Incremental Subgradient Algorithms with Dynamic Step Sizes for Separable Convex Optimizations

Dan Yang* Xiangmei Wang†

Abstract We consider the incremental subgradient algorithm employing dynamic step sizes for minimizing the sum of a large number of component convex functions. The dynamic step size rule was firstly introduced by Goffin and Kiwiel [Math. Program., 1999, 85(1): 207-211] for the subgradient algorithm, soon later, for the incremental subgradient algorithm by Nedic and Bertsekas in [SIAM J. Optim., 2001, 12(1): 109-138]. It was observed experimentally that the incremental approach has been very successful in solving large separable optimizations, and that the dynamic step sizes generally have better computational performance than others in the literature. In the present paper, we propose two modified dynamic step size rules for the incremental subgradient algorithm and analyse the convergence properties of them. At last, the assignment problem is considered and the incremental subgradient algorithms employing different kinds of dynamic step sizes are applied to solve the problem. The computational experiments show that the two modified ones converges dramatically faster and stabler than the corresponding one in [SIAM J. Optim., 2001, 12(1): 109-138]. Particularly, for solving large separable convex optimizations, we strongly recommend the second one (see Algorithm 3.3 in the paper) since it has interesting computational performance and is the simplest one.

Keywords Separable convex optimization; Incremental subgradient algorithm; Dynamic step size; Diminishing step size

1 Introduction

We consider the following separable convex optimization problem

$$\min_{x\in X} f(x) := \sum_{i=1}^{m} f_i(x), \tag{1.1}$$

*College of Mathematics and Statistics, Guizhou University, Guiyang 550025, P. R. China (DanYang-gzu@126.com).

†Corresponding author, College of Mathematics and Statistics, Guizhou University, Guiyang 550025, P. R. China (xmwang2@gzu.edu.cn) Research of this author was partially supported by the National Natural Science Foundation of China (grant 11661019) and the Guizhou Provincial Natural Science Foundation of China (grant 20161039).

where $X \subseteq \mathbb{R}^n$ is a nonempty, closed and convex subset, each $f_i : \mathbb{R}^n \to \mathbb{R}$ is a convex function and m is always a very large number. Such kind of optimizations arise in many applications, such as neural network training, machine learning and tomographic image reconstruction; see, e.g., [2, 3, 4, 5, 6, 7] and the references therein.

One of the ordinary algorithms for solving problem (1.1) is the subgradient algorithm: having x^k, set

$$x^{k+1} := P_X(x^k - \alpha_k \sum_{i \in I} g_{i,k}), \tag{1.2}$$

where $g_{i,k}$ is a subgradient of f_i at x^k, $\alpha_k > 0$ is a step size and P_X stands the projection on X. The subgradient algorithm was proposed by Shor in 1960s ([8]) and has been extensively studied in the literature; see e.g., [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19] and references therein. Since the negative subgradient is not always the descent direction of f, the step size sequence $\{\alpha_k\}$ should be chosen such that the generated sequence approaches the solution set. On can find many classical step sizes for the algorithm in the literature, including the constant step sizes, the diminishing step sizes, the Polyak step sizes and the dynamic step sizes in [8, 9, 10, 11, 12, 13]. The incremental subgradient algorithm seems first proposed and studied by Nedic and Bertsekas in [2] for solving problem (1.1), which is motivated by the idea of the incremental gradient algorithm for separable smooth optimizations (see, e.g., [4, 5, 6, 20, 21] and the references therein). The algorithm is similar to the ordinary subgradient algorithm (1.2). The main difference is that at each iteration, the iterate is updated incrementally by a cycle of m subiterations. Where, each subiteration is a subgradient iteration for a single component function f_i. More precisely, letting x^k be the current iterate, the next iterate x^{k+1} is obtained as follows

$$x^{k+1} = \varphi_{m,k}, \tag{1.3}$$

where $\varphi_{m,k}$ is obtained after the m steps

$$\varphi_{i,k} = P_X[\varphi_{i-1,k} - \alpha_k g_{i,k}], \quad g_{i,k} \in \partial f_i(\varphi_{i-1,k}) \quad i = 1, 2, \cdots, m, \tag{1.4}$$

starting with

$$\varphi_{0,k} = x^k. \tag{1.5}$$

The updates described by (1.4) are referred to as the subiterations of the kth cycle. The authors in [2] established the convergence properties of the incremental subgradient algorithms employing a number of variants of step size rules. In particular, we note that some numerical experiments provided there show that the incremental subgradient algorithm often converges much faster than the corresponding ordinary one (employing the same step sizes), and in most cases, the dynamic step size rule performs better than others. It is known that the classical dynamic step size rule was first introduced for the subgradient algorithm by Goffin and Kiwiel in [1] where two parameters δ_0 and R_0 should be given in advance (noting that δ_0 is updated dynamically and R_0 is fixed).

