
        $\alpha \leftarrow \rho\alpha$
    end
    $\alpha_k \leftarrow \alpha$
    Generate the new point $x_{k+1} = x_k + \alpha_k d_k$
    Compute the parameter $\beta_k$ using (11)
    Generate $d_{k+1}$ using (3)
    Calculate $\eta_{k+1}$ using (9)
    Calculate $\alpha_0$ using (10)
    $k \leftarrow k + 1$
end
```

Theorem 2 *Let $(H1)$, $(H2)$, and Lemmas 1 and 3 hold. Then, for the sequence $\{x_k\}$ generated by Algorithm 1, we have* $\lim_{k\to\infty} \|g_k\| = 0$.

Lemma 3 *Suppose that the search direction d_k with the CG parameter β_k given by (11) is generated by Algorithm 1. Then, an upper bound for d_k is given by $\|d_k\| \leq (1+\omega)\|g_k\|$.*

Lemma 4 *Suppose that x_k is not a stationary point of (1). Then we have*

$$\alpha_k \geq \min\left\{\beta_1\rho, \frac{2(1-\omega)\rho(1-\gamma)}{L(1+\omega)^2}\right\}.$$

Recall that the objective function f is a strongly convex function if there exists a scalar μ such that

$$f(x) \geq f(y) + \nabla f(y)^T(x-y) + \frac{1}{\mu}\|x-y\|^2 \tag{12}$$

Theorem 5 *If $(H1)$, $(H2)$, Eq. (12), and lemmas 1 and 3 hold then the sequence $\{x_k\}$ generated by Algorithm 1 converges to the stationary point x_* at least R-linearly.*

3. Numerical Results

In this section we test the new algorithm to solve a set of standard optimization problems and the non-negative matrix factorization problem, which is a non-convex optimization problem. The implementation level details are in Appendix 6. To demonstrate the efficiency of the proposed algorithm, we compare our algorithm and eight other existing algorithms introduced in [3, 4, 13, 16] on a set of 110 standards test problems. To describe the behaviour of each strategy, we use performance profiles proposed by Dolan and Moré [8]. Note that the performance profile for an algorithm $p_s(\tau) : \mathbb{R} \mapsto [0,1]$ is a non-decreasing, piece-wise constant function, continuous from the right at each breakpoint. Moreover, the value $p_s(1)$ denotes the probability that the algorithm will win against the rest of the algorithms. More information on the performance profile appears in Appendix 6. We plot the performance profile of each algorithm in terms of the total number of outer iterations and the CPU time on the set of standard test problems in Fig. 2. In Fig 2, "Line Search"

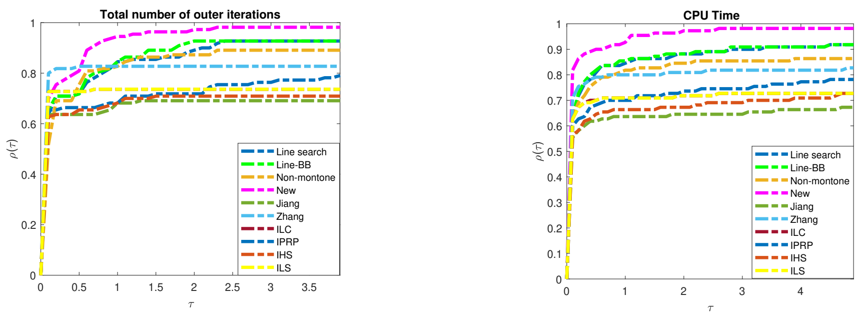

Figure 2: (Left): Performance profiles of the total number of outer iterations, (Right): Performance profiles of CPU Time.

is from [20], "Line-BB" denotes the line search method using BB method for calculating the initial value for the step size, "Non-monotone" only uses the non-monotone line search method, "Zhang" method is described in [13], and "Jiang" method is from [16]. Detailed information about "ILC", "IPRP", "IHS", "ILS" methods can be found in [16].

We also apply our algorithm to solve the Non-Negative Matrix Factorization (NMF) which has several applications in image processing such as face detection problems. Given a non-negative matrix $V \in \mathbb{R}^{m\times n}$, a NMF finds two non-negative matrices $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ with $k \ll \min(m,n)$ such that $X \approx WH$. This problem can be formulated as

$$\min_{W,H\geq 0} F(W,H) = \frac{1}{2}\|V - WH\|_F^2. \tag{13}$$

Equation (13) is a non-convex optimization problem. We compare our method and Zhang's algorithm [13] on some random data sets. We report these results in Appendix 6.

