Filomat 30:12 (2016), 3243–3252
DOI 10.2298/FIL1612243D

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

On Global Convergence Rate of Two Acceleration Projection Algorithms for Solving the Multiple-Sets Split Feasibility Problem

Qiao-Li Dong[a,b], Songnian He[a,b], Yanmin Zhao[c]

[a] *College of Science, Civil Aviation University of China, Tianjin 300300, China.*
[b] *Tianjin Key Lab for Advanced Signal Processing, Civil Aviation University of China, Tianjin 300300, China.*
[c] *School of Mathematical Sciences and Institute of Applied Mathematics, Xuchang University, Xuchang, Henan 461000, China*

Abstract. In this paper, we introduce two fast projection algorithms for solving the multiple-sets split feasibility problem (MSFP). Our algorithms accelerate algorithms proposed in [8] and are proved to have a global convergence rate $O(1/n^2)$. Preliminary numerical experiments show that these algorithms are practical and promising.

1. Introduction

The multiple-sets split feasibility problem (MSFP) was firstly proposed by Censor et al in [6] and arises in many practical fields, such as image reconstruction, signal processing, intensity-modulated radiation therapy (IMRT) and so on [6, 12]. The MSFP received much attention and many researchers proposed algorithms for solving it (see [8, 9, 14, 15, 17, 18, 20] and references therein). The MSFP is to find a point x^* satisfying

$$x^* \in C := \bigcap_{i=1}^{t} C_i \quad \text{such that} \quad Ax^* \in Q := \bigcap_{j=1}^{r} Q_j, \tag{1}$$

where A is an $M \times N$ real matrix, $C_i \subseteq \mathbb{R}^N$, $i = 1, \ldots, t$ and $Q_j \subseteq \mathbb{R}^M$, $j = 1, \ldots, r$ are the nonempty closed convex sets. Specially, when $t = r = 1$, the problem reduces to the split feasibility problem (SFP), which is to find $x^* \in C$ with $Ax^* \in Q$ (see, e.g. [4, 5]).

Assume that the MSFP (1) is consistent, i.e. its solution set, denoted by Γ, is nonempty. It is easy to see that the MSFP (1) is equivalent to the minimization problem

$$\min \frac{1}{2}\|x - P_C(x)\|^2 + \frac{1}{2}\|Ax - P_Q(Ax)\|^2$$

2010 *Mathematics Subject Classification*. 47H05, 47H07,47H10
Keywords. Multiple-sets split feasibility problem, Self-adaptive algorithm, Global rate of convergence, Projection algorithm.
Received: 11 September 2014; Accepted: 20 March 2015
Communicated by Ljubomir Ćirić
Research supported by National Natural Science Foundation of China (No. 61379102) and Fundamental Research Funds for the Central Universities (No. 3122016L006).
Email addresses: dongql@lsec.cc.ac.cn (Qiao-Li Dong), songnianhe@163.com (Songnian He), zhaoym@lsec.cc.ac.cn (Yanmin Zhao)

where P_C and P_Q denote the orthogonal projections onto C and Q, respectively. Generally, the projections of a point onto the sets C and Q are difficult to implement, while sometimes in practice, projections onto individual sets C_i and Q_j are easily calculated. For this purpose, Censor et al [6] defined a proximity function $p(x)$ to measure the distance of a point to all sets

$$p(x) = \frac{1}{2}\sum_{i=1}^{t}\alpha_i\|x - P_{C_i}(x)\|^2 + \frac{1}{2}\sum_{j=1}^{r}\beta_j\|Ax - P_{Q_j}A(x)\|^2, \tag{2}$$

where $\alpha_i > 0$ and $\beta_j > 0$ for all i and j, respectively, and $\sum_{i=1}^{t}\alpha_i + \sum_{j=1}^{r}\beta_j = 1$. We see that

$$\nabla p(x) = \sum_{i=1}^{t}\alpha_i(x - P_{C_i}(x)) + \sum_{j=1}^{r}\beta_j A^T(I - P_{Q_j})Ax.$$

Then, Censor et al [6] considered the following constrained MSFP:

$$\text{find} \quad x^* \in \Omega \quad \text{such that} \quad x^* \quad \text{solves the MSFP}, \tag{3}$$

where $\Omega \subseteq \mathbb{R}^N$ is an auxiliary simple nonempty closed convex set containing at least one solution of the MSFP. For solving the constrained MSFP (3), Censor et al [6] proposed a projection algorithm as follows

$$x_{n+1} = P_\Omega(x_n - s\nabla p(x_n)), \tag{4}$$

where s is a positive number such that $0 < s_L \le s \le s_U < 2/L(p)$, s_L and s_U are two fixed constants, and $L(p)$ is the Lipschitz constant of ∇p.

