Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 18, No. 2, pp. 333–348, 2022

EFFICIENT METHODS FOR CONVEX PROBLEMS WITH BREGMAN BARZILAI-BORWEIN STEP SIZES*

Yifan Shao, Qingsong Wang and Deren Han†

Abstract: The Barzilai-Borwein (BB) method is a popular and efficient algorithm for solving convex optimization problems, which is a gradient method while the step size is selected under some quasi-Newton idea, measured in the Euclidean distance. In this paper, we apply a more general distance measure (e.g., the Bregman divergence) to the BB method. We derive several new BB step sizes formulae and apply them to the mirror descent method and the Frank-Wolfe method. Compared with the two algorithms with traditional BB step sizes, the preliminary numerical experiments for the real data demonstrate that the algorithms with the proposed step sizes are efficient, and can achieve better performance in terms of taking less CPU time to achieve better objective value.

Key words: *Barzilai-Borwein step size, Bregman divergence, mirror descent method, convex optimization*

Mathematics Subject Classification: *90C26, 90C30*

1 Introduction

Our optimization problem of interest is

$$\min_{x\in\mathcal{C}} \; f(x) \tag{1.1}$$

where $\mathcal{C} \subseteq \mathbb{R}^n$ is a nonempty closed convex set and $f : \mathcal{C} \to \mathbb{R}$ is a closed, proper, convex differentiable function. The optimal set of problem (1.1), denoted by X^*, is nonempty. This problem arises in many applications, such as compressed sensing [19], image processing [25], machine learning [29] , and data mining [34].

Due to good performance in solving large-scale optimization problems (1.1) arising from practical applications, the gradient method is very popular [19, 13]. The well-known steepest descent (SD) method is defined as

$$x_{k+1} = x_k - \eta_k \nabla f(x_k), \tag{1.2}$$

where $\eta_k > 0$ is determined by the exact line search as

$$\eta_k = \arg\min_{\eta>0} f(x_k - \eta\nabla f(x_k)). \tag{1.3}$$

*This research is supported by the National Natural Science Foundation of China (NSFC) grants 11625105 and 12131004.

†Corresponding author

Though theoretically, under some conditions the SD method is Q-linearly convergent, it can be very slow, especially when the Hessian of f is ill-conditioned [1]. The Barzilai-Borwein [5] gradient (BB) method, in some extension, avoids the drawback of the SD method and performs much better than the SD method in practice. Hence, the BB gradient algorithms get great attention.

The BB gradient algorithm was proposed by Barzilai and Borwein [5], which still utilizes the negative gradient direction as the search direction, while the step size is not directly selected by a line search manner. The BB gradient algorithm uses the information of the current iteration point and the previous iteration point to determine the BB step size. There are two choices of the step size, i.e.,

$$\eta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}} \quad \text{or} \quad \eta_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}, \tag{1.4}$$

where $s_{k-1} = x_k - x_{k-1}$ and $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$. Barzilai and Borwein [5] presented a convergence analysis in the two-dimensional quadratic case. They established the R-superlinear convergence of the BB method. Based on the work of Raydan [32], Birgin, Martine, and Raydan [8] proposed an effective gradient projection BB algorithm to solve convex constrained optimization problems. Dai and Fletcher [13] studied the projection BB algorithm for solving large-scale constrained quadratic programming. Dai [12] proposed a new gradient algorithm that used the step sizes of the SD algorithm and the BB algorithm alternately. Kafaki and Fatemi [3] derived a new two-point step size from the modified quasi-Newton equation. Dai et al. [15] used the geometric average of two BB steps to get a new step size. Dai and Kou [14] proposed the BB conjugate gradient method by combining the BB algorithm with the conjugate gradient algorithm. Huang et al. [26] introduced a new mechanism to make the BB method have the two-dimensional quadratic termination property. Dai et al. [16] proposed a family of spectral gradient methods, whose step size was determined by a convex combination of the long BB step size and the short BB step size. He et al. [23] proposed to solve variational inequality problems with BB step size projection methods and reported convincing numerical results. For more works on BB-like methods see [21] and references therein.

We observe that the research of the BB step size is based on the Euclidean divergence. It is known that the Euclidean divergence is a kind of the Bregman divergence [9]. It is natural to consider the BB step size based on the Bregman divergence. The mirror descent algorithm [28] is a first-order optimization algorithm that generalizes the classic gradient descent method by the Bregman divergence. It performs better than the classic gradient method in some problems, especially on the unit simplex [7]. Inspired by the mirror descent algorithm, we introduce a method of computing the BB step size based on the Bregman divergence in this paper. We compare the Bregman BB step size with the Euclidean BB step size in the same methods including the mirror descent method and the Frank-Wolfe method.

Our main contributions are summarized as follows: Firstly, we propose a BB step size based mirror descent method, namely MDBB, which applies the classic BB step size to the mirror descent method directly, see Subsection 3.1 for details. Secondly, based on the above MDBB algorithm, we establish a more general MDBB (denote MDBB-I) algorithm that the MDBB algorithm can be regarded as a special case of MDBB-I algorithm, the details can refer to Subsection 3.2. Thirdly, a more general BB step size is applied to the MDBB-I, we get a new algorithm, namely MDBB-II which can cover MDBB and MDBB-I algorithms, and refer to Subsection 3.3 for details. Fourthly, the above three BB step size frameworks are applied to the FW algorithm and get three BB step size based FW algorithms, see Section

4 for details. Finally, the all above algorithms are verified by some numerical experiments (Section 5) which shows the effectiveness of general BB step size techniques proposed in this paper.

The rest of this paper is organized as follows. In Section 2, we briefly review the related theoretical results. In Section 3 and Section 4, the BB step size based on the Bregman divergence is introduced in detail. Results of numerical experiments on the D-optimal design problem are reported and discussed in Section 5. Finally, we conclude this paper in Section 6.