Inspired by these studies, we propose two modified dynamic step size rules for the incremental subgradient algorithm; see, Algorithms 3.2 and 3.3 in Section 3. The convergence properties of them are analyzed, respectively. At last, the assignment problem is considered and the incremental subgradient algorithms employing different kinds of dynamic step sizes are applied to solve the problem. The computational experiments show that the two modified ones converges dramatically faster and stabler than the corresponding one in [2]. Particularly, for solving large separable convex optimizations, we strongly recommend Algorithm 3.3 since it has interesting computational performance and is the simplest one in the sense that there is only one parameter δ_0 should be given in advance.

The paper is organized as follows. As usual, some basic notions, notation and preliminary results are provided in the next section. In section 3, two modified dynamic step size rules for the incremental subgradient algorithm are introduced, together with convergence analyses. In section 4, we consider the assignment problem, and we apply the incremental subgradient algorithms employing different dynamic step sizes to solve the problem, together with some numerical experiments. The last section includes some conclusions.

2 Notation and Preliminaries

Notation and terminologies used in the present paper are standard; the readers are referred to some textbooks for more details; see, e.g., [22, 23]. Let $\mathbb{R}^n$ be the n dimensional Euclidean space and the standard Euclidean norm is denoted by $\|\cdot\|$. Let $X \subseteq \mathbb{R}^n$ be a non-empty subset of $\mathbb{R}^n$ and $x \in X$. The normal cone of the set X at point x is denoted by $N_X(x)$, that is,

$$N_X(x) := \{\upsilon \in \mathbb{R}^n | \langle \upsilon, y - x\rangle \leq 0 \quad \forall y \in X\}.$$

The distance and the projection at the point x in relation to the set X are denoted by $\mathrm{d}_X(x)$ and $P_X(x)$:

$$\mathrm{d}_X(x) := \inf_{y\in X}\{\|x - y\|\}, \quad P_X(x) := \{y \in X : \|x - y\| = \mathrm{d}_X(x)\}.$$

Recall that the subset X is said to be convex if for any $x, y \in X$ and any $\lambda \in (0, 1)$, there is $\lambda x + (1 - \lambda)y \in X$. The following properties of P_X are well known in the case when X is convex.

Proposition 2.1. *Let $X \subset \mathbb{R}^n$ be a closed convex set and $x \in \mathbb{R}^n$. Then $P_X(x)$ is a singleton, and the following assertions hold.*

(i) *$P_X(x) = \{y_x\}$ if and only if*

$$\langle x - y_x, y - y_x\rangle \leq 0 \quad \forall y \in X.$$

(ii) *P_X is nonexpansive, that is,*

$$\|P_X(x_1) - P_X(x_2)\| \leq \|x_1 - x_2\| \quad \forall x_1, x_2 \in \mathbb{R}^n.$$

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function defined on $\mathbb{R}^n$. Recall that f is said to be convex function if the following inequality holds for any $x, y \in \mathbb{R}^n$

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y) \quad \forall \lambda \in (0,1).$$

Let $\partial f(x)$ stands the subdifferential of f at x, which is defined by

$$\partial f(x) = \{g \in \mathbb{R}^n | f(y) \geq f(x) + \langle g, y - x\rangle, y \in \mathbb{R}^n\}.$$

Any element $g \in \partial f(x)$ is called a subgradient of f at x. The following result gives some properties of convex functions.

Proposition 2.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and $x \in \mathbb{R}^n$. Then, f is continuous on $\mathbb{R}^n$ and $\partial f(x)$ is a non-empty bounded closed convex subset in $\mathbb{R}^n$.*

The following proposition provides the necessary and sufficient condition for a point to be a solution of problem (1.1).

Proposition 2.3. *A point $x^* \in X$ is a solution of the problem* (1.1) *if and only if*

$$0 \in \partial f(x^*) + N_X(x^*).$$

3 Incremental subgradient algorithms employing dynamic step sizes and convergence properties

We consider here the incremental subgradient algorithm employing dynamic step sizes for solving the separable convex optimization (1.1):

$$\min_{x \in X} f(x) := \sum_{i=1}^{m} f_i(x),$$

where each $f_i : \mathbb{R}^n \to \mathbb{R}$ is convex and $X \subseteq \mathbb{R}^n$ is a nonepmty, closed and convex subset. Two modified dynamic step sizes for the incremental subgradient algorithm shall be proposed below. For this puepose, we recall the following incremental subgradient algorithm employing the classical dynamic step sizes, which was said to be path-based incremental target level algorithm in [2].

Algorithm 3.1. *([2, path-based incremental target level algorithm])*

Step 0 *Select $x^0 \in \mathbb{R}^n$, $R_0 > 0, \delta_0, C > 0$. Let $k = l = m = k(l) = 0$, $\sigma_0 = 0$ and $f_{rec}^{-1} = +\infty$.*

Step 1 *If $f(x^k) < f_{rec}^{k-1}$, then set $f_{rec}^k = f(x^k)$, $x_{rec}^k = x^k$; otherwise, set $f_{rec}^k = f_{rec}^{k-1}, x_{rec}^k = x_{rec}^{k-1}$.*

Step 2 *If $0 \in \partial f(x^k) + N_X(x^k)$, then stop; otherwise, go to Step1.*

Step 3 *If* $f(x^k) \leq f_{rec}^{k(l)} - \frac{1}{2}\delta_l$, *then set* $k(l+1) = k, \sigma_k = 0$, $\delta_{l+1} = \delta_l$ *and* $l = l + 1$.