4. Conclusion

In this paper, we introduce a new non-monotone conjugate gradient algorithm based on an efficient Barzilai-Borwein step size. We introduce a new non-monotone parameter based on gradient behaviour and determined by a trigonometric function. We use a convex combination of two Barzilai-Borwein step size parameters to compute the step size value in each iteration. We prove that the proposed algorithm has global convergence. We implemented and tested our algorithm on a set of standard test problems and on several instances for the non-negative matrix factorization problem. The proposed algorithm can solve 98% of the problems from a set of standard test instances. For the non-negative matrix factorization, the results indicate that our algorithm is more efficient compared to Zhang' s method [13].

Acknowledgements: The authors would like to thank the reviewers for their insightful comments about comparison with the basic non-monotone method and convergence which lead to Theorem 5.

References

[1] Masoud Ahookhosh and Keyvan Amini. An efficient nonmonotone trust-region method for unconstrained optimization. *Numerical Algorithms*, 59(4):523–540, 2012.

[2] Masoud Ahookhosh and Susan Ghaderi. On efficiency of nonmonotone armijo-type line searches. *Applied Mathematical Modelling*, 43:170–190, 2017.

[3] Masoud Ahookhosh, Keyvan Amini, and Mohammad Reza Peyghami. A nonmonotone trust-region line search method for large-scale unconstrained optimization. *Applied Mathematical Modelling*, 36(1):478–487, 2012.

[4] Keyvan Amini, Masoud Ahookhosh, and Hadi Nosratipour. An inexact line search approach using modified nonmonotone strategy for unconstrained optimization. *Numerical Algorithms*, 66(1):49–78, 2014.

[5] Neculai Andrei. An unconstrained optimization test functions collection. *Adv. Model. Optim*, 10(1):147–161, 2008.

[6] Jonathan Barzilai and Jonathan M Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

[7] Yu-Hong Dai. On the nonmonotone line search. *Journal of Optimization Theory and Applications*, 112(2):315–330, 2002.

[8] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[9] Hamid Esmaeili and Morteza Kimiaei. A new adaptive trust-region method for system of nonlinear equations. *Applied mathematical modelling*, 38(11-12):3003–3015, 2014.

[10] Reeves Fletcher and Colin M Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[11] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2013.

[12] Luigi Grippo, Francesco Lampariello, and Stephano Lucidi. A nonmonotone line search technique for newton's method. *SIAM journal on Numerical Analysis*, 23(4):707–716, 1986.

[13] William W Hager and Hongchao Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on optimization*, 16(1):170–192, 2005.

[14] Lixing Han, Michael Neumann, and Upendra Prasad. Alternating projected barzilai-borwein methods for nonnegative matrix factorization. *Electron. Trans. Numer. Anal*, 36(6):54–82, 2009.

[15] Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.

[16] Xianzhen Jiang and Jinbao Jian. Improved fletcher–reeves and dai–yuan conjugate gradient methods with the strong wolfe line search. *Journal of Computational and Applied Mathematics*, 348:525–534, 2019.

[17] DD Lee and HS Seung. Algorithms for non-negative matrix factorization. nips (2000). *Google Scholar*, pages 556–562.

[18] Ting Li and Zhong Wan. New adaptive barzilai–borwein step size and its application in solving large-scale optimization problems. *The ANZIAM Journal*, 61(1):76–98, 2019.

[19] JL Nazareth. Conjugate-gradient methods, encyclopedia of optimization, c. floudas and p. pardalos, eds, 1999.

[20] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[21] Elijah Polak and Gerard Ribiere. Note sur la convergence de méthodes de directions conjuguées. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 3(R1):35–43, 1969.

[22] Hongchao Zhang and William W Hager. A nonmonotone line search technique and its application to unconstrained optimization. *SIAM journal on Optimization*, 14(4):1043–1056, 2004.

5. Appendix A

5.1. Convergence

The proofs of the various lemmas and the main theorem are presented in this section.