Observe that in the algorithm (4), the determination of the stepsize s depends on the operator (matrix) norm $\|A\|$ (or the largest eigenvalue of A^TA). This means that in order to implement the algorithm (4), one has first to compute (or, at least, estimate) operator norm of A, which is in general not an easy work in practice.

To overcome this difficulty, Zhang et al [16], Zhao and Yang [18, 19] proposed self-adaptive methods where the stepsize has no connection with matrix norms. Their methods actually compute the stepsize by adopting self-adaptive strategies. Recently, Dong and He [8] presented another self-adaptive method by using the backtracking rule.

Note that the algorithms proposed by Censor et al [6], Zhang et al [16] and Zhao and Yang [18, 19] involve the projection to an auxiliary set Ω. In fact, the set Ω is introduced just for the convenience of the proof of the convergence and it may be difficult to determine Ω in some cases. Considering this, Zhao and Yang [20], and Dong and He [8] presented projection algorithms which don't need projection to an auxiliary set Ω.

Algorithms in proposed [8] are proved to have sublinear global rate of convergence $O(1/n)$ and converge quite slowly, especially for large-scale problems (see [2] for details), although they have advantages stated above. Recently, Nesterov [10], Beck and Teboulle [2], and Tseng [13] studied the fast algorithms with global convergence rate $O(1/n^2)$, based on the proximal gradient algorithms for convex optimization. Inspired by them, in this paper, we introduce two algorithms which accelerate the two projection algorithms proposed by Dong and He [8] respectively and share global rate of convergence $O(1/n^2)$. Two algorithms don't need an auxiliary set Ω and the second algorithm is self-adaptive, which uses backtracking rule to get the stepsize. The efficiency of two algorithms is illustrated by some numerical experiments.

2. Preliminaries

In this section, we review some definitions and lemmas which will be used in the main results.

The following lemma is not hard to prove (see [1, 6]).

Lemma 2.1. *Let p be given as in (2). Then*

(i) p is convex and continuously differential;

(ii) $\nabla p(x)$ is Lipschitz continuous with $L(p) = \sum_{i=1}^{t} \alpha_i + \rho(A^TA) \sum_{j=1}^{r} \beta_j$ as the Lipschitz constant, where $\rho(A^TA)$ is the spectral radius of the matrix A^TA.

For any $\tau > 0$, consider the following quadratic approximation of $p(x)$ at a given point y:

$$R_\tau(x, y) := p(y) + \langle x - y, \nabla p(y)\rangle + \frac{\tau}{2}\|x - y\|^2,$$

which admits a unique minimizer

$$F_\tau(y) := \arg\min\{R_\tau(x, y) : \ x \in \mathbb{R}^n\}. \tag{5}$$

Simple algebra shows that (ignoring constant terms in y)

$$\begin{aligned} F_\tau(y) &= \arg\min_x \left\{ \frac{\tau}{2} \left\| x - \left(y - \frac{1}{\tau}\nabla p(y) \right) \right\|^2 \right\} \\ &= y - \frac{1}{\tau}\nabla p(y). \end{aligned} \tag{6}$$

The following lemma is well-known and fundamental property for a smooth function in the class $C^{1,1}$; e.g., [3, 11].

Lemma 2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable function with Lipschitz continuous gradient and Lipschitz constant $L(f)$. Then, for any $L > L(f)$,*

$$f(x) \le f(y) + \langle x - y, \nabla f(y)\rangle + \frac{L}{2}\|x - y\|^2, \quad \text{for every } x, y \in \mathbb{R}^n.$$

We are now ready to state and prove the promised key result.

Lemma 2.3. *(see [2]) Let $y \in \mathbb{R}^n$ and $\tau > 0$ be such that*

$$p(F_\tau(y)) \le R_\tau(F_\tau(y), y). \tag{7}$$

Then for any $x \in \mathbb{R}^n$,

$$p(x) - p(F_\tau(y)) \ge \frac{\tau}{2}\|F_\tau(y) - y\|^2 + \tau\langle y - x, F_\tau(y) - y\rangle.$$

Proof. See the appendix.