Step 4 *If* $\sigma_k > R_0$, *then set* $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \frac{1}{2}\delta_l$ *and* $x^k = x_{rec}^k$. *Choose* $g^k \in \partial f(x_{rec}^k)$, *and set* $l = l + 1$.

Step 5 *Let* $f_{lev}^k = f_{rec}^{k(l)} - \delta_l$. *Calculate* x^{k+1} *via* (1.3)-(1.5) *with the step size*

$$\alpha_k = \gamma_k \frac{f_k - f_{lev}^k}{C^2}, \quad 0 < \underline{\gamma} \leq \gamma_k \leq \overline{\gamma} < 2. \tag{3.1}$$

Step 6 *Set* $\sigma_{k+1} = \sigma_k + \alpha_k C$, $k = k + 1$ *and go to Step1.*

Accordingly, we see that two parameters δ_0 and R_0 are used in Algorithm 3.1 which should be given in advance. The numerical experiments in [2] show that the computational performance of the algorithm relies heavily on these two parameters (noting that δ_0 is updated dynamically by Step 4 and R_0 is fixed). We shall propose two modified dynamic step size rules for the incremental subgradient algorithm. The first one is very similar with Algorithm 3.1, where the only difference is that we add Step 3 below to update parameter R_0 dynamically.

Algorithm 3.2. ***Step 0*** *Select* x^0, $R_0 > 0, \delta_0 > 0$. *Let* $k = l = p = k(l) = \tilde{k}(p) = 0$, $\sigma_0 = 0$ *and* $f_{rec}^{-1} = \infty$.

Steps 1-2 *are same as Steps 1-2 of Algorithm 3.1, respectively.*

Step 3 *If* $f(x^k) \leq f_{rec}^{\tilde{k}(p)} - \frac{1}{p}\delta_0$, *then set* $\tilde{k}(p+1) = k$, $R_{p+1} = \frac{1}{2}R_p$ *and* $p = p + 1$.

Step 4 *If* $\sigma_k > R_p$, *then set* $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \frac{1}{2}\delta_l$ *and* $x^k = x_{rec}^k$. *Choose* $g^k \in \partial f(x_{rec}^k)$, *and set* $l = l + 1$.

Steps 5-7 *are same as Steps 4-6 of Algorithm 3.1, respectively.*

The second one is simpler than Algorithm 3.1 in the sense that only one parameter δ_0 is kept and updated dynamically by a different criteria in Step 3 below.

Algorithm 3.3. ***Step 0*** *Select* x^0, $\delta_0 > 0$, $k = 0$, $l = 0$, $f_{rec}^{-1} = +\infty$.

Steps 1-2 *are same as Steps 1-2 of Algorithm 3.1, respectively.*

Step 3 *If* $f(x^k) \leq f_{rec}^{k-1} - \frac{1}{2}\delta_l$, *then set* $f_{lev}^k = f_{rec}^k - \delta_l$; *otherwise, set* $f_{lev}^k = f_{rec}^{k-1} - \delta_l, l = l + 1, \delta_l = \frac{\delta_0}{\sqrt{l}}$.

Step 4 *is same as Step 5 of Algorithm 3.1.*

Step 5 *Set* $k = k + 1$, *and go to Step 1.*

We always assume for the remainder that the solution set of problem (1.1) is nonempty, that is,

$$X^* = \mathrm{Argmin}_{x \in X} f(x) \neq \emptyset. \tag{3.2}$$

This particularly implies that

$$f^* := \inf_{x \in X} f(x) > -\infty. \tag{3.3}$$

Furthermore, our convergence results need the following assumption, which is also used in [2, Assumption 2.1].

Assumption 3.1. *(subgradient boundedness) There exist constants $C_1, C_2, \ldots, C_m > 0$ such that*

$$\|g\| \leq C_i \quad \forall g \in \partial f_i(x_k) \cup \partial f_i(\varphi_{i-1,k}), \quad i = 1, 2, \ldots, m, \;\; k \in \mathbb{N}. \tag{3.4}$$

We first give the following important lemma, which can be derived directly by a careful look through the proof of [2, Lemma 2.1].