Notation. We start by establishing the notation used throughout the paper. We use $\|\cdot\|$ for the Euclidean norm $\|\cdot\|_2$ and the standard notation $\langle\cdot,\cdot\rangle$ for the Euclidean inner product. We recall that for any set C. $\overline{C}$ denotes the closure of C and int C denotes the interior of C. The effective domain of a function f, i.e., set of all x such that $f(x) < \infty$ is denoted by dom f. The inverse of a function f is denoted by f^{-1}. $\nabla f(x)^{-1}$ is the inverse of $\nabla f(x)$. The other notations are standard from convex analysis [33, 6].

2 Preliminaries

In this section, we recall some theoretical results that will be useful for the analysis in Section 3.

2.1 Bregman Divergence

The Bregman divergence defines a more general divergence, which is widely used in clustering [20, 4], machine learning [31], and information geometry [2]. The definition of the Bregman divergence is the following.

Definition 2.1 (Kernel function [17])**.** A function $\phi : \mathbb{R}^n \to \mathbb{R}\bigcup +\infty$ is called a kernel function on $\mathcal{C}$ if

(i) ϕ is closed convex proper(c.c.p.),

(ii) $\overline{\mathrm{dom}\phi} = \mathcal{C}$, where $\overline{\mathrm{dom}\phi}$ denotes the closure of domϕ.

(iii) ϕ is continuously differentiable and strictly convex on int dom $\phi \neq \varnothing$.

A kernel function ϕ induces a Bregman divergence $D_\phi(\cdot,\cdot)$ defined as

$$D_\phi(x,y) = \phi(x) - \phi(y) - \langle\nabla\phi(y), x-y\rangle \ \ \forall x \in \mathrm{dom}\phi, y \in \mathrm{int\ dom}\phi. \tag{2.1}$$

Examples [4] of the kernel function ϕ and the induced Bregman divergences are listed in Table 1.

2.2 Mirror Descent

We now introduce the mirror descent algorithm. The most common approach to constructing a sequence $\{x_k\}_{k=1}^n$ is based on the gradient descent. The gradient descent update is

$$x_{k+1} = x_k - \eta_k \nabla f(x_k),$$

where $\{\eta_k\}_{k=1}^n$ denotes a sequence of step sizes. Note that the gradient descent step can alternatively be expressed as

$$x_{k+1} = \arg\min_{x\in\mathcal{C}}\{f(x_k) + \langle\nabla f(x_k), x-x_k\rangle + \frac{1}{2\eta_k}\|x-x_k\|^2\}.$$

Domain	$\phi(x)$	$D_\phi(x,y)$
$\mathbb{R}$	x^2	$(x-y)^2$
$\mathbb{R}_+$	$x\log x$	$x\log(\frac{x}{y})-(x-y)$
$[0,1]$	$x\log x+(1-x)\log(1-x)$	$x\log\frac{x}{y}+(1-x)\log\frac{1-x}{1-y}$
$\mathbb{R}$	e^x	$e^x-e^y-(x-y)e^y$
$\mathbb{R}^n$	$\|x\|^2$	$\|x-y\|^2$
$\mathbb{R}^n$	x^TAx	$(x-y)^TA(x-y)$
$n-$Simplex	$\sum_{j=1}^n x^{(j)}\log_2 x^{(j)}$	$\sum_{j=1}^n x^{(j)}\log_2(\frac{x^{(j)}}{y^{(j)}})$
$\mathbb{R}_+^n$	$\sum_{j=1}^n x^{(j)}\log x^{(j)}$	$\sum_{j=1}^n x^{(j)}\log(\frac{x^{(j)}}{y^{(j)}})-\sum_{j=1}^n(x^{(j)}-y^{(j)})$

Table 1: Bregman divergences generated from some convex functions.

By re-expressing the gradient step in this way, Nemirovski and Yudin [30] introduced a generalization of gradient descent as follows:

$$x_{k+1}=\arg\min_{x\in\mathcal{C}}\{f(x_k)+\langle\nabla f(x_k),x-x_k\rangle+\frac{1}{\eta_k}h(x,x_k)\}. \tag{2.2}$$

When $h(\cdot,\cdot):\mathbb{R}^n\times\mathbb{R}^n\to\mathbb{R}_+$ is a penalty function. Clearly, when $h(x,x_k)=\frac{1}{2}\|x-x_k\|^2$ and $\mathcal{C}=\mathbb{R}^n$, we can immediately derive the standard gradient descent method. Hence, the iterative scheme (2.2) is a generalization of gradient descent method. A standard choice for the penalty function $h(\cdot,\cdot)$ is called the Bregman divergence. Let $h(\cdot,\cdot)=D_\phi(\cdot,\cdot)$, the mirror descent step is defined as

$$x_{k+1}=\arg\min_{x\in\mathcal{C}}\{f(x_k)+\langle\nabla f(x_k),x-x_k\rangle+\frac{1}{\eta_k}D_\phi(x,x_k)\}.$$

3 Mirror Descent Method

In this section, we present the mirror descent Barzilai-Borwein (MDBB) method and its two variants for solving problem (1.1).

3.1 Mirror Descent Barzilai-Borwein Method

We first consider the unconstrained problem. The mirror descent step is

$$x_{k+1}=\arg\min_{x}\{f(x_k)+\langle\nabla f(x_k),x-x_k\rangle+\frac{1}{\eta_k}D_\phi(x,x_k)\}.$$

Finding the minimum by differentiation yields the step

$$\nabla\phi(x_{k+1})=\nabla\phi(x_k)-\eta_k\nabla f(x_k),$$

or equivalently,

$$x_{k+1}=\nabla\phi^{-1}(\nabla\phi(x_k)-\eta_k\nabla f(x_k)). \tag{3.1}$$

According to (3.1), we list some of the most classical examples.

Example 3.1. When $\phi(x) = \frac{1}{2}\|x\|^2$, $x \in \mathbb{R}^n$, the gradient vector is $\nabla\phi(x) = x$. Then it shows that

$$x_{k+1} = x_k - \eta_k \nabla f(x_k).$$

It is the standard gradient descent step.

