Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 347401, 5 pages
http://dx.doi.org/10.1155/2013/347401

Research Article

On Two Projection Algorithms for the Multiple-Sets Split Feasibility Problem

Qiao-Li Dong and Songnian He

College of Science, Civil Aviation University of China, Tianjin 300300, China

Correspondence should be addressed to Qiao-Li Dong; dongql@lsec.cc.ac.cn

Received 15 July 2013; Accepted 27 November 2013

Academic Editor: Livija Cveticanin

We present a projection algorithm which modifies the method proposed by Censor and Elfving (1994) and also introduce a self-adaptive algorithm for the multiple-sets split feasibility problem (MSFP). The global rates of convergence are firstly investigated and the sequences generated by two algorithms are proved to converge to a solution of the MSFP. The efficiency of the proposed algorithms is illustrated by some numerical tests.

1. Introduction

The multiple-sets split feasibility problem (MSFP) is to find x^* satisfying

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

where A is an $M \times N$ real matrix, $C_i \subseteq \mathbb{R}^N$, $i = 1, \dots, t$, and $Q_j \subseteq \mathbb{R}^M$, $j = 1, \dots, r$, are the nonempty closed convex sets. This problem was firstly proposed by Censor et al. in [1] and can be a model for many inverse problems where constraints are imposed on the solutions in the domain of a linear operator as well as in the operator's range. Many researchers studied the MSFP and introduced various algorithms to solve it (see [1–7] and the references therein). If $t = r = 1$, then this problem reduces to the feasible case of the split feasibility problem (see, e.g., [8–11]), which is to find $x^* \in C$ with $Ax^* \in Q$.

Assume that the MSFP (1) is consistent; that is, its solution set, denoted by Γ, is nonempty. For convenience reasons, Censor et al. [1] considered the following constrained MSFP:

$$\text{find } x^* \in \Omega \quad \text{such that } x^* \text{ solves the MSFP}, \tag{2}$$

where $\Omega \subseteq \mathscr{H}_1$ is an auxiliary simple nonempty closed convex set containing at least one solution of the MSFP. For solving the constrained MSFP, Censor et al. [1] defined a proximity function $p(x)$ to measure the distance of a point to all sets

$$\begin{aligned} p(x) = {} & \frac{1}{2}\sum_{i=1}^{t}\alpha_i \left\| x - P_{C_i}(x)\right\|^2 \\ & + \frac{1}{2}\sum_{j=1}^{r}\beta_j \left\| Ax - P_{Q_j}A(x)\right\|^2, \end{aligned} \tag{3}$$

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

$$\begin{aligned} \nabla p(x) = {} & \sum_{i=1}^{t}\alpha_i \left(x - P_{C_i}(x)\right) \\ & + \sum_{j=1}^{r}\beta_j A^T \left(I - P_{Q_j}\right) Ax. \end{aligned} \tag{4}$$

Censor et al. [1] proposed a projection algorithm as follows:

$$x_{n+1} = P_\Omega\left(x_n - s\nabla p(x_n)\right), \tag{5}$$

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

Observe that in the algorithm (5) the determination of the stepsize s depends on the operator (matrix) norm $\|A\|$ (or the largest eigenvalue of $A * A$). This means that, in order to implement the algorithm (5), 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. [4] and Zhao and Yang [5, 7] proposed self-adaptive methods where the stepsize has no connection with matrix norms. Their methods actually compute the stepsize by adopting different self-adaptive strategies.

Note that the algorithms proposed by Censor et al. [1], Zhang et al. [4], and Zhao and Yang [5, 7] 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 [6] presented simple projection algorithms which does not need projection to an auxiliary set Ω.

In this paper, we introduce two projection algorithms for solving the MSFP, inspired by Beck and Teboulle's iterative shrinkage-thresholding algorithm for linear inverse problem [12]. The first algorithm modifies Censor et al.'s method which does not need projection to an auxiliary set Ω. The second algorithm is self-adaptive and adopts the backtracking rule to determine the stepsize. We firstly study the global rate of convergence of two algorithms and prove that the sequences generated by the proposed algorithms converge to a solution of the MSFP. Some numerical results are presented, which illustrate the efficiency of the proposed algorithms.