Lemma 3.1. *Let Assumption 3.1 hold and $C := \sum_{i=1}^{m} C_i$ in (3.1). Let $\{x^k\}$ be a sequences generated by the Algorithm 3.2 or Algorithm 3.3. Then the following inequality holds:*

$$\| x^{k+1} - y \|^2 \leq \| x^k - y \|^2 - 2\alpha_k \Big(f(x^k) - f(y) \Big) + \alpha_k^2 C^2 \quad \forall y \in X. \tag{3.5}$$

3.1 Convergence results for Algorithm 3.2

In this subsection, we study Algorithm 3.2 and its convergence properties. To this end, we first give some lemmas. The first lemma is taken from [2, Proposition 2.4], which is regarding the convergence of the incremental subgradient algorithm employing the diminishing step sizes (defined by (3.6)), which is also useful for proving the main result of the present subsection.

Lemma 3.2. *([2, Proposition 2.4]) Let Assumption 3.1 hold, and let $\{x^k\}$ be a sequence generated by (1.3)-(1.5). Suppose that the step sizes $\{\alpha_k\}$ satisfy*

$$\alpha_k > 0, \quad \sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty. \tag{3.6}$$

Then, we have $\lim_{k\to\infty} f(x^k) = f^*$.

Lemma 3.3. *Let $\{x^k\}$ be a sequence generated by Algorithm 3.2. Then, we have $p < \infty$ and $l \to \infty$, and $\lim\limits_{l\to\infty} \delta_l = 0$.*

Proof. From Step 3 of Algorithm 3.2, we see that

$$f(x^{\tilde{k}(j+1)}) \leq f_{rec}^{\tilde{k}(j)} - \frac{1}{j}\delta_1 \quad \forall j \geq 1, \tag{3.7}$$

Summing up the inequalities in (3.7) over $j = 1, 2, \ldots, p$ and noting that $f_{rec}^{\tilde{k}(1)} = f_{rec}^1 = f(x^1)$ (see Step 3), there holds:

$$f(x^{\tilde{k}(p+1)}) \leq f(x^1) - \sum_{i=1}^{p} \frac{\delta_1}{k}. \tag{3.8}$$

Letting $p \to +\infty$, we get that $\lim\limits_{p\to\infty} f(x^{\tilde{k}(p)}) = -\infty$. This contradicts (3.3), and so we have proven $p < \infty$.

Next, we show $l \to \infty$. Assume on the contrary that $l < \infty$. Then, there holds from Step 5 that

$$\sum_{j=k(l)}^{\infty} \alpha_j C \leq R_p.$$

This particularly implies that

$$\lim_{k\to\infty} \alpha_k = 0. \tag{3.9}$$

On the other hand, we see from (3.1) that for any $k \geq k(l)$, there is

$$\alpha_k = \gamma_k \frac{f(x^k) - f_{lev}^k}{C^2} = \gamma_k \frac{f(x^k) - \left(f_{rec}^{k(l)} - \delta_l\right)}{C^2}.$$

Noting from Step 3 that $f(x^k) > f_{rec}^{k(l)} - \frac{1}{2}\delta_l$ (for each $k \geq k(l)$), it follows that $\alpha_k > \frac{\underline{\gamma}\delta_l}{2C^2} > 0$. This contradicts (3.9), and then $l \to \infty$.

We are left to show $\lim_{l\to\infty} \delta_l = 0$. To this end, let $\lim_{l\to\infty} \delta_l = \delta_\infty$. If $\delta_\infty > 0$, then there exists $l_0 \in \mathbb{N}$ such that

$$\delta_l = \delta_\infty \quad \forall l \geq l_0 \tag{3.10}$$

(as $\{\delta_l\}$ is nonincreasing). Recalling that $l \to \infty$, we conclude from Step 4 of Algorithm 3.2 (see also Step 3 of Algorithm 3.1) that

$$f_{rec}^{k(l+1)} \leq f_{rec}^{k(l)} - \frac{1}{2}\delta_l \quad \forall l \geq l_0.$$

Summing up the inequalities in the above expression over $l_0 \leq l \leq n$ and noting (3.10), we get that

$$f_{rec}^{k(n+1)} \leq f_{rec}^{k(l_0)} - \frac{1}{2}(n - l_0)\delta_\infty.$$

Then, we have that $\lim_{l\to\infty} f_{rec}^{k(l)} = -\infty$, which contradicts (3.3). Thus, $\lim_{l\to\infty} \delta_l = 0$ is shown, and the proof is complete. □

We are ready to show the main result of the subsection.

Theorem 3.1. *Let Assumption 3.1 hold and* $C := \sum_{i=1}^{m} C_i$ *in* (3.1). *Let* $\{x^k\}$ *be a sequence generated by Algorithm 3.2. Then* $\lim_{k\to\infty} f_{rec}^k = f^*$.