Proof Proof of Lemma 1:
If $k = 0$, we have

$$d_k^T g_k = -\|g_k\|^2 < 0.$$

If $k \geq 1$, using (3) and (11), and we have:

$$d_k^T g_k = -\|g_k\|^2 + \omega \frac{\|g_k\|}{\|d_{k-1}\|} d_{k-1}^T g_k.$$

Using the Cauchy-Schwarz inequality and we have:

$$d_k^T g_k \leq -\|g_k\|^2 + \omega \|g_k\|^2 = -(1-\omega)\|g_k\|^2. \tag{14}$$

■

Proof Proof of Lemma 3:
The proof is obtained by combining (3) and the triangle inequality, that is:

$$\|d_k\| \leq \|g_k\| + \beta_k \|d_{k-1}\| = (1+\omega)\|g_k\|.$$

■

To prove the convergence results, we need the following elementary lemmas.

Lemma 6 *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then, f_{l_k} is a decreasing sequence.*

Proof We use the definition R_k and (6), which imply that

$$R_k = \eta_k f_{l(k)} + (1-\eta_k) f_k \le \eta_k f_{l(k)} + (1-\eta_k) f_{l(k)} = f_{l(k)}. \tag{15}$$

It follows that:

$$f_{k+1} \le R_k + \gamma \alpha_k g_k^T d_k \le f_{l(k)} + \gamma \alpha_k g_k^T d_k.$$

By using the Lemma 1, we can conclude that:

$$f_{k+1} \le f_{l(k)}. \tag{16}$$

On the other hand, from (6) we have:

$$\begin{aligned} f_{l(k+1)} &= \max_{0 \le j \le m(k+1)} \{f_{k+1-j}\} \\ &\le \max_{0 \le j \le m(k)+1} \{f_{k+1-j}\} = \max\left\{f_{l(k)}, f_{k+1}\right\}. \end{aligned}$$

By applying (16), that is $f_{k+1} \le f_{l(k)}$, we conclude that $f_{l(k+1)} \le f_{l(k)}$. This shows that the sequence $f_{l(k)}$ is a decreasing sequence. ■

Lemma 7 *Suppose that the sequence $\{x_k\}$ is generated by Algorithm 1. Then, for all $k \ge 0$, we have $x_k \in \mathcal{L}(x_0)$.*

Proof Definition of $f_{l(k+1)}$ implies that $f_{k+1} \le f_{l(k+1)}$ for any $k \ge 0$. Therefore, we have:

$$\begin{aligned} f_{k+1} &= \eta_{k+1} f_{k+1} + (1-\eta_{k+1}) f_{k+1} \\ &\le \eta_{k+1} f_{l(k+1)} + (1-\eta_{k+1}) f_{k+1} = R_{k+1}, \qquad \forall\, k \in N_k \end{aligned} \tag{17}$$

By using definition of R_k, we can conclude that $R_0 = f_0$. Now, by induction, assuming $x_i \in \mathcal{L}(x_0)$, for all $i = 1, 2, \ldots, k$, we show that $x_{k+1} \in \mathcal{L}(x_0)$. Relations (6) and (15) together with Lemma 6 imply that:

$$f_{k+1} \le f_{l(k+1)} \le f_{l(k)} \le f_0,$$

which implies that the sequence x_k is contained in $\mathcal{L}(x_0)$. ■

The next part of this section describes some convergence results for the Algorithm 1.

Lemma 8 *Suppose that Algorithm 1 generates the sequence $\{x_k\}$ and $(H1)$–$(H2)$ hold. Then, the sequence $\{f_{l(k)}\}$ is convergent.*

Proof By using Lemma 6 and the fact that $f_{l(0)} = f_0$ imply that the sequence $\{x_{l(k)}\}$ remains in level set $\mathcal{L}(x_0)$. On the other hand, this fact $f(x_k) \le f(x_{l(k)})$ proves that the sequence $\{x_k\}$ remains in $\mathcal{L}(x_0)$. Therefore, $(H1)$ together with Lemma 6 imply that the sequence $\{f_{l(k)}\}$ is convergent. ■

Lemma 9 *Suppose that (H1) holds and the direction* d_k *satisfies the first item of Lemma 6. Then for the sequence* $\{x_k\}$ *generated by Algorithm 1, we have:*

$$\lim_{k\to\infty} f_{l(k)} = \lim_{k\to\infty} f_k.$$

Proof The proof is similar to Lemma 2 in [4]. Therefore, we omit it here. ■

Lemma 10 *Suppose that (H1) holds and the direction* d_k *satisfies the first item of Lemma 6. If the sequence* $\{x_k\}$ *generated by Algorithm 1, then we have:*

$$\lim_{k\to\infty} R_k = \lim_{k\to\infty} f_k.$$

Proof Using (15) and (17), we conclude that:

$$f_k \leq R_k \leq f_{l(k)}$$

By applying Lemma 9, we obtain the result. ■

Now, we can prove the Lemma 4.