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, 13]).

Lemma 1. *Let p be given as in* (3). *Then*

(i) *p is convex and continuously differential,*

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

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

$$R_\tau(x, y) := p(y) + \langle x - y, \nabla p(y)\rangle + \frac{\tau}{2}\|x - y\|^2, \tag{6}$$

which admits a unique minimizer

$$F_\tau(y) := \operatorname{argmin}\left\{R_\tau(x, y) : x \in \mathbb{R}^N\right\}. \tag{7}$$

Simple algebra shows that (ignoring constant terms in y)

$$\begin{aligned} F_\tau(y) &= \operatorname*{argmin}_x \left\{\frac{\tau}{2}\left\|x - \left(y - \frac{1}{\tau}\nabla p(y)\right)\right\|^2\right\} \\ &= y - \frac{1}{\tau}\nabla p(y). \end{aligned} \tag{8}$$

The following lemma is well known and a fundamental property for a smooth function in the class $C^{1,1}$; for example, see [14, 15].

Lemma 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 $L > L(f)$,*

$$\begin{aligned} f(x) \le f(y) + \langle x - y, \nabla f(y)\rangle + \frac{L}{2}\|x - y\|^2, \\ \text{for every } x, y \in \mathbb{R}^n. \end{aligned} \tag{9}$$

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

Lemma 3 (see [12]). *Let $y \in \mathbb{R}^n$ and $\tau > 0$ be such that*

$$p(F_\tau(y)) \le R_\tau(F_\tau(y), y). \tag{10}$$

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

$$p(x) - p(F_\tau(y)) \ge \frac{\tau}{2}\|F_\tau(y) - y\|^2 + \tau\langle y - x, F_\tau(y) - y\rangle. \tag{11}$$

Proof. From (10), we have

$$p(x) - p(F_\tau(y)) \ge p(x) - R_\tau(F_\tau(y), y). \tag{12}$$

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

$$p(x) \ge p(y) + \langle x - y, \nabla p(y)\rangle. \tag{13}$$

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

$$\begin{aligned} R_\tau(F_\tau(y), y) = p(y) + \langle F_\tau(y) - y, \nabla p(y)\rangle \\ + \frac{\tau}{2}\|F_\tau(y) - y\|^2. \end{aligned} \tag{14}$$

Therefore, using (12)–(14), it follows that

$$\begin{aligned} p(x) - p(F_\tau(y)) &\ge -\frac{\tau}{2}\|F_\tau(y) - y\|^2 \\ &\quad + \langle x - F_\tau(y), \nabla p(y)\rangle \\ &= -\frac{\tau}{2}\|F_\tau(y) - y\|^2 \\ &\quad + \tau\langle x - F_\tau(y), y - F_\tau(y)\rangle \\ &= \frac{\tau}{2}\|F_\tau(y) - y\|^2 + \tau\langle y - x, F_\tau(y) - y\rangle, \end{aligned} \tag{15}$$

where in the first equality above we used (8). □

Remark 4. Note that, from Lemmas 1 and 2, it follows that if $\tau \ge L(p)$, then the condition (10) is always satisfied for $F_\tau(y)$.

3. Two Projection Algorithms

In this section, we propose two projection algorithms which do not need an auxiliary set Ω; one modifies the algorithm introduced by Censor et al. [1] and the other is a self-adaptive algorithm which solves the MSFP without prior knowledge of spectral radius of the matrix $A^T A$.

Algorithm 5. Let $L_1 \ge L(p)$ be a fixed constant and take $\tau_n \in (L(p), L_1)$. Let x_0 be arbitrary. For $n = 0, 1, 2, \ldots$, compute

$$x_{n+1} = x_n - \frac{1}{\tau_n}\nabla p(x_n). \tag{16}$$

Remark 6. Algorithm 5 is different from Censor et al.'s algorithm (5) in [1] and it does not need the projection to an auxiliary simple nonempty closed convex set Ω. In Algorithm 5, we take $\tau_n > L(p)$ instead of $\tau_n > L(p)/2$ (as in Censor et al.'s algorithm) which is restricted to a smaller range.