Proof. Let $\lim_{k\to\infty} f_{rec}^k = f_\infty$ (which exists because of (3.3) and the decreasing monotonicity of $\{f_{rec}^k\}$). Suppose on the contrary that $f_\infty > f^*$, and let $f^{**} \in (f^*, f_\infty)$, $\epsilon = f_\infty - f^{**} > 0$. Noting $\lim_{l\to\infty} \delta_l = 0$ by Lemma 3.3, there is $\bar{l} \in \mathbb{N}$ such that

$$\delta_l \leq \epsilon = f_\infty - f^{**} \quad \forall l \geq \bar{l}.$$

Then, recalling that $\{f_{rec}^k\}$ is decreasing monotone, we see from Step 6 of Algorithm 3.2 (see also Step 5 of Algorithm 3.1) that

$$f_{lev}^k \geq f_\infty - \epsilon \geq f^{**} \quad \forall k \geq k(l). \tag{3.11}$$

By the continuity of f and the convexity of X, one can take $\bar{y} \in X$ such that $f(\bar{y}) = f_{**}$. Applying Lemma 3.1 (to $y = \bar{y}$), we get that

$$\begin{aligned}\|x^{k+1} - \bar{y}\|^2 &\leq \|x^k - \bar{y}\|^2 - 2\alpha_k \left(f(x^k) - f(\bar{y})\right) + \alpha_k^2 C^2 \\ &\leq \|x^k - \bar{y}\|^2 - 2\alpha_k \left(f(x^k) - f_{lev}^k\right) + \alpha_k^2 C^2 \quad \forall k \geq k(\bar{l}),\end{aligned}$$

where the second inequality is from (3.11). In view of (3.1), it follows that

$$\|x^{k+1} - \bar{y}\|^2 \leq \|x^k - \bar{y}\|^2 - \tfrac{2}{\gamma_k}\alpha_k^2 C^2 + \alpha_k^2 C^2 \leq \|x^k - \bar{y}\|^2 - (\tfrac{2}{\overline{\gamma}} - 1)C^2\alpha_k^2 \quad \forall k \geq k(\bar{l}).$$

Summing up the inequalities over $k \geq k(\bar{l})$, we see that

$$\sum_{k \geq k(\bar{l})} (\frac{2}{\overline{\gamma}} - 1)C^2\alpha_k^2 \leq \|x^{k(\bar{l})} - \bar{y}\|^2.$$

Then, we have

$$\sum_{k=k(\bar{l})}^{\infty} \alpha_k^2 < \infty. \tag{3.12}$$

Noting $p < \infty$ by Lemma 3.3, we set

$$L = \{l \in \mathbb{N} | \delta_l = \frac{\delta_{l-1}}{2}, k(l) \geq \tilde{k}(p)\}.$$

Then, the cardinality of L is infinite because of $\lim_{l\to\infty} \delta_l = 0$. Taking into account of Steps 4 and 7 of Algorithm 3.2, we get that

$$\sum_{k=k(l)}^{k(l+1)} C\alpha_k \geq R_p \quad \forall l \in L.$$

This implies that

$$\sum_{k=0}^{\infty} \alpha_k \geq \sum_{l \in L} \sum_{k=k(l)}^{k(l+1)} \alpha_k > \sum_{l \in L} \frac{R_p}{C} = \infty.$$

Combining this and (3.12), Lemma 3.2 is applicable to showing that $\lim_{k\to\infty} f_{rec}^k = f^*$, which is a contradiction. Therefore, $\lim_{k\to\infty} f_{rec}^k = f^*$ is show, completing the proof. □

3.2 Convergence result for Algorithm 3.3

In this subsection, we study the convergence properties of Algorithm 3.3. To proceed, we show the following lemma.

Lemma 3.4. *Let $\{x_k\}$ be the sequence generated by algorithm 3.3. Then, we have $l \to \infty$ and $\lim_{l\to\infty} \delta_l = 0$.*

Proof. If $l < \infty$, then by Step 3 of Algorithm 3.3, the exists $k_0 \in \mathbb{N}$ such that

$$f(x^k) \leq f(x^{k-1}) - \frac{\delta_l}{2} \quad \forall k \geq k_0.$$

Summing up the inequalities over $k \geq k_0$, we get that

$$f(x^k) \leq f(x^{k_0-1}) - (k-k_0)\frac{\delta_l}{2} \quad \forall k \geq k_0.$$

Thus, there holds $\lim_{k\to\infty} f(x^k) = -\infty$, which contradicts (3.3) and so $l \to \infty$. Furthermore, by definition of δ_l, it is clear that $\lim_{l\to\infty} \delta_l = \lim_{l\to\infty} \frac{\delta_0}{l} = 0$. The proof is complete. □

The following theorem is regarding the convergence property of Algorithm 3.3.

Theorem 3.2. *Let Assumption 3.1 hold and* $C := \sum_{i=1}^{m} C_i$ *in* (3.1). *Let* $\{x_k\}$ *be a sequence generated by Algorithm 3.3. Then, we have* $\lim_{k\to\infty} f_{rec}^k = f^*$.