Remark 2.4. *Note that from lemmas 2.1 and 2.2, it follows that if $\tau \ge L(p)$, then the condition (7) is always satisfied for $F_\tau(y)$.*

3. Two Fast Projection Algorithms

In [8], Dong and He proposed two projection algorithms as follows:

Algorithm 3.1. *Let $L_1 \ge L(p)$ be a fixed constant and given $\tau_n \in (L(p), L_1)$. Let x_0 be arbitrary. For $n = 1, 2, \ldots$, compute*

$$x_{n+1} = x_n - \frac{1}{\tau_n}\nabla p(x_n). \tag{8}$$

Algorithm 3.2. *Given $\gamma > 0$ and $\eta > 1$. Let x_0 be arbitrary. For $n = 1, 2, \ldots$, find the smallest nonnegative integer m_n such that $\tau_n = \gamma\eta^{m_n}$ and*

$$x_{n+1} = x_n - \frac{1}{\tau_n}\nabla p(x_n), \tag{9}$$

which satisfies

$$p(x_{n+1}) - p(x_n) + \langle \nabla p(x_n), x_n - x_{n+1}\rangle \leq \frac{\tau_n}{2}\|x_n - x_{n+1}\|^2. \tag{10}$$

Dong and He [8] presented the following theorem which shows the convergence of the sequences $\{x_n\}$ and illustrates that two algorithms share a sublinear global rate of convergence $O(1/n)$.

Theorem 3.1. *Let $\{x_n\}$ be a sequence generated by Algorithm 3.1 or Algorithm 3.2. Then $\{x_n\}$ converges to a solution of the MSFP (1), and furthermore for any $n \geq 1$ it holds,*

$$p(x_n) \leq \frac{\alpha L(p)\|x_0 - x^*\|^2}{2n}, \quad \forall x^* \in \Gamma.$$

In this section, we introduce two algorithms which accelerate the algorithms 3.1 and 3.2, respectively. The global rate of convergence of the two algorithms are investigated and the sequence $\{p(x_n)\}$ has the better complexity rate $O(1/n^2)$.

Algorithm 3.3. *Let $L_1 \geq L(p)$ be a fixed constant and given $\tau_n \in (L(p), L_1)$. Let x_0 be arbitrary and set $y_1 = x_0, t_1 = 1$. For $n = 1, 2, \ldots$, compute*

$$x_n = y_n - \frac{1}{\tau_n}\nabla p(y_n), \tag{11}$$

$$t_{n+1} = \frac{1 + \sqrt{1 + 4t_n^2}}{2}, \tag{12}$$

and

$$y_{n+1} = x_n + \left(\frac{t_n - 1}{t_{n+1}}\right)(x_n - x_{n-1}). \tag{13}$$

Algorithm 3.4. *Given $\gamma > 0$ and $\eta > 1$. Let x_0 be arbitrary and set $y_1 = x_0, t_1 = 1$. For $n = 1, 2, \ldots$, find the smallest nonnegative integer m_n such that $\tau_n = \gamma\eta^{m_n}$ and*

$$x_n = y_n - \frac{1}{\tau_n}\nabla p(y_n), \tag{14}$$

which satisfies

$$p(x_n) - p(y_n) + \langle \nabla p(y_n), y_n - x_n\rangle \leq \frac{\tau_n}{2}\|y_n - x_n\|^2. \tag{15}$$

Compute

$$t_{n+1} = \frac{1 + \sqrt{1 + 4t_n^2}}{2}, \tag{16}$$

and

$$y_{n+1} = x_n + \left(\frac{t_n - 1}{t_{n+1}}\right)(x_n - x_{n-1}).$$

Lemma 3.2.

$$\beta L(p) \leq \tau_n \leq \alpha L(p). \tag{17}$$

where $\alpha = \frac{L_1}{L(p)}, \beta = 1$ *in Algorithm 3.3 and* $\alpha = \eta, \beta = \frac{\gamma}{L(p)}$ *in Algorithm 3.4.*

Proof. It is easy to verify (17) for Algorithm 3.3. By $\eta > 1$ and the choice of τ_n, we get $\tau_n \geq \gamma$. From lemma 2.2, it follows that inequality (15) is satisfied for $\tau_n \geq L(p)$, where $L(p)$ is the Lipschitz constant of ∇p. So, for Algorithm 3.4 one has $\tau_n \leq \eta L(p)$ for every $n \geq 1$.