Proof Proof of Lemma 4:
We consider two cases: If $\frac{\alpha_k}{\rho} \geq \beta_1$, which implies that $\alpha_k \geq \beta_1\rho$ and it completes the proof. Now, we assume that $\frac{\alpha_k}{\rho} < \beta_1$. In this case we have $\alpha_k < \beta_1\rho$. Therefore, the non-monotone condition does not hold, i.e.,

$$f(x_k + \frac{\alpha_k}{\rho} d_k) > R_k + \gamma \frac{\alpha_k}{\rho} g_k^T d_k \tag{18}$$

Now, by using the mean value theorem, i.e., Cauchy–Schwarz inequality, we conclude that:

$$\begin{aligned} f(x_k + \alpha d_k) &= f(x_k) + \alpha g_k^T d_k + \int_0^1 \alpha(g(x_k + t\alpha d_k) - g_k)^T d_k dt \\ &\leq f(x_k) + \alpha g_k^T d_k + \alpha\|d_k\| \int_0^1 \|(g(x_k + t\alpha d_k) - g_k^T)\| dt \end{aligned} \tag{19}$$

Now, by using $(H2)$ and the fact that $R_k \geq f_k$, we have:

$$\begin{aligned} f(x_k + \alpha d_k) &\leq f(x_k) + \alpha g_k^T d_k + L\alpha^2\|d_k\|^2 \int_0^1 t dt \\ &= f(x_k) + \alpha g_k^T d_k + \frac{L}{2}\alpha^2\|d_k\|^2 \\ &\leq R_k + \alpha g_k^T d_k + \frac{L}{2}\alpha^2\|d_k\|^2 \end{aligned} \tag{20}$$

By putting $\alpha = \frac{\alpha_k}{\rho}$ and combining (18) with (20), we conclude that:

$$R_k + \frac{\alpha_k}{\rho} g_k^T d_k + \frac{L}{2\rho^2}\alpha_2^2\|d_k\|^2 \geq R_k + \gamma \frac{\alpha_k}{\rho} g_k^T d_k.$$

Using Lemmas 1 and 3, we conclude that:

$$\frac{L}{2\rho}(1+\omega)^2\alpha_2^2\|g_k\|^2 \geq \frac{L}{2\rho}\alpha_2^2\|d_k\|^2 > -(1-\gamma)g_k^T d_k \geq (1-\omega)(1-\gamma)\|g_k\|^2. \tag{21}$$

It implies that:

$$\alpha_k \geq \frac{2(1-\omega)\rho(1-\gamma)}{L(1+\omega)^2}.$$

■

Now, we prove the Theorem 2, which shows that Algorithm 1 has global convergence.