Algorithm 7. Given $\gamma > 0$ and $\eta > 1$, let x_0 be arbitrary. For $n = 0, 1, 2, \ldots$, find the smallest nonnegative integer m_n such that $\tau_n = \gamma\eta^{m_n}$ and

$$x_{n+1} = x_n - \frac{1}{\tau_n}\nabla p\left(x_n\right), \tag{17}$$

which satisfies

$$\begin{aligned} p\left(x_{n+1}\right) - p\left(x_n\right) + \left\langle \nabla p\left(x_n\right), x_n - x_{n+1}\right\rangle \\ \leq \frac{\tau_n}{2}\left\|x_n - x_{n+1}\right\|^2. \end{aligned} \tag{18}$$

Remark 8. Note that the sequence of function values $\{p(x_n)\}$ produced by Algorithms 5 and 7 is nonincreasing. Indeed, for every $n \geq 1$,

$$\begin{aligned} p\left(x_{n+1}\right) &\leq R_{\tau_n}\left(x_{n+1}, x_n\right) \\ &\leq R_{\tau_n}\left(x_n, x_n\right) = p\left(x_n\right), \end{aligned} \tag{19}$$

where the first inequality comes from Lemma 2 for Algorithm 5 and from (18) for Algorithm 7, and the second inequality follows from (7). τ_n in (19) is either chosen by the backtracking rule (18) or $\tau_n \in (L(p), L_1)$, where $L(p)$ is a given Lipschitz constant of ∇p.

Lemma 9. *There holds*

$$\beta L\left(p\right) \leq \tau_n \leq \alpha L\left(p\right), \tag{20}$$

where $\alpha = L_1/L(p)$, $\beta = 1$ in Algorithm 5 and $\alpha = \eta$, $\beta = \gamma/L(p)$ in Algorithm 7.

Proof. It is easy to verify (20) for Algorithm 5. By $\eta > 1$ and the choice of τ_n, we get $\tau_n \geq \gamma$. From Lemma 2, it follows that inequality (18) is satisfied for $\tau_n \geq L(p)$, where $L(p)$ is the Lipschitz constant of ∇p. So, for Algorithm 7 one has $\tau_n \leq \eta L(p)$ for every $n \geq 1$. □

Remark 10. From Lemma 9, it follows that backtracking rule (18) is well defined.

Remark 11. In algorithm "ISTA with backtracking" proposed by Beck and Teboulle [12], they took $\tau_n = \tau_{n-1}\eta^{m_n}$, with $\tau_0 > 0$ and $\eta > 1$. It is obvious that τ_n increases with n. It is verified that small τ_n is more efficient than a larger one in numerical experiments (see Table 1). So, in Algorithm 7, we take $\tau_n = \gamma\eta^{m_n}$ for backtracking rule which is smaller than the one in the algorithm of Beck and Teboulle.

Theorem 12. *Let $\{x_n\}$ be a sequence generated by Algorithm 5 or Algorithm 7. Then $\{x_n\}$ converges to a solution of the MSFP (1), and furthermore for any $n \geq 1$ it holds that*

$$p\left(x_n\right) \leq \frac{\alpha L\left(p\right)\left\|x_0 - x^*\right\|^2}{2n}, \quad \forall x^* \in \Gamma. \tag{21}$$

Proof. Invoking Lemma 3 with $x = x^*$, $y = x_k$, and $\tau = \tau_k$, we obtain

$$\begin{aligned} \frac{2}{\tau_k}\left(p\left(x^*\right) - p\left(x_{k+1}\right)\right) &\geq \left\|x_{k+1} - x_k\right\|^2 \\ &\quad + 2\left\langle x_k - x^*, x_{k+1} - x_k\right\rangle \\ &= \left\|x_{k+1} - x^*\right\|^2 - \left\|x_k - x^*\right\|^2, \end{aligned} \tag{22}$$

which combined with (20) and the fact that $p(x^*) = 0$, $p(x_{k+1}) \geq 0$ yields

$$-\frac{2}{\alpha L\left(p\right)}p\left(x_{k+1}\right) \geq \left\|x_{k+1} - x^*\right\|^2 - \left\|x_k - x^*\right\|^2, \tag{23}$$

which implies

$$\left\|x_{k+1} - x^*\right\| \leq \left\|x_k - x^*\right\|. \tag{24}$$

So $\{x_n\}$ is a Fejér monotone sequence. Summing the inequality (23) over $k = 0, 1, \ldots, n-1$ gives

$$-\frac{2}{\alpha L\left(p\right)}\sum_{k=0}^{n-1}p\left(x_{k+1}\right) \geq \left\|x_n - x^*\right\|^2 - \left\|x_0 - x^*\right\|^2. \tag{25}$$