Remark 3.3. *In the algorithm 'FISTA with backtracking', Beck and Teboulle [2] took* $\tau_n = \tau_{n-1}\eta^{m_n}$, *with* $\tau_0 > 0, \eta > 1$. *It is obvious that* τ_n *increases with n and consequently doesn't satisfy the right inequality of (17), which is essential in the proof of the convergence theorem. In Algorithm 3.4, we take* $\tau_n = \gamma\eta^{m_n}$ *which satisfies the right inequality of (17) (see the proof of Lemma 3.2).*

The next result provides the key recursive relation for the sequence $\{p(x_n)\}$.

Lemma 3.4. *(see [2]) The sequences* $\{x_n, y_n\}$ *generated via Algorithm 3.3 or Algorithm 3.4 satisfy for every* $n \geq 1$

$$\frac{2}{\tau_n}t_n^2 v_n - \frac{2}{\tau_{n+1}}t_{n+1}^2 v_{n+1} \geq \|u_{n+1}\|^2 - \|u_n\|^2,$$

where $v_n := p(x_n)$, $u_n := t_n x_n - (t_n - 1)x_{n-1} - x^*$ *with* $x^* \in \Gamma$.

Proof. See Appendix.

We also need the following trivial facts.

Lemma 3.5. *Let* $\{a_n, b_n\}$ *be positive sequences of reals satisfying*

$$a_n - a_{n+1} \geq b_{n+1} - b_n, \quad \forall n \geq 1, \text{ with } a_1 + b_1 \leq c,\ c > 0.$$

Then $a_n \leq c$ *for every* $n \geq 1$.

Lemma 3.6. *The positive sequence* $\{t_n\}$ *generated in (12) with* $t_1 = 1$ *satisfies* $t_n \geq (n+1)/2$ *for all* $n \geq 1$.

Theorem 3.7. *Let* $\{x_n\}$ *be generated by Algorithm 3.3 or Algorithm 3.4. Then for any* $n \geq 1$

$$p(x_n) \leq \frac{2\alpha L(p)\|x_0 - x^*\|^2}{(n+1)^2}, \quad \forall x^* \in \Gamma. \tag{18}$$

Proof. Let us define the quantities

$$a_n := \frac{2}{\tau_n}t_n^2 v_n, \quad b_n := \|u_n\|^2, \quad c := \|y_1 - x^*\|^2 = \|x_0 - x^*\|^2,$$

and recall (cf. Lemma 3.4) that $v_n := p(x_n)$. Then, by Lemma 3.4 we have for every $n \geq 1$

$$a_n - a_{n+1} \geq b_{n+1} - b_n,$$

and hence assuming that $a_1 + b_1 \leq c$ holds true, invoking Lemma 3.5, we obtain that

$$\frac{2}{\tau_n}t_n^2 v_n \leq \|x_0 - x^*\|^2,$$

which combined with $t_n \geq (n+1)/2$ (by Lemma 3.6) yields

$$v_n \leq \frac{2\tau_n\|x_0 - x^*\|^2}{(n+1)^2}.$$

Utilizing the upper bound on τ_n given in (17), the desired result (18) follows. Thus, all that remains is to prove the validity of the relation $a_1 + b_1 \leq c$. Since $t_1 = 1$, and using the definition of u_n given in Lemma 3.4, we have here

$$a_1 = \frac{2}{\tau_1} t_1^2 v_1 = \frac{2}{\tau_1} v_1, \quad b_1 = \|u_1\|^2 = \|x_1 - x^*\|^2.$$

Applying Lemma 2.3 to the points $x := x^*, y := y_1$ with $\tau = \tau_1$, we get

$$p(x^*) - p(x_1) \geq \frac{\tau_1}{2}\|x_1 - y_1\|^2 + \tau_1\langle y_1 - x^*, x_1 - y_1\rangle. \tag{19}$$

Thus, using $p(x^*) = 0$, we obtain

$$\begin{aligned} -p(x_1) &\geq \frac{\tau_1}{2}\|x_1 - y_1\|^2 + \tau_1\langle y_1 - x^*, x_1 - y_1\rangle \\ &= \frac{\tau_1}{2}\{\|x_1 - x^*\|^2 - \|y_1 - x^*\|^2\}. \end{aligned}$$

Consequently,

$$\frac{2}{\tau_1} v_1 \leq \|y_1 - x^*\|^2 - \|x_1 - x^*\|^2,$$

that is, $a_1 + b_1 \leq c$ holds true.

