AN OPTIMAL ALGORITHM FOR CONSTRAINED DIFFERENTIABLE CONVEX OPTIMIZATION

CLÓVIS C. GONZAGA[†], ELIZABETH W. KARAS[‡], AND DIANE R. ROSSETTO[§][¶]

June 6, 2011

Abstract. We describe three algorithms for solving differentiable convex optimization problems constrained to simple sets in $\mathbb{R}^n$, i.e., sets on which it is easy to project an arbitrary point. The first two algorithms are optimal in the sense that they achieve an absolute precision of ε in relation to the optimal value in $O(1/\sqrt{\varepsilon})$ iterations using only first order information. This complexity depends on a Lipschitz constant L for the function derivatives and on a strong convexity constant $\mu \geq 0$. The first algorithm extends to the constrained case a well-known method devised by Nesterov [7] for unconstrained problems, and includes a procedure guessing the unknown value of L. The complexity analysis follows a simpler geometric approach. The other algorithms have several enhancements, including line searches that improve the performance: the second algorithm is enhanced and optimal; the third relaxes somewhat the optimality to obtain the best practical performance. Numerical tests for box-constrained quadratic problems are presented in the last section.

1. Introduction.

We study the nonlinear programming problem

$$(P) \qquad \begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \Omega, \end{array}$$

where $\Omega \subset \mathbb{R}^n$ is a closed convex set and $f : \mathbb{R}^n \to \mathbb{R}$ is convex and continuously differentiable, with a Lipschitz constant $L > 0$ for the gradient and a convexity parameter $\mu \geq 0$. It means that for all $x, y \in \mathbb{R}^n$,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \tag{1.1}$$

and

$$f(x) \geq f(y) + \nabla f(y)^T(x - y) + \frac{1}{2}\mu\|x - y\|^2. \tag{1.2}$$

If $\mu > 0$, the function is said to be *strongly convex*. Note that $\mu \leq L$.

Simple sets: we assume that Ω is a "simple" set, in the following sense: given an arbitrary point $x \in \mathbb{R}^n$, an oracle is available to compute $P_\Omega(x) = \operatorname*{argmin}_{y\in\Omega} \|x - y\|$, the orthogonal projection onto the set Ω. A well-known algorithm for solving Problem (P) is the projected gradient method described by Bertsekas [2]. Our methods will be based only on first-order information, and each iteration will contain a projected gradient step.

Optimal methods: the main reference for this paper is the book by Yurii Nesterov [7], and our algorithms are extensions of his basic method described in Chapter 2. His method [7, Algorithm 2.2.6] solves unconstrained convex problems and is in some sense a short steps method. We shall show how to modify his method to deal with constrained problems, and then show how to improve the speed at the cost of inexact projected line searches.

[†]Department of Mathematics, Federal University of Santa Catarina. Cx. Postal 5210, 88040-970 Florianópolis, SC, Brazil; e-mail: clovis@mtm.ufsc.br.

[‡]Department of Mathematics, Federal University of Paraná. Cx. Postal 19081, 81531-980 Curitiba, PR, Brazil; e-mail: ewkaras@ufpr.br.

[§]Department of Mathematics, University of São Paulo. SP, Brazil; e-mail: dianerr@ime.usp.br.

[¶]The authors are supported by CNPq.

For a class of problems and a specific information accessible about each problem (the oracle), a method is called "optimal" if its worst case complexity is proved to be minimal. This is well explained in the classical book by Nemirovskii and Yudin [6]. For our problem, it is shown in this reference that the absolute precision attainable (in the worst case) in k steps of a method using only first order information is given by $f(x_k) - f^* = O\left(1/k^2\right)$, where f^* is the value of an optimal solution. This is again proved by Nesterov [7], who also shows that the classical constant step steepest descent (as well as the projected gradient method) cannot ensure a performance better than $O\left(1/k\right)$.

Nesterov's algorithms cited above are optimal. He also describes methods for constrained problems, which are extensively developed in [8]. Nesterov's approach is applied to penalty methods by Lan, Lu and Monteiro [5], and interior descent methods based on Bregman distances are described by Auslender and Teboulle [1]. This method has been generalized to a non-interior method using projections by Rossetto [9].