Invoking Lemma 3 one more time with $x = y = x_k$ and $\tau = \tau_k$ yields

$$\frac{2}{\tau_k}\left(p\left(x_k\right) - p\left(x_{k+1}\right)\right) \geq \left\|x_k - x_{k+1}\right\|^2. \tag{26}$$

Since $\tau_k \geq \beta L(p)$ (see (20)) and $p(x_k) - p(x_{k+1}) \geq 0$ (see (19)), it follows that

$$\frac{2}{\beta L\left(p\right)}\left(p\left(x_k\right) - p\left(x_{k+1}\right)\right) \geq \left\|x_k - x_{k+1}\right\|^2. \tag{27}$$

Multiplying the last inequality by k and summing over $k = 0, \ldots, n-1$, we obtain

$$\begin{aligned} \frac{2}{\beta L\left(p\right)}\sum_{k=0}^{n-1}\left(kp\left(x_k\right) - \left(k+1\right)p\left(x_{k+1}\right) + p\left(x_{k+1}\right)\right) \\ \geq \sum_{k=0}^{n-1}k\left\|x_k - x_{k+1}\right\|^2, \end{aligned} \tag{28}$$

which simplifies to

$$\begin{aligned} \frac{2}{\beta L\left(p\right)}\left(-np\left(x_n\right) + \sum_{k=0}^{n-1}p\left(x_{k+1}\right)\right) \\ \geq \sum_{k=0}^{n-1}k\left\|x_k - x_{k+1}\right\|^2. \end{aligned} \tag{29}$$

Adding (25) and (29) times β/α, we get

$$\begin{aligned} -\frac{2n}{\alpha L\left(p\right)}p\left(x_n\right) \geq \left\|x_n - x^*\right\|^2 + \frac{\beta}{\alpha}\sum_{k=0}^{n-1}k\left\|x_k - x_{k+1}\right\|^2 \\ - \left\|x_0 - x^*\right\|^2, \end{aligned} \tag{30}$$

Table 1: Computational results for Example 13 with different algorithms.

Initial point	Algorithm 5 with different τ					Algorithm 7	
	$1.01L(p)$	$1.1L(p)$	$1.2L(p)$	$1.3L(p)$	$1.4L(p)$		
	Iter.	Iter.	Iter.	Iter.	Iter.	Iter.	InIt.
(0, 0, 0, 0, 0)	96	104	114	123	132	7	22
(20, 10, 20, 10, 20)	1246	1358	1482	1606	1730	35	77
(100, 0, 0, 0, 0)	1256	1368	1493	1618	1743	39	90
(1, 1, 1, 1, 1)	1228	1338	1460	1582	1704	28	54

Table 2: Computational results for Example 14 with different dimensions and different numbers of C_i and Q_j.

	N		20	30	40	50	60
$t = 5$,	Algorithm 5	Iter.	515	675	774	875	1098
		Sec.	0.093	0.125	0.156	0.203	0.485
		Iter.	11	8	7	7	7
$r = 5$	Algorithm 7	InIt.	71	94	105	104	133
		Sec.	0.016	0.031	0.047	0.062	0.078
$t = 10$,	Algorithm 5	Iter.	772	1412	1456	1583	1614
		Sec.	0.328	0.625	0.782	1.047	1.297
		Iter.	14	13	9	8	7
$r = 15$	Algorithm 7	InIt.	76	92	120	122	140
		Sec.	0.031	0.063	0.078	0.094	0.125
$t = 30$,	Algorithm 5	Iter.	854	1467	2100	2246	2448
		Sec.	0.406	0.828	1.437	1.875	3.516
		Iter.	15	13	13	13	9
$r = 40$	Algorithm 7	InIt.	78	88	113	123	144
		Sec.	0.032	0.047	0.093	0.188	0.297