Remark 3.8. *In Note 2 of the article [13], Tseng discussed the choice of t_n in fast algorithms and concluded that t_n needs to satisfy the inequality*

$$t_{n+1}^2 - t_{n+1} \leq t_n^2. \tag{20}$$

The condition (20) allows $\{t_n\}$ to increase, but not too fast. For fastest convergence, $\{t_n\}$ should increase as fast as possible, as the proof of Theorem 3.7 suggests. It is easy to verify that the choice

$$t_n = \frac{n+1}{2}$$

also satisfies (20). Solving (20) with " $\leq$ " replaced by "=" yields (12) which tends to infinity somewhat faster.

Remark 3.9. *Different from Theorem 3.1, there isn't convergence of the sequence $\{x_n\}$ in Theorem 3.7 for algorithms 3.3 and 3.4. Combettes and Pesquet [7] concluded that the convergence of the sequence $\{x_n\}$ generated by Algorithm 3.3 or Algorithm 3.4 is no longer guaranteed in general.*

Table 1: Computational results for example 1 with different algorithms

	Algorithm 3.1			Algorithm 3.3			Algorithm 3.4	
	1.01L(p)	1.1L(p)	1.2L(p)	1.01L(p)	1.1L(p)	1.2L(p)		
Initial point	Iter.	Iter.	Iter.	Iter.	Iter.	Iter.	Iter.	InIt.
(0,0,0,0,0)	96	104	114	52	57	62	2	10
(20,10,20,10,20)	1246	1358	1482	629	685	747	8	24
(100,0,0,0,0)	1256	1368	1493	634	690	753	10	31
(1,1,1,1,1)	1228	1338	1460	621	676	737	3	16

Table 2: Computational results for example 2 with different dimensions and different numbers of C_i and Q_j.

	N		20	30	40	50	60
$t = 5$,	Algorithm 3.1	Iter.	482	551	747	889	1085
		Sec.	0.032	0.047	0.078	0.188	0.265
	Algorithm 3.3	Iter.	88	95	112	123	137
		Sec.	0.015	0.016	0.016	0.016	0.016
$r = 5$	Algorithm 3.2	Iter.	9	7	6	7	6
		InIt.	105	145	177	209	210
		Sec.	0.016	0.016	0.032	0.047	0.047
	Algorithm 3.4	Iter.	6	5	5	5	5
		InIt.	89	122	147	167	171
		Sec.	0.015	0.016	0.016	0.031	0.047
$t = 10$,	Algorithm 3.1	Iter.	599	794	872	1327	1713
		Sec.	0.109	0.188	0.329	0.609	1.187
	Algorithm 3.3	Iter.	135	158	166	210	241
		Sec.	0.031	0.031	0.046	0.094	0.172
$r = 15$	Algorithm 3.2	Iter.	11	8	6	7	7
		InIt.	126	158	177	213	244
		Sec.	0.032	0.047	0.062	0.110	0.172
	Algorithm 3.4	Iter.	5	5	6	5	5
		InIt.	105	139	147	179	202
		Sec.	0.031	0.031	0.047	0.125	0.125
$t = 30$,	Algorithm 3.1	Iter.	953	1250	2100	2246	2448
		Sec.	0.453	0.703	1.500	1.875	4.125
	Algorithm 3.3	Iter.	282	331	445	461	484
		Sec.	0.125	0.188	0.313	0.407	0.843
$r = 40$	Algorithm 3.2	Iter.	15	11	12	13	9
		InIt.	126	145	182	224	232
		Sec.	0.062	0.094	0.125	0.203	0.391
	Algorithm 3.4	Iter.	9	5	7	10	9
		InIt.	105	123	147	181	188
		Sec.	0.047	0.079	0.109	0.172	0.344

4. Numerical experiments

In order to verify the theoretical assertions, we present some numerical experiments in this section. We apply algorithms 3.3 and 3.4 to solve two test problems of [16] (examples 1 and 2), and compare the numerical results with those of the algorithms 3.1 and 3.2 proposed by Dong and He [8].

For convenience, we denote the vector with all elements 0 by e_0, and the vector with all elements 1 by e_1 in what follows. In the numerical results listed in the following tables, 'Iter.' and 'Sec.' denote the number of iterations and the cpu time in seconds, respectively. For algorithms 3.2 and 3.4, 'InIt.' denotes the number of total iterations of finding suitable τ_n in (10) and (15).