Example 3.2. When $\phi(x) = \sum_{i=1}^n x^{(i)}\log(x^{(i)}) - x^{(i)}$, $x \in \mathbb{R}_+^n$, the gradient vector is $\nabla\phi(x) = (\log x^{(1)}, \log x^{(2)}, \cdots, \log x^{(n)})^T$. We can get

$$x_{k+1}^{(i)} = e^{\log(x_k^{(i)}) - \eta_k \nabla f(x_k^{(i)})} = x_k^{(i)} e^{(-\eta_k \nabla f(x_k^{(i)}))}, i = 1, \cdots, n.$$

Example 3.3. When $\phi(x) = \sum_{i=1}^n -\log x^{(i)}$, $x \in \mathbb{R}_{++}^n$, the gradient vector is $\nabla\phi(x) = (-\frac{1}{x^{(1)}}, \cdots, -\frac{1}{x^{(n)}})^T$. The update of variable x is

$$x_{k+1}^{(i)} = \frac{x_k^{(i)}}{\eta_k x_k^{(i)} \nabla f(x_k)^{(i)} + 1}.$$

Example 3.4. When $\phi(x) = \sum_{i=1}^n e^{x^{(i)}}$, $x \in \mathbb{R}^n$, the gradient vector is

$$\nabla\phi(x) = (e^{x^{(1)}}, e^{x^{(2)}}, \cdots, e^{x^{(n)}})^T.$$

We can get

$$x_{k+1}^{(i)} = \log(e^{x_k^{(i)}} - \eta_k \nabla f(x_k^{(i)})).$$

In this paper, we use the gradient descent step (3.1) to update variable and use (1.4) to compute the BB step size. We name the new method as the Mirror Descent Barzilai-Borwein (MDBB) method. We have to search for the minimizer of the problem (1.1) within the nonempty closed convex set $\mathcal{C}$. This point x'_{k+1} computed by (3.1) might not be in the convex feasible region $\mathcal{C}$, so we project x'_{k+1} back to a close by x_{k+1} in $\mathcal{C}$, i.e.,

$$x_{k+1} = \mathcal{P}_{x\in\mathcal{C}}(x, x'_{k+1}) = \arg\min_{x\in\mathcal{C}} \|x - x'_{k+1}\|^2.$$

A formal description of the MDBB algorithm is as follows.

Algorithm 1 MDBB: The mirror descent Barzilai-Borwein method

Initialization: select an initial point x_0, an initial step η_0.
$x'_1 = \arg\min_x\{f(x_0) + \langle\nabla f(x_0), x - x_0\rangle + \frac{1}{\eta_0} D_\phi(x, x_0)\}$,
$x_1 = \mathcal{P}_{x\in\mathcal{C}}(x, x'_1)$.
for $k = 1$ to n **do**
  $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$,
  $\eta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ or $\eta_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$,
  $x'_{k+1} = \arg\min_x\{f(x_k) + \langle\nabla f(x_k), x - x_k\rangle + \frac{1}{\eta_k} D_\phi(x, x_k)\}$,
  $x_{k+1} = \mathcal{P}_{x\in\mathcal{C}}(x, x'_{k+1})$.
end for

$\boxed{3.2}$ MDBB-I

In this subsection, we present a variant of the MDBB, named MDBB-I, for solving the problem (1.1). We review the origin of the quasi-Newton method. The Taylor expansion is applied to the objective function $f(x)$ at the point x_k, and the second-order approximation is as follows

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T G_k (x - x_k), \tag{3.2}$$

where $G_k = \nabla^2 f(x_k)$ is Hessian matrix of $f(x_k)$. Now, we find the gradient of function $f(x)$ and deduce its derivatives in detail,

$$\nabla f(x) \approx \nabla f(x_k) + G_k (x - x_k). \tag{3.3}$$

Set $x = x_{k-1}$ in (3.3), we obtain

$$\nabla f(x_k) - \nabla f(x_{k-1}) \approx G_k (x_k - x_{k-1}). \tag{3.4}$$

Let $s_{k-1} = x_k - x_{k-1}, y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$, (3.4) can be expressed as

$$y_{k-1} \approx G_k s_{k-1} \quad \text{or} \quad s_{k-1} \approx G_k^{-1} y_{k-1} \tag{3.5}$$

which is called the quasi-Newton condition. Let B_k approximates G_k, the quasi-Newton condition can be expressed as

$$y_{k-1} = B_k s_{k-1} \quad \text{or} \quad s_{k-1} = B_k^{-1} y_{k-1}. \tag{3.6}$$

In the BB step size, $B_k = \frac{1}{\eta_k} I$ is required to satisfy the quasi-Newton condition.

$$\begin{aligned} &\min_{\eta} ||B_k s_{k-1} - y_{k-1}||^2 \\ \text{or} \quad &\min_{\eta} ||s_{k-1} - B_k^{-1} y_{k-1}||^2. \end{aligned} \tag{3.7}$$

The η_k in (1.4) is the solution of the two optimization problems in (3.7), respectively.

Based on the generation procedure of the BB step size, we present a new BB step size. In the mirror descent method,

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{\eta_k} D_\phi(x, x_k). \tag{3.8}$$

Differentiate both sides of the formula (3.8),

$$\nabla f(x) - \nabla f(x_k) \approx \frac{1}{\eta_k} (\nabla \phi(x) - \nabla \phi(x_k)). \tag{3.9}$$

In (3.9), set $x = x_{k-1}$, $s_k = \nabla\phi(x_k) - \nabla\phi(x_{k-1})$, and $y_k = \nabla f(x_k) - \nabla f(x_{k-1})$. Let $B_k = \frac{1}{\eta_k} I$ and make it satisfy the quasi-Newton condition (3.6). Combined with (3.7), we can get two new step sizes η_k. The current $s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1})$ is more general than the original $s_{k-1} = x_k - x_{k-1}$. Here are three examples to illustrate the difference.

