Linear and Nonlinear Analysis
Volume 6, Number 2, 2020, 281–286
Yokohama Publishers
Copyright 2020
ISSN 2188-8167

COMPUTATION TIME OF ITERATIVE METHODS FOR NONSMOOTH CONVEX OPTIMIZATION WITH FIXED POINT CONSTRAINTS OF QUASI-NONEXPANSIVE MAPPINGS

KENGO SHIMIZU AND HIDEAKI IIDUKA

Abstract. There are various computation methods for minimizing the sum of convex functions over the intersection of fixed point constraints of quasi-nonexpansive mappings. We have determined the actual computation times of the parallel proximal point, parallel subgradient, and incremental subgradient methods by using parallel computing on multi-core processors for a concrete convex optimization problem. The results show that the larger the number of cores, the shorter the actual computation times of the two parallel methods and that, when the number of cores is fixed, the parallel proximal point method is faster than the parallel subgradient method.

1. Introduction

Convex optimization over the intersection of fixed points of (quasi-)nonexpansive mappings [4, Problem 2.1] is of great interest since it has several potentially useful applications, such as network resource allocation [3, 5] and machine learning [2, 6]. Considering the convex optimization problem enables us to deal with convex optimization with complicated constraints. For example, let us consider convex optimization over the set of minimizers of a smooth, convex function g with Lipschitz continuous gradient ∇g. Since the set coincides with the fixed point set of $\mathrm{Id} - \lambda\nabla g$, where $\lambda > 0$ depends on the Lipschitz constant of ∇g, convex optimization over the set of minimizers of a smooth convex function can be regarded as convex optimization over the fixed point set.

Various iterative methods, such as parallel and incremental subgradient methods [4] and the parallel proximal point method [7], have been presented for solving the convex optimization problem, and their convergence analyses have been presented [4, 7]. We have determined the actual computation times of these methods using parallel computing on multi-core processors for a concrete convex optimization problem.

Let $\mathbb{R}^N$ be an N-dimensional Euclidean space with inner product $\langle\cdot,\cdot\rangle$ and its norm $\|\cdot\|$. We use the standard notation $\mathbb{N}$ for the natural numbers including zero. Let Id be the identity mapping on $\mathbb{R}^N$. Let $f \colon \mathbb{R}^N \to (-\infty, +\infty]$ be proper, lower semicontinuous, and convex. Then, the proximity operator of f [1, Definition 12.23], denoted by Prox_f, maps every $x \in \mathbb{R}^N$ to the unique minimizer of $f(\cdot)+(1/2)\|x-\cdot\|^2$. The subdifferential [1, Definition 16.1] of f is defined for all $x \in \mathbb{R}^N$ by $\partial f(x) :=$

2010 *Mathematics Subject Classification.* 65K05, 90C25, 90C90.

Key words and phrases. Fixed point, incremental method, nonsmooth convex optimization, parallel computing, proximal point method, quasi-nonexpansive mapping.

$\{u \in \mathbb{R}^N \colon f(y) \geq f(x) + \langle y - x, u\rangle \ (y \in \mathbb{R}^N)\}$. The fixed point set of a mapping $Q \colon \mathbb{R}^N \to \mathbb{R}^N$ is denoted by $\mathrm{Fix}(Q) := \{x \in \mathbb{R}^N \colon Q(x) = x\}$. The sublevel set of a function $g \colon \mathbb{R}^N \to \mathbb{R}$ is denoted by $\mathrm{lev}_{\leq 0} g := \{x \in \mathbb{R}^N \colon g(x) \leq 0\}$.

2. Numerical comparisons

Here we consider the following problem [4, Problem 5.1]. Let $a_{i,j} > 0$, $b_{i,j}, d_i \in \mathbb{R}$ $(i \in \mathcal{I} = \{1, 2, \ldots, I\}, j = 1, 2, \ldots, N)$, and $c_i := (c_{i,j})_{j=1}^N \in \mathbb{R}^N$ with $\|c_i\| = 1$. Then,

$$\text{minimize } f(x) := \sum_{i \in \mathcal{I}} f_i(x) \text{subject to } x \in \bigcap_{i \in \mathcal{I}} \mathrm{Fix}(Q_i) = \bigcap_{i \in \mathcal{I}} \mathrm{lev}_{\leq 0} g_i, \tag{2.1}$$

where $f_i \colon \mathbb{R}^N \to \mathbb{R}$ and $Q_i \colon \mathbb{R}^N \to \mathbb{R}^N$ are defined for all $x := (x_j)_{j=1}^N \in \mathbb{R}^N$ by