Example 1 ([16]). Consider the SFP as finding $x \in C = \{x \in \mathbb{R}^5 | \ \|x\| \leq 0.25\}$ such that $Ax \in Q = \{y = (y_1, y_2, y_3, y_4)^T \in \mathbb{R}^4 \,|\, 0.6 \leq y_j \leq 1,\ j = 1, 2, 3, 4\}$, where

$$A = \begin{pmatrix} 2 & -1 & 3 & 2 & 3 \\ 1 & 2 & 5 & 2 & 1 \\ 2 & 0 & 2 & 1 & -2 \\ 2 & -1 & 0 & -3 & 5 \end{pmatrix}.$$

The weights of $p(x)$ were set to $\alpha = 0.9$, $\beta = 0.1$. In the implementation, we took $p(x) < \varepsilon = 10^{-9}$ as the stopping criterion as in [16].

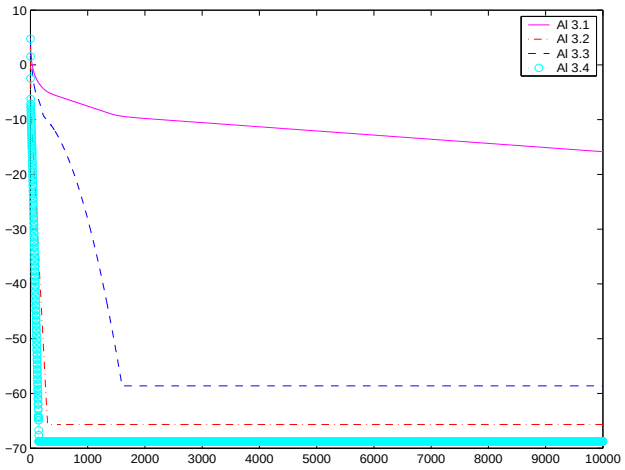

Figure 1: Comparison of the value errors $log(p(x_n))$ of algorithms 3.1, 3.2, 3.3 and 3.4.

For algorithms 3.1 and 3.3, we tested $\tau_n = 1.01L(p), 1.1L(p), \ldots, 1.9L(p)$ and the numerical results were reported in table 1 with different initial point x_0. (Since the number of iterations for $\tau_n = 1.3L(p), 1.4L(p), \ldots, 1.9L(p)$ was larger than those for $\tau_n \leq 1.2L(p)$, we only reported the results for $\tau_n \leq 1.2L(p)$.) We took $\gamma = 2$ and $\eta = 1.2$ for algorithms 3.2 and 3.4. We can see from table 1 that algorithms 3.1 and 3.3 were efficient when choosing a *suitable* τ_n ($\tau_n \in (L(p), 1.1L(p))$ was the best choice for the current example), while the number of iterations of Algorithm 3.3 was smaller than those of Algorithm 3.1 which is consistent with our theoretical analysis, and the number of iterations of Algorithm 3.4 was smaller than those for algorithms 3.1 and 3.3.

Example 2 ([16]). Consider the MSFP where $A = (a_{ij})_{N\times N} \in \mathbb{R}^{N\times N}$ and $a_{ij} \in (0, 1)$ generated randomly:

$$C_i = \{x \in \mathbb{R}^N \,|\, \|x - d_i\| \leq r_i\}, \quad i = 1, 2, \ldots, t,$$
$$Q_j = \{y \in \mathbb{R}^N \,|\, L_j \leq y \leq U_j\}, \quad j = 1, 2, \ldots, r,$$

where d_i is the center of the ball C_i, $e_0 \leq d_i \leq 10e_1$, and $r_i \in (40, 50)$ is the radius, d_i and r_i are both generated randomly. L_j and U_j are the boundary of the box Q_j, and are also generated randomly, satisfying $20e_1 \leq L_j \leq 30e_1$, $40e_1 \leq U_j \leq 80e_1$, respectively. The weights of $p(x)$ were $1/(t + r)$. The stopping criterion was $p(x) < \varepsilon = 10^{-4}$ with the initial point $x_0 = e_0 \in \mathbb{R}^N$.

We tested the algorithms 3.1, 3.2, 3.3 and 3.4 with different t and r in different dimensional Euclidean space. In algorithms 3.1 and 3.3, since a smaller τ_n was more efficient than a larger one, we chose $\tau_n = 1.01L(p)$ in the experiment. We took $\gamma = 1$, $\eta = 1.1$ for algorithms 3.2 and 3.4. For comparison, the same random values were taken in each test for four algorithms. The numerical results were listed in table 2, from which we could observe the efficiency of the algorithms 3.3 and 3.4, both from the points of view of number of iterations and cpu time. We found that the algorithms 3.3 and 3.4, in fact, accelerated the algorithms 3.1 and 3.2, respectively. It should be noted that the self-adaptive algorithm 3.2 behaved better that the fast algorithm 3.3 and the reason is worth further research.