Example 3.5. When $\phi(x) = \frac{1}{2}\|x\|^2$, $x \in \mathbb{R}^n$, the $D_\phi(x, y) = \frac{1}{2}\|x - y\|^2$, it shows that

$$s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1}) = x_k - x_{k-1}.$$

Example 3.6. When $\phi(x) = \sum_{i=1}^{n} -\log(x^{(i)})$, $x \in \mathbb{R}^n_{++}$, the associated Bregman divergence is

$$D_\phi(x, y) = \sum_{i=1}^{n} (-\log(\frac{x^{(i)}}{y^{(i)}}) + \frac{x^{(i)}}{y^{(i)}} - 1).$$

we get

$$\begin{aligned} s_{k-1} &= \nabla\phi(x_k) - \nabla\phi(x_{k-1}) \\ &= (\frac{1}{x_k^{(1)}} - \frac{1}{x_{k-1}^{(1)}}, \frac{1}{x_k^{(2)}} - \frac{1}{x_{k-1}^{(2)}}, \cdots, \frac{1}{x_k^{(n)}} - \frac{1}{x_{k-1}^{(n)}})^T. \end{aligned}$$

Example 3.7. When $\phi(x) = \sum_{i=1}^{n} e^{x^{(i)}}$, $x \in \mathbb{R}^n$, the associated Bregman divergence

$$D_\phi(x, y) = \sum_{i=1}^{n} e^{x^{(i)}} - \sum_{i=1}^{n} e^{y^{(i)}} - \nabla\phi(y)^T(x - y).$$

We obtain $s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1}) = (e^{x_k^{(1)}} - e^{x_{k-1}^{(1)}}, \cdots, e^{x_k^{(n)}} - e^{x_{k-1}^{(n)}})^T$.

We apply the type of BB step size to the MDBB method (Algorithm 1) to get the new algorithm, i.e., the MDBB-I method, which is formally described below.

Algorithm 2 MDBB-I: Variant I of the mirror descent Barzilai-Borwein method

1: Initialization: select an initial point x_0, an initial step η_0.
2: $x_1' = \arg\min_x \{f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \frac{1}{\eta_0} D_\phi(x, x_0)\}$.
3: $x_1 = \mathcal{P}_{x \in \mathcal{C}}(x, x_1')$.
4: **for** $k = 1$ to n **do**
5: $\quad s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1})$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$,
6: $\quad \eta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ or $\eta_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$,
7: $\quad x_{k+1}' = \arg\min_x \{f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{1}{\eta_k} D_\phi(x, x_k)\}$,
8: $\quad x_{k+1} = \mathcal{P}_{x \in \mathcal{C}}(x, x_{k+1}')$.
9: **end for**

3.3 MDBB-II

The problem (3.7) can be seen as minimizing the distance of $B_k s_{k-1}$ and y_{k-1} (or s_{k-1} and $B_k^{-1} y_{k-1}$) based on the Euclidean divergence. So we can use the Bregman divergence $D_\phi(B_k s_{k-1}, y_{k-1})$ to replace the $\|B_k s_{k-1} - y_{k-1}\|^2$ (Also, $\|s_{k-1} - B_k^{-1} y_{k-1}\|^2$ can be replace by $D_\phi(s_{k-1}, B_k^{-1} y_{k-1})$.), which is a general method to minimize the distance. We use two examples to illustrate this method.

Example 3.8. Let $\phi(x) = \frac{1}{2}\|x\|^2$, where $x \in \mathbb{R}^n$. The Bregman divergence associated with ϕ is

$$D_\phi(y, x) = \frac{1}{2}\|y - x\|^2.$$

To minimize $D_\phi(B_k^{-1} y_{k-1}, s_{k-1})$ is equivalent to minimize $\|B_k^{-1} y_{k-1} - s_{k-1}\|^2$. The step size is the original BB step size.

Example 3.9. Let $\phi(x) = \frac{1}{4}\|x\|^4$ defined on $\mathbb{R}^n$. The Bregman divergence associated with $\phi(x)$ is

$$\begin{aligned} D_\phi(y,x) &= \phi(y) - \phi(x) - \langle \nabla\phi(x), y - x\rangle \\ &= \frac{1}{4}\|y\|^4 + \frac{3}{4}\|x\|^4 - \|x\|^2 \cdot x^T y. \end{aligned}$$

Let $B_k = \frac{1}{\eta_k} I$ and we have

$$\begin{aligned} D_\phi(B_k^{-1}y_{k-1}, s_{k-1}) &= \frac{1}{4}\|B_k^{-1}y_{k-1}\|^4 + \frac{3}{4}\|s_{k-1}\|^4 - \|s_{k-1}\|^2 \cdot s_{k-1}^T B_k^{-1} y_{k-1} \\ &= \frac{1}{4}\eta_k^4\|y_{k-1}\|^4 + \frac{3}{4}\|s_{k-1}\|^4 - \|s_{k-1}\|^2 \cdot s_{k-1}^T \cdot \eta_k y_{k-1}. \end{aligned} \tag{3.10}$$

To minimize (3.10) for η_k, we get

$$\eta_k = \left(\frac{\|s_{k-1}\|^2 \cdot s_{k-1}^T y_{k-1}}{\|y_{k-1}\|^4}\right)^{\frac{1}{3}}.$$

This is a new BB step size based on the Bregman divergence (3.10). We apply this BB step size to the MDBB-I algorithm (Algorithm 2) to get a new method, i.e., MDBB-II algorithm, which is formally described below.