Proof Proof of Theorem 2:
We have:

$$\begin{aligned} R_k + \gamma\alpha_k g_k^T d_k &\geq f_{k+1} \\ \Rightarrow R_k - f_{k+1} &\geq -\gamma\alpha_k g_k^T d_k \geq c_1\gamma\alpha_k\|g_k\|^2 \geq 0. \end{aligned} \tag{22}$$

This fact together with Lemma 10 imply that $\lim_{k\to\infty}\|g_k\| = 0$. This shows that proposed algorithm has global convergence. ■

Lemma 11 *Suppose $(H1)$ and $(H2)$ hold, the direction d_k satisfies (1) and (3), and the sequence $\{x_k\}$ is generated by Algorithm 1. Then, for any $l \geq 1$ we have:*

$$\max_{1\leq i\leq N} f(x_{Nl+i}) \leq \max_{1\leq i\leq N} f(x_{N(l-1)+i}) + \gamma \max_{0\leq i\leq N-1}\left[\alpha_{Nl+i}g_{Nl+i}^T d_{Nl+i}\right].$$

Proof From Lemma 6, we have:

$$\begin{aligned} f(x_{Nl+1}) &\leq R_{Nl} + \gamma\alpha_{Nl}g_{Nl}^T d_{Nl} \\ &\leq \max_{1\leq i\leq m(Nl)} f(x_{Nl-i}) + \alpha_{Nl}g_{Nl}^T d_{Nl}. \end{aligned}$$

The rest of the proof is similar to Lemma 2.1 in [7]. ■

Lemma 12 (Therorem 2.1 in [7]) *Suppose $(H1)$ and $(H2)$ hold, the direction d_k satisfies (1) and (3), and the sequence $\{x_k\}$ is generated by Algorithm 1. Then there exists a constant c_2 such that:*

$$\|g_{k+1}\| \leq c_2\|g_k\|$$

Lemma 13 *[Theorem 3.1 in [7]] Suppose $(H1)$ and $(H2)$ hold, $f(x)$ be a strongly convex function and the direction d_k satisfies (1) and (3) and the sequence $\{x_k\}$ is generated by Algorithm 1. Then there exists constants $c_3 > 0$ and $c_4 \in (0,1)$, such that:*

$$f(x_k) - f(x_*) \leq c_3 c_4^k\left[f(x_1) - f(x_*)\right]$$

Proof Proof of Theorem 5:
Using (12) and setting $y = x_*$ and $x = x_k$. Therefore, we have:

$$f(x_k) \geq f(x_*) + \frac{1}{2w}\|x_k - x_*\| \Rightarrow \|x_k - x_*\| \leq 2w(f(x_k) - f(x_*))$$

Using Lemma 13, we have:

$$\|x_k - x_*\| \leq c_4^k \left[2wc_3(f(x_1) - f(x_*))\right] = c_4^k r_k,$$

where $r = 2wc_3(f(x_1) - f(x_*))$. By setting $v_k = rc_4^k$, we get that $v_* = 0$. It follows that:

$$\lim_{k\to\infty} \frac{v_{k+1} - v_*}{v_k - v_*} = c_4 < 1.$$

This implies that the sequence $\{x_k\}$ converges to x_* at least R-linearly [20]. ■

6. Appendix B

6.1. Numerical Results

Here, we present some implementation level details. Since our algorithm improves the non-monotone scheme, we chose two algorithms from the non-monotone line search category and six algorithms from the Wolfe line search area for performing the comparison. We select the following two state of art algorithms in the non-monotone category. These algorithms calculate the value of step size using a the non-monotone line search strategy in each iteration.

- Ahookhosh's strategy in [3]
- Amini's strategy in [4]

Following six other algorithms that use the Wolfe line search conditions to compute step size in each iteration are used for comparison.

- β^{Jiang} proposed in [16]
- β^{Zhang} proposed in [13]
- β^{ILC} proposed in [16]
- β^{IPRP} proposed in [16]
- β^{IHS} proposed in [16]
- β^{ILS} proposed in [16]

All algorithms were coded in MATLAB 2017 environment and tested on a laptop (Intel(R) Core(TM) i5-7200U CPU 3.18 GHz with 12GB RAM). For all algorithms, we use the following initial values.

$$\gamma = 10^{-4}, \qquad N = 5, \qquad \rho = 0.75.$$

All the experiments terminate when the following conditions are met:

- $\|g_k\| < 10^{-6}$
- The number of iterations is greater than 20000.

As parameter ω can take positive real values in the interval $(0,1)$, there are many choices for it. We tried several strategies and chosen the following strategy, which perform better. The key idea of this choice is from the structure of the conjugate gradient parameter proposed by Jiang [16].

$$\omega_k = \begin{cases} 0.001 & if\ \frac{|g_k^T d_{k-1}|}{-g_{k-1}^T d_{k-1}} \leq 0, \\ 0.999 & if\ \frac{|g_k^T d_{k-1}|}{-g_{k-1}^T d_{k-1}} \geq 1, \\ \frac{|g_k^T d_{k-1}|}{-g_{k-1}^T d_{k-1}} & otherwise. \end{cases}$$

The following subsection presents the results on a set of 110 standard test problems.

6.2. A set of standard test problems

We run all the algorithms on 110 standard test problems from [5] with dimensions ranging between 2 to $5,000,000$. When the algorithm stops under the second condition, i.e., the number of iterations is greater than 20000, the method is deemed to fail for solving the corresponding test problem. The comparison between considered algorithms is based on the number of function evaluations, the number of gradient evaluations, the number of iterations, and the CPU time(s).