The logarithm of $p(x_n)$ of the four algorithms for 10000 iterations was described in Figure 1. One could see that after 10000 iterations algorithms 3.1, 3.2, 3.3 and 3.4 reached an accuracy of approximately 10^{-7}, 10^{-29}, 10^{-26} and 10^{-30}, respectively. The results produced by algorithms 3.3 (resp. 3.4) were better than those produced by algorithms 3.1 (resp. 3.2) by several (resp. one) orders of magnitude, and it clearly demonstrated the effective performance of two fast algorithms. As the above analysis, the efficiency of

Algorithm 3.2 was better than that of Algorithm 3.3.

Acknowledgements The authors would like to thank Dr. Wenxing Zhang for his help in numerical experiments and the reviewers for their pertinent comments and suggestions.

Appendix

Proof of Lemma 2.3. From (7), we have

$$p(x) - p(F_\tau(y)) \geq p(x) - R_\tau(F_\tau(y), y). \tag{21}$$

Now, from the fact that p is convex, it follows

$$p(x) \geq p(y) + \langle x - y, \nabla p(y)\rangle. \tag{22}$$

On the other hand, by the definition of $R_\tau(x, y)$, one has

$$R_\tau(F_\tau(y), y) = p(y) + \langle F_\tau(y) - y, \nabla p(y)\rangle + \frac{\tau}{2}\|F_\tau(y) - y\|^2. \tag{23}$$

Therefore, using (21)-(23), it follows that

$$\begin{aligned} p(x) - p(F_\tau(y)) &\geq -\frac{\tau}{2}\|F_\tau(y) - y\|^2 + \langle x - F_\tau(y), \nabla p(y)\rangle \\ &= -\frac{\tau}{2}\|F_\tau(y) - y\|^2 + \tau\langle x - F_\tau(y), y - F_\tau(y)\rangle \\ &= \frac{\tau}{2}\|F_\tau(y) - y\|^2 + \tau\langle y - x, F_\tau(y) - y\rangle, \end{aligned}$$

where in the first equality above we used (6).

Proof of Lemma 3.4. First we apply Lemma 2.3 at the points $(x := x_n, y := y_{n+1})$ with $\tau = \tau_{n+1}$, and likewise at the points $(x := x^*, y := y_{n+1})$, to get

$$\begin{aligned} 2\tau_{n+1}^{-1}(v_n - v_{n+1}) &\geq \|x_{n+1} - y_{n+1}\|^2 + 2\langle x_{n+1} - y_{n+1}, y_{n+1} - x_n\rangle, \\ -2\tau_{n+1}^{-1}v_{n+1} &\geq \|x_{n+1} - y_{n+1}\|^2 + 2\langle x_{n+1} - y_{n+1}, y_{n+1} - x^*\rangle, \end{aligned}$$

where we used the fact that $p(x^*) = 0$ and $x_{n+1} = F_{\tau_{n+1}}(y_{n+1})$. To get a relation between v_n and v_{n+1}, we multiply the first inequality above by $(t_{n+1} - 1)$ and add it to the second inequality:

$$\frac{2}{\tau_{n+1}}((t_{n+1} - 1)v_n - t_{n+1}v_{n+1}) \geq t_{n+1}\|x_{n+1} - y_{n+1}\|^2 + 2\langle x_{n+1} - y_{n+1}, t_{n+1}y_{n+1} - (t_{n+1} - 1)x_n - x^*\rangle.$$

Multiplying the last inequality by t_{n+1} and using the relation $t_n^2 = t_{n+1}^2 - t_{n+1}$ which holds thanks to (12), we obtain

$$\frac{2}{\tau_{n+1}}(t_n^2v_n - t_{n+1}^2v_{n+1}) \geq \|t_{n+1}(x_{n+1} - y_{n+1})\|^2 + 2t_{n+1}\langle x_{n+1} - y_{n+1}, t_{n+1}y_{n+1} - (t_{n+1} - 1)x_n - x^*\rangle.$$

Applying the usual Pythagoras relation

$$\|b - a\|^2 + 2\langle b - a, a - c\rangle = \|b - c\|^2 - \|a - c\|^2,$$

to the right-hand side of the last inequality with

$$a := t_{n+1}y_{n+1}, \quad b := t_{n+1}x_{n+1}, \quad c := (t_{n+1} - 1)x_n + x^*,$$

we thus get

$$\frac{2}{\tau_{n+1}}(t_n^2 v_n - t_{n+1}^2 v_{n+1}) \geq \|t_{n+1}x_{n+1} - (t_{n+1}-1)x_n - x^*\|^2 - \|t_{n+1}y_{n+1} - (t_{n+1}-1)x_n - x^*\|^2.$$