Algorithm 3 MDBB-II: Variant II of the mirror descent Barzilai-Borwein method

Initialization: select an initial point x_0, an initial step η_0.
$x_1' = \arg\min_x \{f(x_0) + \langle \nabla f(x_0), x - x_0\rangle + \frac{1}{\eta_0} D_\phi(x, x_0)\}$.
$x_1 = \mathcal{P}_{x\in\mathcal{C}}(x, x_1')$.
for $k = 1$ to n **do**
    $s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1})$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$,
    $\eta_k = \left(\frac{\|s_{k-1}\|^2 \cdot s_{k-1}^T y_{k-1}}{\|y_{k-1}\|^4}\right)^{\frac{1}{3}}$,
    $x_{k+1}' = \arg\min_x \{f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{\eta_k} D_\phi(x, x_k)\}$,
    $x_{k+1} = \mathcal{P}_{x\in\mathcal{C}}(x, x_{k+1}')$.
end for

4 Frank-Wolfe Algorithm

The Frank-Wolfe (FW) algorithm is also known as the projection-free or condition gradient algorithm [22]. The main advantages of this algorithm are to avoid the projection step and to ensure that the update vector remains inside the feasible domain. The method is formally described below.

There are various ways to set the parameter η_k in order to guarantee the convergence of the FW method. We apply the Bregman BB step size above to the FW method.

(1) FWBB: $\eta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ or $\eta_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$, where $s_{k-1} = x_k - x_{k-1}$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

(2) FWBB-I: $\eta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ or $\eta_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$, where $s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1})$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

Algorithm 4 FW: Frank-Wolfe Algorithm

Input:$f : \mathcal{C} \to \mathbb{R}$.
Initialize:any $x_1 \in \mathcal{C}$.
for $k = 1 \cdots N$ **do**
  $v_k = \arg\min_{v \in \mathcal{C}} \langle v, \nabla f(x_k) \rangle$,
  $x_{k+1} = (1 - \eta_k)x_k + \eta_k v_k$, $\eta_k \in [0, 1]$.
end for

(3) FWBB-II: $\eta_k = \left(\frac{\|s_{k-1}\|^2 \cdot s_{k-1}^T y_{k-1}}{\|y_{k-1}\|^4} \right)^{\frac{1}{3}}$, where $s_{k-1} = \nabla\phi(x_k) - \nabla\phi(x_{k-1})$, $y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$.

Specifically, the FWBB method is formally described below.

Algorithm 5 FWBB: Frank-Wolfe Barzilai-Borwein Algorithm

Input:$f : \mathcal{C} \to \mathbb{R}$.
Initialize:any $x_0 \in \mathcal{C}$, an initial step $\eta_0 \in [0, 1]$.
$v_1 = \arg\min_{v \in \mathcal{C}} \langle v, \nabla f(x_0) \rangle$.
$x_1 = (1 - \eta_0)x_0 + \eta_0 v_1$.
for $k = 1 \cdots N$ **do**
  $s_{k-1} = x_k - x_{k-1}, y_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$,
  $\eta_k = \frac{s_{k-1}^T s_{k-1}}{s_{k-1}^T y_{k-1}}$ or $\eta_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}$,
  $v_k = \arg\min_{v \in \mathcal{C}} \langle v, \nabla f(x_k) \rangle$,
  $x_{k+1} = (1 - \eta_k)x_k + \eta_k v_k$.
end for

Remark: To ensure convergence, the Frank-Wolfe requires some bounds on the parameter η_k. In this paper, we choose $\eta_k = \max(\min(\eta_k, 1), 0.0001)$ to ensure the BB stepsize in FWBB meet the requirements of the Frank-Wolfe Algorithm.

5 Numerical Results

In this section, we present the numerical results. We use two types of the gradient algorithms outlined in Section 3 and Section 4, i.e, the mirror descent algorithm and the FW algorithm, to clarify that the Bregman BB step size can improve the convergence rate of the algorithm compared with the Euclidean BB step size. All the codes are written in Python 3.8 and run on a personal computer with Intel (R) Core (TM) i5-8265u CPU @ 1.60GHz 1.80GHz, 8G RAM.

An appropriate stopping criterion for any optimization algorithm is paramount to ensure that an accurate solution is located. Our termination criterion is

$$|f(x_k) - f(x_{k-1})| < \epsilon,$$

where $f(x_k)$ are function value at k iteration and ϵ is some user-defined tolerance, such as 10^{-3}, 10^{-4}. This stopping criterion is a relatively common condition, see [11, 18, 24] for details. When the iteration termination condition reaches the accuracy or the number of iterations reaches the maximum number of iterations (Nmax=30000), the operation stops. The stopping criterion is implemented in all numerical results.

In statistics, the D-optimal design problem corresponds to maximizing the determinant of the Fisher information matrix [27]. The D-optimal design problem is defined as

$$\begin{aligned} &\min f(x) := -\log(\det(\textstyle\sum_{i=1}^{n} x^{(i)} v_i v_i^T)) \\ &\text{s.t} \textstyle\sum_i^n x^{(i)} = 1, x^{(i)} \geq 0, i = 1, \cdots, n, \end{aligned}$$

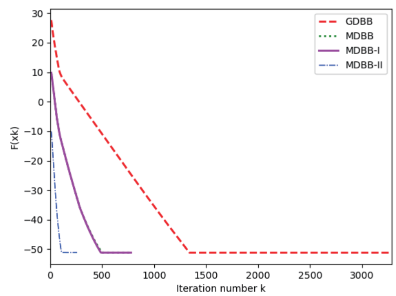
where $v_i \in \mathbb{R}^m, i = 1, \cdots, n,\ n \geq m+1$. We construct the D-optimal design instances from LibSVM data [10]. In particular, we consider several regression datasets - the goal is to find the most relevant data points where one shall run the experiment to evaluate the corresponding label.

In this part, we solve the D-optimal design problem by using Gradient Descent Barzilai-Borwein(GDBB), MDBB, MDBB-I and MDBB-II algorithms. In MDBB, MDBB-I, MDBB-II algorithms, we choose $\phi(x) = \sum_{i=1}^{n} e^{x^{(i)}}$ for simplicity.