To visualize the complete behaviour of the algorithms, we use the performance profiles proposed by Dolan and Moré [8]. Note that the performance profile $p_s(\tau) : \mathbb{R} \mapsto [0,1]$ for an algorithm is a non-decreasing, piece-wise constant function, continuous from the right at each breakpoint. Moreover, the value of $p_s(1)$ denotes the probability that the algorithm will win over the rest.

Suppose that K is a set of n_k test functions and S is a set of n_s solvers. For $s \in S$ and function $k \in K$, consider $a_{p,s}$ as the number of gradient evaluations, objective function evaluations, CPU Time, or the number of iterations required to solve function $k \in K$ by algorithm $s \in S$. Then the algorithms comparison is based on the performance ratio as follows:

$$r_{k,s} = \frac{a_{k,s}}{\min\{a_{k,s},\ k \in K,\ s \in S\}}$$

We obtain the overall evaluation of each algorithm by:

$$p_s(\tau) = \frac{1}{n_k}\, size\ \{k \in K : r_{k,s} \leq \tau\} \tag{23}$$

In general, solvers with high values of $p_s(\tau)$ or in the upper right of the figure represent the best algorithm.

The performance profile in terms of function evaluations and the number of gradient evaluations are presented in figures 3 and 4, respectively.

Figures 5 and 6 show the performance profile in terms of the number of iterations and the CPU time(s) for the proposed algorithm and eight other algorithms.

In Fig 3–6, “Line Search” is presented in [20], “Line-BB” denotes the line search method using BB method for calculating initial value of step size, ”Non-monotone” only uses the non-monotone

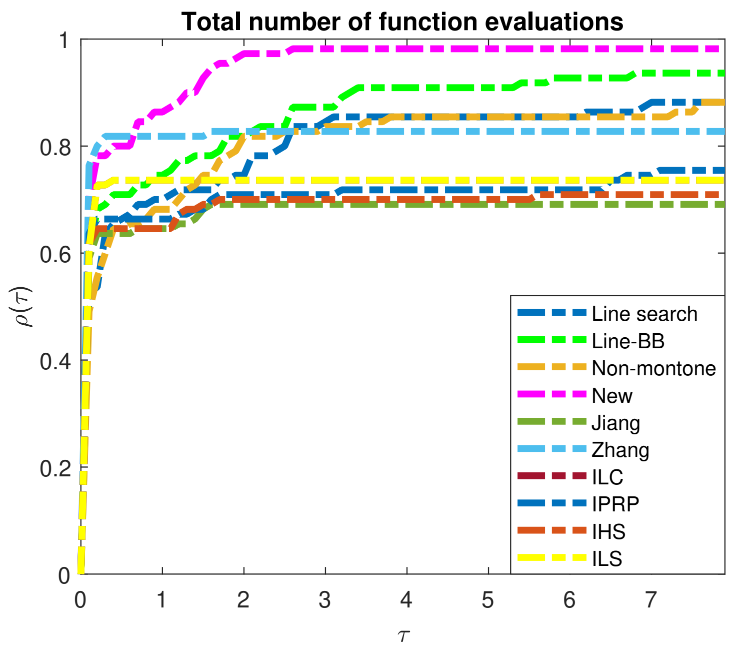

Figure 3: Performance profiles of the total number of function evaluations

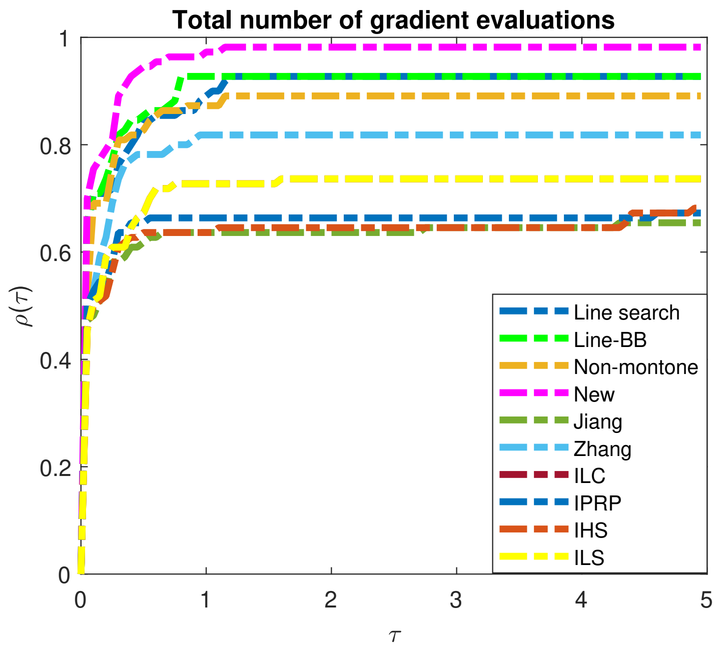

Figure 4: Performance profiles of the total number of gradient evaluations

line search method, “Zhang” method suggested in [13], “Jiang” method introduced in [16], and more information about “ILC”, “IPRP”, “IHS”, “ILS” can be found in [16].