$$f_i(x) := \sum_{j=1}^N a_{i,j} |x_j - b_{i,j}| \ \text{ and } Q_i(x) := \begin{cases} x - \dfrac{g_i(x)}{\|z_i(x)\|^2} z_i(x) & \text{if } g_i(x) > 0, \\ x & \text{if } x \in \mathrm{lev}_{\leq 0} g_i, \end{cases}$$

and $g_i \colon \mathbb{R}^N \to \mathbb{R}$ is defined for all $x \in \mathbb{R}^N$ by

$$g_i(x) := \begin{cases} \langle c_i, x\rangle + d_i & \text{if } \langle c_i, x\rangle > -d_i, \\ 0 & \text{otherwise}, \end{cases}$$

and $z_i(x)$ is any vector in $\partial g_i(x)$. Let $Q_{\alpha,i} := \alpha \mathrm{Id} + (1 - \alpha) Q_i$, where $\alpha \in (0, 1)$.

The incremental subgradient method (ISM) [4, Algorithm 4.1] is shown in Algorithm 1.

Algorithm 1 ISM

```
Require: $(\gamma_n)_{n \in \mathbb{N}} \subset (0, +\infty)$
1: $n \leftarrow 0$, $x_0 := x_{0,0} \in H$
2: loop
3:     for $i = 1$ to $i = I$ do
4:         $g_{n,i} \in \partial f_i(Q_{\alpha,i}(x_{n,i-1}))$
5:         $x_{n,i} := Q_{\alpha,i}(x_{n,i-1}) - \gamma_n g_{n,i}$
6:     end for
7:     $x_{n+1} = x_{n+1,0} := x_{n,I}$
8:     $n \leftarrow n + 1$
9: end loop
```

The parallel subgradient method (PSM) [4, Algorithm 3.1] is shown in Algorithm 2.

Algorithm 2 PSM

```
Require: $(\gamma_n)_{n\in\mathbb{N}} \subset (0,+\infty)$
1: $n \leftarrow 0$, $x_0 \in H$
2: loop
3:     for $i = 1$ to $i = I$ do
4:         $g_{n,i} \in \partial f_i(Q_{\alpha,i}(x_n))$
5:         $x_{n,i} := Q_{\alpha,i}(x_n) - \gamma_n g_{n,i}$
6:     end for
7:     $x_{n+1} = \frac{1}{I}\sum_{i\in\mathcal{I}} x_{n,i}$
8:     $n \leftarrow n+1$
9: end loop
```

The parallel proximal point method (PPM) [7, Algorithm 1] is shown in Algorithm 3.

Algorithm 3 PPM

```
Require: $(\gamma_n)_{n\in\mathbb{N}} \subset (0,+\infty)$
1: $n \leftarrow 0$, $x_0 \in H$
2: loop
3:     for $i = 1$ to $i = I$ do
4:         $x_{n,i} := Q_i\left(\mathrm{Prox}_{\gamma_n f_i}(x_n)\right)$
5:     end for
6:     $x_{n+1} = \frac{1}{I}\sum_{i\in\mathcal{I}} x_{n,i}$
7:     $n \leftarrow n+1$
8: end loop
```

The actual computation times of these methods were determined experimentally using a FUJITSU PRIMERGY RX2540 M4 system with a 2.40 GHz Intel Xeon Gold 6148 CPU processor, 384 GB memory, and the Red Hat Enterprise Linux 7.6 OS. The three methods were implemented in Python 3.6.9 with the NumPy 1.18.1 package. We set $I = 256$, $N = 1000$, and $\alpha = 1/2$ and chose randomly $a_i \in (0, 100]$, $b_i \in [-100, 100)$, $d_i \in [-1, 0)$, and $c_{i,j} \in [-0.5, 0.5)$. Step size was $\gamma_n := 10^{-3}/n$, which satisfies the conditions in the convergence analyses [4, 7] of PPM, PSM, and ISM. Two performance measures were used for $n \in \mathbb{N}$:

$$D_n := \sum_{i\in\mathcal{I}} \|x_n - Q_i(x_n)\| \text{ and } F_n := \sum_{i\in\mathcal{I}} f_i(x_n),$$

where $(x_n)_{n\in\mathbb{N}}$ is the sequence generated by each of the three algorithms with the randomly chosen initial point $x_0 \in [0, 1)^N$.