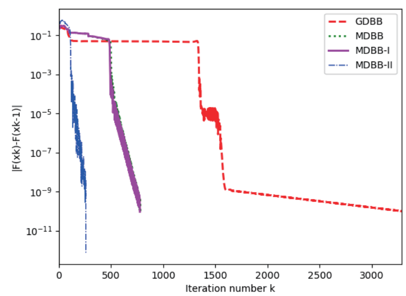
Table 2: Performance comparison on D-optimal design problems on six different datasets: bodyfat ($n = 252, m = 14$), housing ($n = 506, m = 13$), mpg ($n = 392, m = 7$), space-ga ($n = 3107, m = 6$), mg ($n = 1385, m = 6$), cpusmall ($n = 8192, m = 12$). "iter" is the number of iterations, "time[s]" is CPU time in seconds, "the absolute value of relative error" is $|f(x_k) - f(x_{k-1})|$.

Data	Algorithm	$\epsilon = 10^{-8}$		$\epsilon = 10^{-10}$		$\epsilon = 10^{-12}$	
		iter	time[s]	iter	time[s]	iter	time[s]
bodyfat	GDBB	601	0.441	618	0.494	632	0.487
	MDBB	250	0.221	273	0.294	295	0.301
	MDBB-I	244	0.234	268	0.241	292	0.287
	MDBB-II	88	0.095	107	0.116	122	0.130
housing	GDBB	1574	1.434	3285	3.229	6383	7.015
	MDBB	678	0.718	782	0.877	886	1.080
	MDBB-I	673	0.668	777	0.903	889	1.050
	MDBB-II	214	0.238	258	0.328	258	0.391
mpg	GDBB	1027	0.730	1099	0.982	1174	1.145
	MDBB	659	0.653	734	0.644	810	0.680
	MDBB-I	624	0.467	690	0.707	756	0.628
	MDBB-II	174	0.160	197	0.208	232	0.243
space-ga	GDBB	5364	15.045	5559	15.339	5853	15.316
	MDBB	3393	9.024	4205	10.632	4854	11.816
	MDBB-I	3838	9.295	3838	9.762	4372	10.614
	MDBB-II	412	1.395	457	1.347	499	1.264
mg	GDBB	6018	14.237	11378	23.287	16429	23.215
	MDBB	2605	5.314	4621	9.145	6921	10.673
	MDBB-I	2071	4.350	3274	7.837	4580	7.020
	MDBB-II	284	0.741	370	0.950	472	0.792
cpusmall	GDBB	10948	90.454	11134	93.223	11202	129.883
	MDBB	4284	34.422	4634	41.444	4993	55.218
	MDBB-I	4179	34.878	4493	38.409	4818	56.013
	MDBB-II	484	4.451	579	5.225	660	7.507

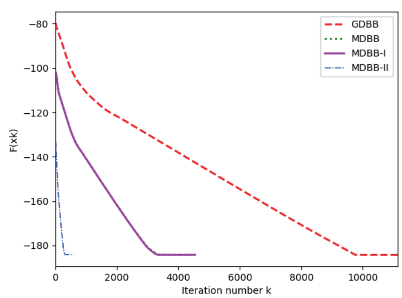
It can be seen from the Figure 1 and Table 2 that the Bregman BB step size can improve the convergence rate of the algorithm compared with the Euclidean BB step size. In particular, the time cost of the MDBB-II method is greatly reduced and its convergence rate is



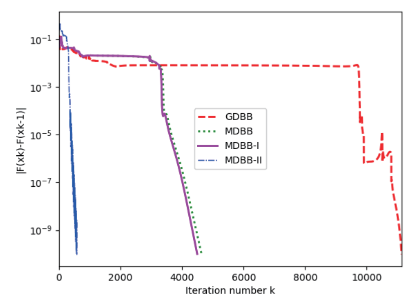
(a) The value of the objective function on housing data.



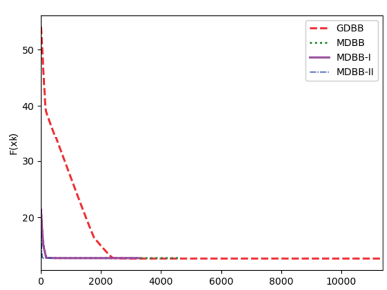
(b) The absolute value of relative error on housing data.



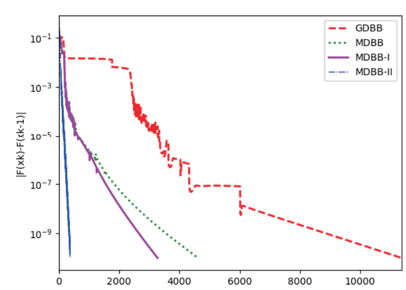
(c) The value of the objective function on cpusmall data.



(d) The absolute value of relative error on cpusmall data.



(e) The value of the objective function on mg data.



(f) The absolute value of relative error on mg data.

Figure 1: Comparison of GDBB, MDBB, MDBB-I, and MDBB-II algorithms on the D-optimal design problem in housing, cpusmall, and mg data, where tol$=10^{-10}$.The first column is the value of the objective function. The second column is the absolute value of relative error.

faster compared with the others.

In MDBB, we only change the generation of x_k. But this method reduce nearly half of the number of the iterations of the GDBB, which is really effective. The performance of the MDBB-I is similar to the MDBB. In the MDBB-II method, we use $\min_\eta D_\phi(B_k^{-1}y_{k-1}, s_{k-1})$ to replace $\min_\eta \|B_k^{-1}y_{k-1} - s_{k-1}\|^2$, which is a more generalized way to measure the gap between $B_k^{-1}y_{k-1}$ and s_{k-1}. In the MDBB-II method, we used the BB step size $\eta_k = \left(\frac{\|s_{k-1}\|^2 \cdot s_{k-1}^T y_{k-1}}{\|y_{k-1}\|^4}\right)^{\frac{1}{3}}$.