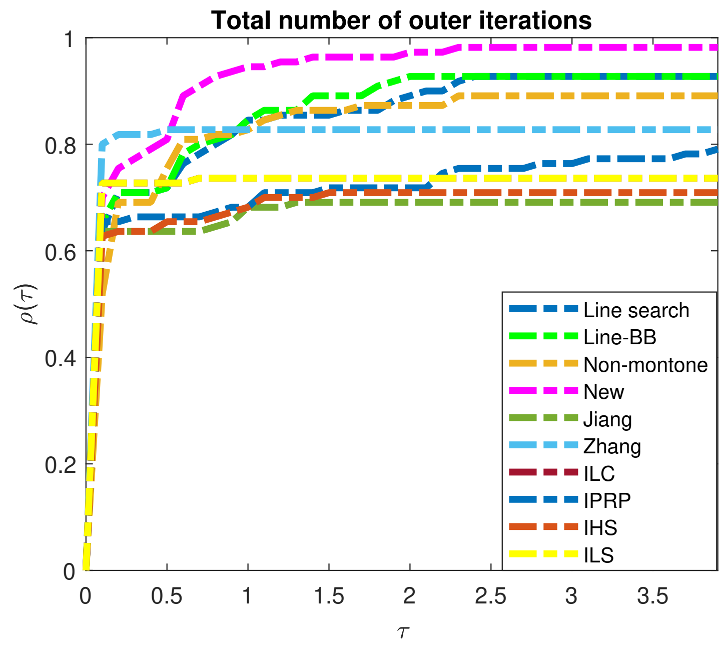

Figure 5: Performance profiles of the total number of outer iterations

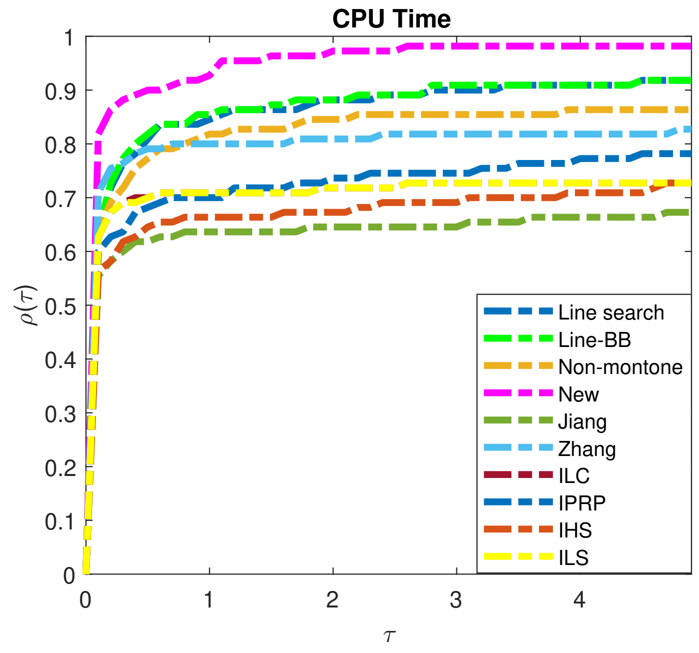

Figure 6: Performance profiles of CPU Time

We conclude from the figures that the proposed algorithm can solve 98% of the test problems. The performance profiles for the number of iterations, total CPU, time, number of gradient evaluations, and number of function evaluations indicate that the proposed method has a high computational performance compared to the other methods.