Figure 1 plots the actual calculation times against the number of cores. Since ISM cannot be implemented on multi-core processors, the results for ISM are shown for only one core. The stopping condition was $n = 10^4$. Figure 1 indicates that, when

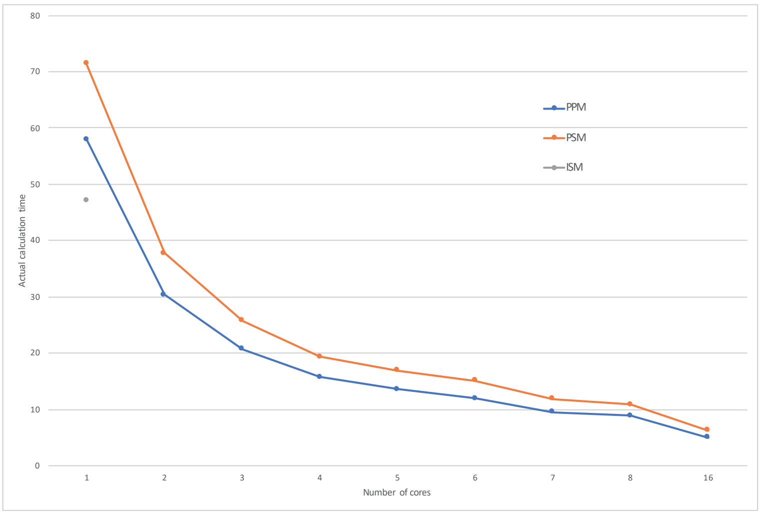

FIGURE 1. Actual calculation time against number of cores

the number of cores was one, the actual calculation time of PPM (resp. PSM) was 57.96 (resp. 71.37) s and the actual calculation time of ISM was 47.00 s. Although ISM was the fastest on one core, PPM and PSM can be implemented on multiple cores, so they can solve the problem faster than ISM. For example, when two cores were used, PPM (resp. PSM) took 30.32 (resp. 37.76) s to solve the problem, which is about half the time taken by PPM (resp. PSM) for serial calculation. Furthermore, when 16 cores were used, PPM (resp. PSM) took 5.06 (resp. 6.24) s. Therefore, the greater the number of cores, the shorter the actual computation time of the parallel methods.

TABLE 1. Values of F_n and D_n when two cores were used

	F_n	D_n
PPM	640230921.5034176	0.5281329190044164
PSM	640229653.1002244	2.71517836075151 56

TABLE 2. Values of F_n and D_n when four cores were used

	F_n	D_n
PPM	640230925.6562285	0.420344254264138 35
PSM	640228917.1070017	0.7841617699210753

Tables 1, 2, 3, and 4 show the values of F_n and D_n when the number of cores used were, respectively, 2, 4, 8, and 16 and the stopping condition was 4 s. There were no significant differences in F_n. For PPM and PSM, D_n decreased as the number of cores increased, with PPM showing a smaller value of D_n. This means that, when the number of cores is fixed, PPM solves the problem faster than PSM.

Table 3. Values of F_n and D_n when eight cores were used

	F_n	D_n
PPM	640232204.0944406	0.34500715972864293
PSM	640228658.1872368	0.4505422109431341

Table 4. Values of F_n and D_n when 16 cores were used

	F_n	D_n
PPM	640236043.7194792	0.260459959391738
PSM	640229163.6883407	0.390958207796753

3. Conclusion

Experimental determination of the actual computation time of parallel incremental and proximal methods for minimizing the sum of convex functions over the intersection of fixed point constraints of quasi-nonexpansive mappings showed that the parallel proximal point method has a shorter computation time than the parallel and incremental subgradient methods.

Acknowledgments

The authors are grateful to Professor Wataru Takahashi of the Tokyo Institute of Technology for giving us the chance to submit our paper to Linear and Nonlinear Analysis. This work was supported by the Japan Society for the Promotion of Science (JSPS KAKENHI Grant Number JP18K11184).

References

[1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.

[2] Y. Hayashi and H. Iiduka, *Optimality and convergence for convex ensemble learning with sparsity and diversity based on fixed point optimization*, Neurocomputing **273** (2018), 367–372.

[3] H. Iiduka, *Fixed point optimization algorithms for distributed optimization in networked systems*, SIAM J. Optim. **23** (2013), 1–26.

[4] H. Iiduka, *Convergence analysis of iterative methods for nonsmooth convex optimization over fixed point sets of quasi-nonexpansive mappings*, Math. Program. **159** (2016), 509–538.

[5] H. Iiduka, *Distributed optimization for network resource allocation with nonsmooth utility functions*, IEEE Trans. Control. Netw. Syst. **6** (2019), 1354–1365.

[6] H. Iiduka, *Stochastic fixed point optimization algorithm for classifier ensemble*, IEEE Trans. Cybern. (2020).

[7] K. Shimizu, K. Hishinuma and H. Iiduka, *Parallel computing proximal method for nonsmooth convex optimization with fixed point constraints of quasi-nonexpansive mappings*, Applied Set-Valued Analysis and Optimization **2** (2020), 1–17.

Manuscript received 14 July 2020

K. Shimizu
Computer Science Course, Graduate School of Science and Technology, Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan
E-mail address: kengo@cs.meiji.ac.jp

H. Iiduka
Department of Computer Science, Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan
E-mail address: iiduka@cs.meiji.ac.jp