Next, we use the FWBB, FWBB-I, and FWBB-II methods mentioned above to solve the D-optimal design problem. In the FW algorithm numerical experiments, we project the BB step size in $[0, 1]$ to satisfy the FW algorithm. Our numerical results are shown here.

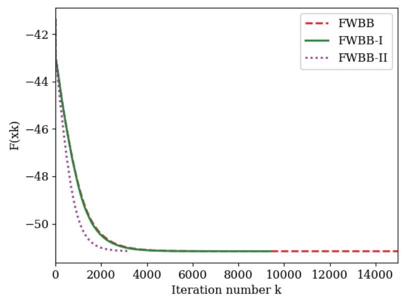
Table 3: Performance comparison on D-optimal design problems on four different datasets: housing ($n = 506, m = 13$), mg ($n = 1385, m = 6$), abalone ($n = 4177, m = 8$), cpusmall ($n = 8192, m = 12$). "iter" is the number of iterations, "time[s]" is CPU time in seconds, "the absolute value of relative error" is $|f(x_k) - f(x_{k-1})|$.

Data	eplison	$\epsilon = 10^{-4}$		$\epsilon = 10^{-6}$		$\epsilon = 10^{-8}$	
		iter	time[s]	iter	time[s]	iter	time[s]
housing	FWBB	2772	0.415	3771	0.374	14967	1.470
	FWBB-I	2694	0.395	3409	0.398	9433	1.010
	FWBB-II	2056	0.284	2220	0.297	3217	0.436
mg	FWBB	4281	0.445	8631	0.903	11576	1.167
	FWBB-I	4172	0.528	7972	0.993	10379	1.359
	FWBB-II	1644	0.248	2208	0.323	2715	0.394
abalone	FWBB	2784	0.565	6186	1.289	14380	2.962
	FWBB-I	2752	1.946	5900	1.629	10284	5.290
	FWBB-II	2559	0.946	2936	0.841	5319	1.727
cpusmall	FWBB	12526	3.333	6362	1.322	25307	6.961
	FWBB-I	12384	5.798	5657	1.466	22666	10.396
	FWBB-II	9252	4.305	2850	0.876	11985	5.590

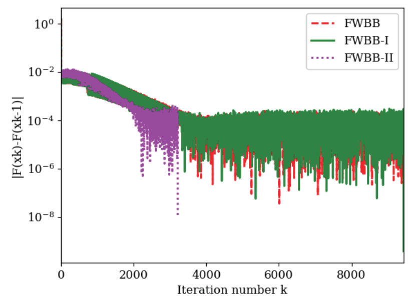
It can be seen from Figure 2 and Table 3 that the FWBB-I and the FWBB-II can reduce the number of iterations and improve the convergence rate of the FW method compared with the FWBB. On cpusmall data, the FWBB-II and FWBB-I spend more time than the FWBB. The main reason is the Bregman BB step size has exponential function and logarithmic function, it spend much time to compute the Bregman BB step size.

6 Conclusion

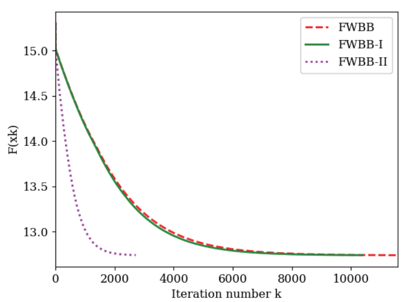
In this paper, we proposed some BB step sizes based on the Bregman divergence. We also applied these new BB step sizes formulae to the mirror descent method and the Frank-Wolfe method for the convex optimization problem. And we implemented these methods to the D-optimal design problem and reported some preliminary results to demonstrate the effectiveness of the proposed algorithms.



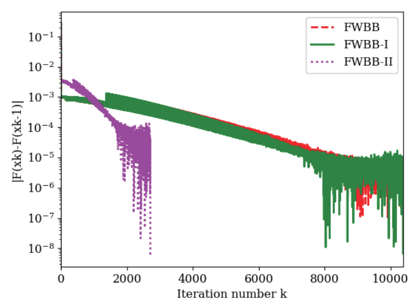
(a) The value of the objective function on housing data.



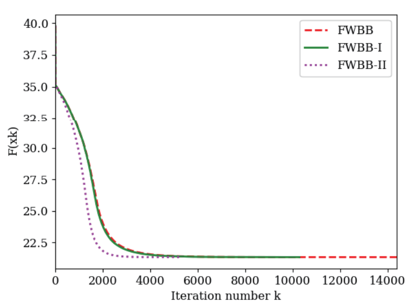
(b) The absolute value of relative error on housing data.



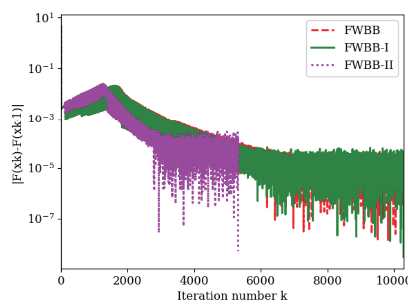
(c) The value of the objective function on mg data.



(d) The absolute value of relative error on mg data.



(e) The value of the objective function on abalone data.



(f) The absolute value of relative error on abalone data.

Figure 2: Comparison of FWBB-I, FWBB-II, and FWBB on D-optimal design problems in housing, mg, and abalone data, where tol=10^{-8}. The first column is the value of the objective function. The second column is the absolute value of relative error.

Acknowledgment

The authors would like to appreciate two anonymous reviewers for their insightful comments and constructive suggestions, which help us improve the paper greatly.

References

[1] H. Akaike, On a successive transformation of probability distribution and its application to the analysis of the optimum gradient method, *Ann. I. Stat. Math.* 11 (1959) 1–16.

[2] S. Amari and A. Cichocki, Information geometry of divergence functions, *B. Pol. Acad. Sci-Tech.* 58 (2010) 183–195.