6.3. Some non-negative matrix factorization test problems

Here, we apply our algorithm to Non-Negative Matrix Factorization (NMF) which has several applications in image processing, such as face detection problems. Given a non-negative matrix $V \in \mathbb{R}^{m\times n}$, a NMF finds two non-negative matrices $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ with $k \ll \min(m, n)$ such that

$$V \approx WH. \tag{24}$$

This problem can be formulated as a non-convex optimization problem:

$$\min_{W,H\geq 0} F(W, H) = \frac{1}{2}\|V - WH\|_F^2. \tag{25}$$

In recent years, several iterative approaches have been introduced for solving (25), for example, [14, 17]. The alternating non-negative least squares (ANLS) framework is a popular approach for solving (25), which finds the optimal solution by solving the following two convex sub-problems:

$$W^{k+1} = \text{arg}\min_{W\geq 0} F(W, H^k) = \frac{1}{2}\|V - WH^k\|_F^2, \tag{26}$$

and

$$H^{k+1} = \text{arg}\min_{H\geq 0} F(W^{k+1}, H) = \frac{1}{2}\|V - W^{k+1}H\|_F^2. \tag{27}$$

To solve this problem, we use the following strategy:

S0 Algorithm starts with the initial point, i.e., $\bar{W} \geq 0$ and $\bar{H} \geq 0$, set $k = 0$.

S1 Stop if $\|[\nabla_H F(\bar{W^k}, \bar{H^k}), \nabla_W F(\bar{W^k}, \bar{H^k})]\|_F \leq \epsilon \|[\nabla_H F(\bar{W^0}, \bar{H^0}), \nabla_W F(\bar{W^0}, \bar{H^0})]\|_F$.

S2 To get W^{k+1}, solve the sub-problem: $\min_{W\geq 0} F(W, \bar{H}^k) = \frac{1}{2}\|V - W\bar{H}^k\|_F^2$.

S4 Set $\bar{W}^{k+1} = W^{k+1}$.

S2 To get H^{k+1}, solve the sub-problem: $\min_{H\geq 0} F(\bar{W}^{k+1}, H) = \frac{1}{2}\|V - \bar{W}^{k+1}H\|_F^2$.

S5 Set $\bar{H}^{k+1} = H^{k+1}$.

S6 Set $k := k + 1$ and go to S1.

Now, we use the above setup to solve some NMF problems using our algorithm and compare it to Zhang's algorithm [13] which had the best results for solving a set of standard test problems.

To this end, we generate a random matrix V as random with elements in $[0, 1]$. We run the algorithm for matrices with ranks $\{5, 10, 15, 20, 40, 50\}$. For each case, we run each of the algorithms 10 times. We, calculated the average of the results and presented them in Table 1. In this table, m and n denote the number of rows and columns of matrix V, k denote the matrix rank. The number of outer iterations is denoted by "Iter". We use the "Niter" for the number of inner iterations. The value of gradient is denoted by "Pgn". The CPU time and error for each of the algorithms are denoted by "Time" and "Error" respectively.

As we see that in most cases, the proposed algorithm performs better than previous best algorithm due to Zhang [13].

$m\quad n\quad k$	$Iter$	$Niter$	Pgn	$Time$	$Error$	$Algorithm$
$50 \times 25 \times 5$	29.20	187.30	0.0045	0.02	0.014	Zhang
	26.70	72.60	0.0043	0.02	0.014	New
$100 \times 50 \times 5$	24.20	165.30	0.058	0.03	0.12	Zhang
	22.30	85.80	0.033	0.01	0.12	New
$100 \times 200 \times 15$	25.30	196.10	0.015	0.137	0.086	Zhang
	17.50	112.80	0.018	0.032	0.085	New
$200 \times 100 \times 10$	19.00	134.80	.093	0.10	0.08	Zhang
	17.10	103.40	.073	0.03	0.08	New
$300 \times 100 \times 20$	25.00	212.40	0.55	0.32	0.08	Zhang
	20.60	131.50	0.44	0.15	0.08	New
$300 \times 500 \times 20$	29.00	128.90	1.98	0.75	0.054	Zhang
	25.80	90.80	0.96	0.14	0.054	New
$500 \times 100 \times 20$	36.10	231.80	7.6	0.48	0.06	Zhang
	31.70	90.70	4.8	0.12	0.05	New
$1000 \times 500 \times 50$	36.10	187.50	18.5	2.89	0.04	Zhang
	32.40	130.80	12.75	1.06	0.03	New

Table 1: The results of performing new algorithm and Zhang's algorithm on some random datasets