Therefore, with y_{n+1} (cf. (13)) and u_n defined by

$$t_{n+1}y_{n+1} = t_{n+1}x_n + (t_n - 1)(x_n - x_{n-1}), \quad \text{and} \quad u_n = t_n x_n - (t_n - 1)x_{n-1} - x^*,$$

it follows that

$$\frac{2}{\tau_{n+1}}(t_n^2 v_n - t_{n+1}^2 v_{n+1}) \geq \|u_{n+1}\|^2 - \|u_n\|^2,$$

which combined with the inequality $\tau_{n+1} \geq \tau_n$ yields

$$\frac{2}{\tau_n}t_n^2 v_n - \frac{2}{\tau_{n+1}}t_{n+1}^2 v_{n+1} \geq \|u_{n+1}\|^2 - \|u_n\|^2.$$

The proof is completed.

Acknowledgements The authors would like to express their thanks the referees for their helpful suggestions.

References

[1] J. P. Aubin, Optima and Equilibria: An Introduction to Nonlinear Analysis (Berlin: Springer), 1993.
[2] A. Beck, M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sciences 2 (2000) 183–202.
[3] D. P. Bertsekas, Nonlinear Programming, 2nd ed., Athena Scientific, Belmont, MA, 1999.
[4] C. L. Byrne, A unified treatment of some iterative algorithms in signal processing and image reconstruction, Inverse Probl. 20 (2004) 103-120.
[5] Y. Censor, T. Elfving, A multiprojection algorithm using Bregman projections in a product space, Numer. Algorithms 8 (1994) 221–239.
[6] Y. Censor, T. Elfving, N. Kopf, T. Bortfeld, The multiple-sets split feasibility problem and its applications for inverse problems, Inverse Probl. 21 (2005) 2071–2084.
[7] P. L. Combettes, J. C. Pesquet, Proximal splitting methods in signal processing, in: Fixed-Point Algorithms for Inverse Problems in Science and Engineering, (H. H. Bauschke, R. S. Burachik, P. L. Combettes, V. Elser, D. R. Luke, and H. Wolkowicz, Editors), pp. 185–212. Springer, New York, 2011.
[8] Q. L. Dong, S. He, Two projection algorithms for the multiple-sets split feasibility problem, J. Appl. Math. Volume 2013, Article ID 347401, 5 pages.
[9] Z. Li, D. Han, W. Zhang, A self-adaptive projection-type method for nonlinear multiple-sets split feasibility problem, Inverse Probl. Sci. Eng. 21 (2013) 155–170.
[10] Y. Nesterov, Introductory Lectures on Convex Optimization: A Basic Course, Kluwer, 2003.
[11] J. M. Ortega, W. C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Classics Appl. Math. 30, SIAM, Philadelphia, 2000.
[12] R. J. Palta, T. R. Mackie, Intensity-modulated radiation therapy: the state of art (medical physics monograph 29) (madison, WI: medical physics publishing), 2003.
[13] P. Tseng, On accelerated proximal gradient methods for convex-concave optimization, (2008).
[14] H. K. Xu, A variable Krasnosel'skiĭ-Mann algorithm and the multiple-set split feasibility problem, Inverse Probl. 22 (2006) 2021–2034.
[15] Y. Yao, R. Chen, G. Marino, Y. C. Liou, Applications of fixed-point and optimization methods to the multiple-set split feasibility problem, J. Appl. Math. Vol. 2012 Article ID 927530, 21 pages.
[16] W. Zhang, D. Han, Z. Li, A self-adaptive projection method for solving the multiple-sets split feasibility problem, Inverse Probl. 25 (2009) 115001., 16pp.
[17] W. Zhang, D. Han, X. Yuan, An efficient simultaneous method for the constrained multiple-sets split feasibility problem, Comput. Optim. Appl. 52 (2012) 825–843.
[18] J. Zhao, Q. Yang, Self-adaptive projection methods for the multiple-sets split feasibility problem, Inverse Probl. 27 (2011) 035009.
[19] J. Zhao, Q. Yang, Several acceleration schemes for solving the multiple-sets split feasibility problem, Linear Algebra Appl. 437 (2012) 1648–1657.
[20] J. Zhao, J. Zhang, Q. Yang, A simple projection method for solving the multiple-sets split feasibility problem, Inverse Probl. Sci. Eng. 21 (2013) 537–546.