[3] S. Babaie-Kafaki and M. Fatemi, A modified two-point stepsize gradient algorithm for unconstrained minimization, *Optim. Methods Softw.* 28 (2013) 1040–1050.

[4] A. Banerjee, S. Merugu, I.S. Dhillon and J. Ghosh, Clustering with Bregman divergences, *J. Mach. Learn. Res.* 6 (2005) 1705–1749.

[5] J. Barzilai and J.M. Borwein, Two-point step size gradient methods, *IMA J. Numer. Anal.* 8 (1988) 141–148.

[6] H.H. Bauschke and P.L. Combettes, Convex analysis and monotone operator theory in Hilbert spaces, *CMS Books in Mathematics*, (2001) 1–468.

[7] A. Bental, T. Margalit and A. Nemirovski, The ordered subsets mirror descent optimization method with applications to tomography, *SIAM J. Optim.* 12 (2001) 79–108.

[8] B.G. Birgin, J. M. Martínez and M. Raydan, Inexact spectral projected gradient methods on convex sets, *Ima J. Numer. Anal.* 23 (2003) 539–559.

[9] L.M. Bregman, The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming, *Ussr Compu. Math. Mathe. Phys.* 7 (1967) 200–217.

[10] C. Chang and C. Lin, LIBSVM: A library for support vector machines, *ACM Trans. Intell. Syst. Technol.* 2 (2011) 1–27.

[11] E. Cheung and Y.Y. Li, Projection free rank-drop steps, in: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017, pp. 1539–1545.

[12] Y.H. Dai and Y. Yuan, Alternate minimization gradient method, *Ima. J. Numer. Anal.* 3 (2003) 377–393.

[13] Y.H. Dai and R. Fletcher, Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming, *Numer. Math.* 100 (2005) 21–47.

[14] Y.H. Dai and C. Kou, A Barzilai-Borwein conjugate gradient method, *Sci. China Math.* 59 (2016) 1511–1524.

[15] Y.H. Dai, M. Al-Baali and X. Yang, A positive Barzilai-Borwein-like stepsize and an extension for symmetric linear systems, *Numer. Func. Anal. Opt.*, (2015) 59–75.

[16] Y.H. Dai, Y. Huang and X.W. Liu, A family of spectral gradient methods for optimization, *Comput. Optim. Appl.* 74 (2019) 43–65.

[17] R. Dragomir, A. Taylor, A. d'Aspremont and J. Bolte, Optimal complexity and certification of Bregman first-order methods, *Math. Program.* 2021.

[18] P.E. Dvurechensky, P. Ostroukhov, K. Safin, S. Shtern and M. Staudigl, Self-concordant analysis of Frank-Wolfe algorithms, in: *Proceedings of the 37th International Conference on Machine Learning*, vol. 119, 2020, pp. 2814–2824.

[19] M.A. T. Figueiredo, R.D. Nowak and S.J. Wright, Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems, *IEEE J-STSP.* 1 (2007) 586–597.

[20] A. Fischer, Quantization and clustering with Bregman divergences, *J. Multivar. Anal.* 101 (2010) 2207–2221.

[21] R. Fletcher, On the Barzilai-Borwein method, *Optimization and Control with Applications* (2005) 235–256.

[22] M. Frank and P. Wolfe, An algorithm for quadratic programming, *Nav. Res. Log.* 3 (1956) 95–110.

[23] H.J. He, D.R. Han and Z.B. Li, Some projection methods with the BB step sizes for variational inequalities, *J. Comp. Appl. Math.* 236 (2012) 2590–2604.

[24] Z. Harchaoui, A. Juditsky and A. Nemirovski, Conditional gradient algorithms for norm–regularized smooth convex optimization, *Math. Program.* 152 (2015) 75–112.

[25] H. Huang, Efficient reconstruction of 2D images and 3D surfaces, PhD thesis, *University of British Columbia*, 2008.

[26] Y. Huang, Y.H. Dai and X.W. Liu, Equipping Barzilai-Borwein method with two dimensional quadratic termination property, preprint, arXiv:2010.12130, 2020.

[27] J. Kiefer and J. Wolfowitz, The equivalence of two extremum problems, *Canadian J. Math.* 12 (2016) 363–366.

[28] H.H. Lu, R. M. Freund and Y. E. Nesterov, Relatively smooth convex optimization by first-order methods and applications, *SIAM J. Optim.* 28 (2018) 333–354.

[29] C. Mu, Y.Q. Zhang, J. Wright, and D. Goldfarb, Scalable robust matrix recovery: Frank-Wolfe meets proximal methods, *SIAM J. Sci. Comput.* 38 (2016) 3291–3317.

[30] A. S. Nemirovsky and D.B. Yudin, Problem complexity and method efficiency in optimization, *J. Oper. Res. Soc.* 1983.

[31] R. Nock, B. Magdalou, E. Briys, and F. Nielsen, *Matrix Information Geometry*, Springer Berlin Heidelberg, 2013.

[32] M. Raydan, The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem, *SIAM J. Optim.* 7 (1997) 26–33.

[33] R.T. Rockafellar, *Convex Analysis*, Princeton University Press, 1970.

[34] S. Patrick, *Optimization in Signal and Image Processing*, Wiley, 2009.

Manuscript received 30 June 2021
revised 5 September 2021
accepted for publication 19 October 2021

Yifan Shao
LMIB of the Ministry of Education
School of Mathematical Sciences
Beihang University
Beijing, 100191, China
E-mail address: sy1909123@buaa.edu.cn

Qingsong Wang
LMIB of the Ministry of Education
School of Mathematical Sciences
Beihang University, Beijing, 100191, China
E-mail address: nothing2wang@hotmail.com

Deren Han
LMIB of the Ministry of Education
School of Mathematical Sciences
Beihang University, Beijing, 100191, China
E-mail address: handr@buaa.edu.cn