and hence, it follows that

$$p(x_n) \le \frac{\alpha L(p)\|x_0 - x^*\|^2}{2n}, \quad \forall x^* \in \Gamma, \tag{31}$$

which yields

$$\lim_{n\to\infty} p(x_n) = 0. \tag{32}$$

Since $\{x_n\}$ is Fejér monotone, it is bounded. To prove the convergence of $\{x_n\}$, it only remains to show that all converging subsequences have the same limit. Suppose in contradiction that two subsequences $\{x_{n_k}\}$ and $\{x_{n_l}\}$ converge to different limits $\widehat{x}$ and $\widetilde{x}$, respectively ($\widehat{x} \neq \widetilde{x}$). We are to show that $\widehat{x}$ is a solution of the MSFP. The continuity of $p(x)$ then implies that

$$0 \le p(\widehat{x}) = \lim_{k\to\infty} p(x_{n_k}) = \lim_{n\to\infty} p(x_n) = 0. \tag{33}$$

Therefore, $p(\widehat{x}) = 0$; that is, $\widehat{x} \in C = \bigcap_{i=1}^{t} C_i$ and $A\widehat{x} \in Q = \bigcap_{j=1}^{r} Q_j$; that is, $\widehat{x}$ is a solution of the MSFP. Similarly, we can show that it is a solution of the MSFP. Now, by Fejér monotonicity of the sequence $\{x_n\}$, it follows that the sequence $\{\|x_n - \widehat{x}\|\}$ is bounded and nonincreasing and thus has a limit $\lim_{n\to\infty}\|x_n - \widehat{x}\| = l_1$. However, we also have $\lim_{n\to\infty}\|x_n - \widehat{x}\| = \lim_{k\to\infty}\|x_{n_k} - \widehat{x}\| = 0$, and $\lim_{n\to\infty}\|x_n - \widehat{x}\| = \lim_{l\to\infty}\|x_{n_l} - \widehat{x}\| = \|\widetilde{x} - \widehat{x}\|$, so that $l_1 = 0 = \|\widetilde{x} - \widehat{x}\|$, which is obviously a contradiction. Thus $\{x_n\}$ converges to a solution of the MSFP (1). The proof is completed. □

4. Numerical Tests

In order to verify the theoretical assertions, we present some numerical tests in this section. We apply Algorithms 5 and 7 to solve two test problems of [4] (Examples 13 and 14) and compare the numerical results of two algorithms.

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." denoted the number of iterations and the CPU time in seconds, respectively. For Algorithm 7, "InIt." denoted the number of total iterations of finding suitable τ_n in (18).

Example 13 (see [4]). Consider a split feasibility problem as finding $x \in C = \{x \in \mathbb{R}^5 \mid \|x\| \le 0.25\}$ such that $Ax \in Q = \{y = (y_1, y_2, y_3, y_4)^T \in \mathbb{R}^4 \mid 0.6 \le y_j \le 1, j = 1, 2, 3, 4\}$, where the matrix

$$A = \begin{pmatrix} 2 & -1 & 3 & 2 & 3 \\ 1 & 2 & 5 & 2 & 1 \\ 2 & 0 & 2 & 1 & -2 \\ 2 & -1 & 0 & -3 & 5 \end{pmatrix}. \tag{34}$$

The weights of $p(x)$ were set to $\alpha = 0.9$ and $\beta = 0.1$. In the implementation, we took $p(x) < \varepsilon = 10^{-9}$ as the stopping criterion as in [4].

For Algorithm 5, 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 points x_0. (Since the number of iterations for $\tau_n = 1.5L(p), 1.6L(p), \ldots, 1.9L(p)$ was larger than those for $\tau_n \le 1.4L(p)$, we only reported the results for $\tau_n \le 1.4L(p)$.) We took $\gamma = 1$ and $\eta = 1.1$ for Algorithm 7. Table 1 shows that Algorithm 5 was 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 5 was still larger than those for Algorithm 7.