Proof. Let $\lim_{k\to\infty} f_{rec}^k = f_\infty$ (which exists as we have explained at the beginning of the proof of Theorem 3.2). Suppose on the contrary that $f_\infty > f^*$. Noting Step 3 of Algorithm 3.3, we see that either $f_{lev}^k = f_{rec}^k - \delta_l$ or $f_{lev}^{k-1} = f_{rec}^k - \delta_l$ for all $l \in \mathbb{N}$. Recalling $\lim_{k\to\infty} f_{rec}^k = f_\infty$, $l \to \infty$ and $\lim_{l\to\infty} \delta_l = 0$ (from Lemma 3.4), there exists $k_0 \in \mathbb{N}$ such that

$$f_{lev}^k > f^* \quad \forall k \geq k_0. \tag{3.13}$$

Now, take $x^* \in X^* \subseteq X$ (which exists by the blanket assumption (3.2)). Then, by assumption, Lemma 3.1 is applicable (to $y := x^*$) to getting that

$$\|x^{k+1} - x^*\|^2 \leq \|x^k - x^*\|^2 - 2\alpha_k(f(x^k) - f^*) + \alpha_k^2 C^2.$$

In view of (3.1) and (3.13), there holds

$$\begin{aligned} \|x^{k+1} - x^*\|^2 &\leq \|x^k - x^*\|^2 - \gamma_k(2-\gamma_k)\tfrac{(f(x^k)-f^k_{lev})^2}{C^2} \\ &\leq \|x^k - x^*\|^2 - \underline{\gamma}(2-\overline{\gamma})\tfrac{(f(x^k)-f^k_{lev})^2}{C^2} \quad \forall k \geq k_0. \end{aligned}$$

Summing up the inequalities over $k \geq k_0$, it is clear that

$$\sum_{k\geq k_0}^{\infty} \frac{\underline{\gamma}(2-\overline{\gamma})}{C^2}(f(x^k) - f_{lev}^k)^2 \leq \|x^{k_0} - x^*\|^2 < \infty. \tag{3.14}$$

On the other hand, recall again that $l \to \infty$ (from Lemma 3.4). By definition of Algorithm 3.3, we see that l increases only when the second case of Step 3 occurs. Then, we conclude that for each $l \in \mathbb{N}$, there exists $k_l \in \mathbb{N}$ such that

$$f(x^{k_l}) > f_{rec}^{k_l-1} - \frac{1}{2}\delta_l = f_{lev}^{k_l} + \frac{1}{2}\delta_l \quad \text{and} \quad k_{l+1} > k_l.$$

Then, there holds

$$f(x^{k_l}) - f_{lev}^{k_l} > \frac{\delta_l}{2} \quad \forall l \in \mathbb{N}. \tag{3.15}$$

By definition of $\delta_l (:= \frac{\delta_0}{\sqrt{l}})$, we obtain that

$$\sum_{k=0}^{\infty} (f(x^k) - f_{lev}^k)^2 \geq \sum_{l=0}^{\infty} (f(x^{k_l}) - f_{lev}^{k_l})^2 > \sum_{l=0}^{\infty} \frac{\delta_l^2}{4} = \sum_{l=0}^{\infty} \frac{\delta_0^2}{4l} = \infty.$$

This contradicts (3.14), completing the proof. □

4 Numerical results

In this section, Algorithms 3.1, 3.2 and 3.3 are applied to solve the assignment problem. The problem is to assign m jobs to n machines; see more details in [2]. If job i is performed at machine j, it costs $a_{i,j}$ and requires p_{ij} time units. Given the total available time t_j at machine j, we want to find the minimum cost assignment of the jobs to the machines. The problem is formulated as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} y_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} y_{ij} = 1, \quad i = 1, 2, \ldots, m, \\
& \sum_{i=1}^{m} p_{ij} y_{ij} \leq t_j, \quad j = 1, 2, \ldots, n, \\
& y_{ij} = 1 \text{ or } 0, \quad i = 1, 2, \ldots, m, \ j = 1, 2, \ldots, n,
\end{aligned}$$

where y_{ij} is the assignment variable, which is equal to 1 if the ith job is assigned to the jth machine and is equal to 0 otherwise. As pointed out in [2], the dual problem is

$$\begin{aligned}
& \max f(x) := \sum_{i=1}^{m} f_i(x) \\
& \text{s.t.} \quad x \geq 0,
\end{aligned}$$

where

$$f_i(x) := \min_{y_{ij}=1 \text{ or } 0, \ \sum_{j=1}^{n} y_{ij} = 1} \sum_{j=1}^{n} (a_{ij} + x_j p_{ij}) y_{ij} - \frac{1}{m} \sum_{j=1}^{n} t_j x_j, \quad i = 1, 2, \ldots, m.$$

As done in [2], each f_i is evaluated as follows:

$$f_i(x) := a_{ij^*} + x_{j^*} p_{ij^*} - \frac{1}{m} \sum_{j=1}^{n} t_j x_j \quad \forall x \geq 0,$$

where $j^* \in \{1, 2, \ldots, n\}$ is so that

$$a_{ij^*} + x_{j^*} p_{ij^*} = \min_{1 \leq j \leq n} \{a_{ij} + x_j p_{ij}\}.$$