Example 14 (see [4]). Consider the MSFP, where $A = (a_{ij})_{N\times N} \in \mathbb{R}^{N\times N}$ and $a_{ij} \in (0,1)$ generated randomly:

$$C_i = \left\{x \in \mathbb{R}^N \mid \|x - d_i\| \le r_i\right\}, \quad i = 1, 2, \ldots, t,$$
$$Q_j = \left\{y \in \mathbb{R}^N \mid L_j \le y \le U_j\right\}, \quad j = 1, 2, \ldots, r, \tag{35}$$

where $d_i \in \mathbb{R}^N$ is the center of the ball C_i, $e_0 \le d_i \le 10e_1$, 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 $20e_1 \le L_j \le 30e_1$, $40e_1 \le U_j \le 80e_1$, 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 Algorithms 5 and 7 with different t and r in different dimensional Euclidean space. In Algorithm 5, since a smaller τ_n is more efficient than a larger one, we take $\tau_n = 1.01L(p)$ in the experiment. We take $\gamma = 1$ and $\eta = 1.2$ for Algorithm 7. For comparison, the same random values were taken in each test. The numerical results were listed in Table 2, from which we can observe the efficiency of the self-adaptive Algorithm 7, both from the points of view of number of iterations and CPU time.

Conflicts of Interests

There is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors express their thanks to Dr. Wenxing Zhang for his help in numerical tests. This research was supported by the National Natural Science Foundation of China (no. 11201476) and Fundamental Research Funds for the Central Universities (no. 3122013D017).

References

[1] Y. Censor, T. Elfving, N. Kopf, and T. Bortfeld, "The multiple-sets split feasibility problem and its applications for inverse problems," *Inverse Problems*, vol. 21, no. 6, pp. 2071–2084, 2005.

[2] Z. Li, D. Han, and W. Zhang, "A self-adaptive projection-type method for nonlinear multiple-sets split feasibility problem," *Inverse Problems in Science and Engineering*, vol. 21, no. 1, pp. 155–170, 2013.

[3] H.-K. Xu, "A variable Krasnosel'skii-Mann algorithm and the multiple-set split feasibility problem," *Inverse Problems*, vol. 22, no. 6, pp. 2021–2034, 2006.

[4] W. Zhang, D. Han, and Z. Li, "A self-adaptive projection method for solving the multiple-sets split feasibility problem," *Inverse Problems*, vol. 25, no. 11, article 115001, 2009.

[5] J. Zhao and Q. Yang, "Self-adaptive projection methods for the multiple-sets split feasibility problem," *Inverse Problems*, vol. 27, no. 3, article 035009, 2011.

[6] J. Zhao and Q. Yang, "A simple projection method for solving the multiple-sets split feasibility problem," *Inverse Problems in Science and Engineering*, vol. 21, no. 3, pp. 537–546, 2013.

[7] J. Zhao and Q. Yang, "Several acceleration schemes for solving the multiple-sets split feasibility problem," *Linear Algebra and Its Applications*, vol. 437, no. 7, pp. 1648–1657, 2012.

[8] C. Byrne, "A unified treatment of some iterative algorithms in signal processing and image reconstruction," *Inverse Problems*, vol. 20, no. 1, pp. 103–120, 2004.

[9] Y. Censor and T. Elfving, "A multiprojection algorithm using Bregman projections in a product space," *Numerical Algorithms*, vol. 8, no. 2–4, pp. 221–239, 1994.

[10] Q.-L. Dong, Y. Yao, and S. He, "Weak convergence theorems of the modified relaxed projection algorithms for the split feasibility problem in Hilbert spaces," *Optimization Letters*, 2013.

[11] S. He and W. Zhu, "A note on approximating curve with 1-norm regularization method for the split feasibility problem," *Journal of Applied Mathematics*, vol. 2012, Article ID 683890, 10 pages, 2012.

[12] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[13] J.-P. Aubin, *Optima and Equilibria: An Introduction to Nonlinear Analysis*, vol. 140 of *Graduate Texts in Mathematics*, Springer, Berlin, Germany, 1993.

[14] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, Mass, USA, 2nd edition, 1999.

[15] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, vol. 30 of *Classics in Applied Mathematics*, SIAM, Philadelphia, Pa, USA, 2000.