Moreover, a subgradient $g = (g_1, g_2, \cdots, g_n)^T$ of f_i at x is given by

$$g_j = \begin{cases} -\dfrac{t_j}{m} & \text{if } j \neq j^*, \\ p_{ij^*} - \dfrac{t_j^*}{m} & \text{if } j = j^*. \end{cases}$$

In our experiments, we chose $n = 4$ and $m = 800$. The data $\{a_{ij}\}$ and $\{p_{ij}\}$ for the problems are generated randomly according to a uniform distribution over intervals $[1, 5]$ and $[1, 10]$, respectively, and the values t_j were calculated according to the formula $t_j = \frac{1}{2n} \sum_{i=1}^{m} p_{ij}$. The experiment results are reported in Tables 4.1-4.3.

Tables 4.1 and 4.2 report the numerical results of applying Algorithms 3.1-3.3 with the same parameters $R_0 = 5$, $\delta_0 = 5 \times 10^4$ and the same initial point (Algorithm 3.3 is only with the parameter $\delta_0 = 5 \times 10^4$). Where the notation f_{rec}^k/time stands the best estimations of the cost f^* and the CPUtime of the kth iteration. Both tables show that Algorithms 3.2 and 3.3 converges dramatically faster than Algorithm 3.1.

Table 4.3 demonstrates the number of iterations required for Algorithms 3.1-3.3 and parameter choices to achieve a given threshold cost $\bar{f}$. The notation used in the tables is as follows:

> 300 means that the value $\bar{f}$ has not been achieved within 300 iterations.

$R_0/\delta_0/iter$ are the values of the parameters R_0, δ_0 and the number of iterations needed to achieve or exceed $\bar{f}$.

It is shown in Table 4.3 that the computational performance of Algorithm 3.1 relies heavily on parameters (R_0 and δ_0), and Algorithms 3.2 and 3.3 are always more effective and stabler than Algorithm 3.1.

Table 4.1

$x^0 = (0, 0, 0, 0), R_0 = 5, \delta_0 = 5 \times 10^4, f^* \approx 1949.25$

Incremental subgradient algorithm with dynamic step sizes			
Iter k	Algorithm 3.1 f_{rec}^k/time (s)	Algorithm 3.2 f_{rec}^k/time (s)	Algorithm 3.3 f_{rec}^k/time (s)
1	1230.00/0.07	1230.00/0.07	1230.00/0.07
10	1891.03/0.40	1949.01/0.35	1940.36/0.35
50	1913.81/1.59	1949.03/1.35	1949.18/1.35
100	1919.98/4.85	1949.18/4.53	1949.22/4.53
500	1930.93/103.05	1949.25/103.57	1949.25/103.01

Table 4.2

$x^0 = (3, 4, 5, 6), R_0 = 10, \delta_0 = 5 \times 10^5, f^* \approx 1949.25$

Incremental subgradient algorithm with dynamic step sizes			
Iter k	Algorithm 3.1 f_{rec}^ktime (s)	Algorithm 3.2 f_{rec}^k/time (s)	Algorithm 3.3 f_{rec}^k/time (s)
1	690.25/0.07	690.25/0.07	690.25/0.07
10	1435.32/0.18	1449.00/0.18	1632.92/0.18
50	1534.44/1.50	1641.19/1.39	1920.82/1.38
100	1548.73/4.50	1772.63/4.50	1949.01/4.60
500	1571.28/102.32	1949.03/102.59	1949.19/102.14

Table 4.3

$f^* \approx 1949.25, \bar{f} = 1949$

Incremental subgradient algorithm with dynamic step sizes			
Initial point x^0	Algorithm 3.1 $R_0/\delta_0/iter$	Algorithm 3.2 $R_0/\delta_0/iter$	Algorithm 3.3 $\delta_0/iter$
(0,0,0,0)	$1/5 \times 10^4/ > 300$	$1/5 \times 10^4/10$	$5 \times 10^4/19$
(0,0,0,0)	$10/5 \times 10^4/52$	$10/5 \times 10^4/10$	$5 \times 10^4/19$
(0,0,0,0)	$20/5 \times 10^4/30$	$20/5 \times 10^4/10$	$5 \times 10^4/19$
(0,0,0,0)	$30/5 \times 10^4/ > 300$	$30/5 \times 10^4/10$	$5 \times 10^4/19$
(0,0,0,0)	$30/6 \times 10^4/ > 300$	$30/6 \times 10^4/10$	$6 \times 10^4/19$
(0,0,0,0)	$10/1 \times 10^5/196$	$10/1 \times 10^5/30$	$1 \times 10^5/139$
(0,0,0,0)	$20/1 \times 10^5/95$	$10/1 \times 10^5/38$	$1 \times 10^5/139$
(0,0,0,0)	$40/7 \times 10^4/98$	$40/7 \times 10^4/51$	$7 \times 10^4/8$
(0.8,0.5,0.1,1.5)	$5/2 \times 10^4/62$	$5/2 \times 10^4/18$	$2 \times 10^4/38$
(0.8,0.5,0.1,1.5)	$20/6 \times 10^4/136$	$20/6 \times 10^4/39$	$6 \times 10^4/35$
(3,4,5,6)	$20/4.5 \times 10^5/ > 300$	$20/4.5 \times 10^5/170$	$4.5 \times 10^5/93$
(3,4,5,6)	$10/5 \times 10^5/ > 300$	$10/5 \times 10^5/277$	$5 \times 10^5/85$

5 Conclusions

In the present paper, we propose two modified dynamic step size rules for the incremental subgradient algorithm (Algorithms 3.2 and 3.3), and we establish the convergence results of them. Numerical experiments show that the modified ones converges dramatically faster and stabler than the classical one in [2]. Particularly, for solving large separable convex optimizations, we strongly recommend Algorithm 3.3 since it has interesting computational performance and is the simplest one in the sense that there is only one parameter δ_0 should be given in advance. In our future study, we shall use the randomization and the ε-subgradients in the context of the incremental approaches.

References

[1] Goffin, J.L., Kiwiel, K.C.,: Convergence of a simple subgradient level method. Math. Program. **85**(1), 207–211 (1999)

[2] Bertsekas, P.D., Nedic, A.,: Incremental subgradient methods for nondifferentiable optimization. SIAM J. Optim. **12**(1), 109–138 (2001)

[3] Kim, S., Ahn, H.,: Convergence of a generalized subgradient method for nondifferentiable convex optimization. Math. Program. **50**(1-3), 75–80 (1991)

[4] Bertsekas, D.P.,: A new class of incremental gradient methods for least squares problems. SIAM J. Optim. **7**(4), 913–926 (1997)

[5] Gaivoronski, A.A.,: Convergence analysis of parallel backpropagation algorithm for nural network. Optim. Methods Soft. **4**(2), 117–134 (1994)

[6] Grippo, L.,: A class of unconstrained minimization methods for neural network training. Optim. Methods Soft. **4**(2), 135–150 (1994)

[7] Moriyama, H., Yamashita, N., Fukushima, M.,: The incremental Gauss-Newton algorithm with adaptive stepsize rule. J. Comput. Appl. Math. **26**(2), 107–141 (2003)

[8] Shor, N.Z.,: Minimization Methods for Nondifferentiable Functions. Kiev: Naukova Dumka. (1979)

[9] Dem'yanov, V.F., Vasil'ev, L.V.,: Nondifferential Optimization Optim Soft. New York (1985)

[10] Poljak, B.T.,: Minimization of nonsmooth functionals. Gaea. **300**(1), 752–754 (1985)

[11] Minoux, M.,: Solving integer minimum cost flows with separable convex cost objective polynomially. Springer Berlin Heidelberg (1986)

[12] Kaskavelis, C.A., Caramanis, M.C.,: Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. AIIE Transactions. **30**(11), 1085–1097 (1998)

[13] Bertsekas, D.P.,: Nonlinear Programming. Journal of the Operational Research Society. **48**(3): 334-334. (1997)

[14] Larsson, T., Patriksson, T., Strömberg, A.,: On the convergence of conditional ε-subgradient methods for convex programs and convex-concave saddle-point problems. J Oper Res. **151**(3), 461–473 (2003)

[15] Ruszczyński, A.,: A merit function approach to the subgradient method with averaging. Optim Methods and Softw. **23**(1), 161–172 (2008)

[16] Nesterov, Y.,: Primal-dual subgradient methods for convex problems. Math Program.**120**(1), 221–259 (2009)

[17] Burachik, R.S., Iusem, A.N., Jefferson, J.G.,: An inexact modified subgradient algorithm for primal-dual problems via augmented Lagrangians. J Optim Theory Appl. **157**(1), 108–131 (2013)

[18] Wang, X.M.,: Subgradient algorithms on Riemannian manifolds of lower bounded curvatures. Optimization. **67**(1), 1–16.(2018)

[19] Yao, Y.H., Naseer, S., Yao, J.C.,: Projected subgradient algorithms for pseudomonotone equilibrium problems and fixed points of pseudocontractive operators. Mathematics. **8**(4), 461–476 (2020)

[20] Solodov, M.V., Zavriev, S.K.,: Incremental gradient algorithms with stepsizes bounded away from zero. Comput. Optim. Appl. **11**(1), 23–35 (1998)

[21] Tseng, P.,: An Incremental gradient(-projection) method with momentum term and adaptive stepsize rule. SIAM J.Optim. **8**(2), 506–531 (1998)

[22] Hiriart-Urruty, J.B., Lemarchal, L.,: Fundamentals of convex analysis. Springer, New York (2011)

[23] Rockafellar, R.T.,: Convex Analysis. Princeton Mathematical Series (1970)

[24] Martello, S., Toth, P.,: Knapsack problems. Wily J. New York (1990)