In this paper we show that Nesterov's basic algorithm can be very easily extended to the constrained case, and the proof is based on a simple geometrical construction developed in Section 2. The basic (short steps) method developed in this section includes a procedure for updating estimated values L for the Lipschitz constant: these estimated values only increase, and no updating is done if L is an actual Lipschitz constant for $f(\cdot)$ on the set Ω.

Section 3 presents algorithms with several enhancements, including line searches to improve the efficiency of each iteration, and an improvement in the points from where the projected gradient steps are taken.

Numerical results are presented in Section 4.

Here we describe some useful tools.

Simple quadratic functions

We shall call "simple" a quadratic function $\phi : \mathbb{R}^n \to \mathbb{R}$ with $\nabla^2\phi(x) = \gamma I$, $\gamma \in \mathbb{R}$, $\gamma > 0$. The following facts are easily proved for such functions:

- $\phi(\cdot)$ has a unique minimizer $v \in \mathbb{R}^n$ (which we shall refer as the *center* of the quadratic), and the function can be written as
$$x \in \mathbb{R}^n \mapsto \phi(v) + \frac{\gamma}{2}\|x - v\|^2.$$
- Given a point $x \in \mathbb{R}^n$ and a closed convex set $\Omega \subset \mathbb{R}^n$,
$$\operatorname*{argmin}_{x\in\Omega} \phi(x) = P_\Omega(v),$$
where P_Ω represents the orthogonal projection onto the set Ω.
- Given $x \in \mathbb{R}^n$,
$$v = x - \frac{1}{\gamma}\nabla\phi(x), \tag{1.3}$$
and
$$\phi(x) - \phi(v) = \frac{1}{2\gamma}\|\nabla\phi(x)\|^2. \tag{1.4}$$

Given a simple quadratic ϕ with center $v \in \mathbb{R}^n$, we can construct another simple quadratic ϕ_+ with center $v_+ \in \Omega$ with the same Hessian and such that $\phi_+ \leq \phi$ in Ω:

Lemma 1.1. *Let Ω be a convex set in $\mathbb{R}^n$, and consider a simple quadratic defined in $\mathbb{R}^n$ by $\phi(x) = \phi^* + \gamma\|x - v\|^2/2$. Define $\phi_+(x) = \phi_+^* + \gamma\|x - v_+\|^2/2$, with $v_+ = P_\Omega v$ and $\phi_+^* = \phi(v_+)$. Then for all $x \in \Omega$, $\phi_+(x) \le \phi(x)$.*

Proof. By definition, $v_+ = P_\Omega(v) = \operatorname*{argmin}_{x\in\Omega} \phi(x)$. Hence, by optimality, $-\nabla\phi(v_+)$ belongs to the normal cone of Ω at v_+, i.e., for all $x \in \Omega$

$$\nabla\phi(v_+)^T(x - v_+) \ge 0.$$

It follows that for all $x \in \Omega$,

$$\begin{aligned}\phi(x) &= \phi(v_+) + \nabla\phi(v_+)^T(x - v_+) + \frac{\gamma_+}{2}\|x - v_+\|^2\\ &\ge \phi_+^* + \frac{\gamma_+}{2}\|x - v_+\|^2\\ &= \phi_+(x).\end{aligned}$$

□

2. The algorithm. Consider problem (P), assume that a strong convexity constant $\mu \ge 0$ is known and assume that a Lipschitz constant estimate $L > \mu$ is given. We state the basic algorithm and then comment on each iteration. We include in the statement of the algorithm the definitions of the relevant functions (lower and upper approximations of $f(\cdot)$ and the simple quadratics).

Algorithm 1. *Basic Algorithm*

```
Data: $x_0 \in \Omega$, $v_0 = x_0$, $\mu > 0$ (convexity constant),
    $L > \mu$ (estimated Lipschitz constant), $\gamma_0 = L$, $\phi_0^* = f(x_0)$.
$k = 0$
REPEAT
    Compute $\alpha_N$ as the largest root of $L\alpha^2 = \alpha\mu + (1-\alpha)\gamma_k$
    Set $\theta_N = \frac{\gamma_k}{\gamma_k + \mu\alpha_N}\alpha_N$
    $y_N = x_k + \theta_N(v_k - x_k)$
    Compute $g = \nabla f(y_N)$
        Define $x \mapsto u(x) = f(y_N) + g^T(x - y_N) + \frac{L}{2}\|x - y_N\|^2$
    Compute $x_N = P_\Omega\left(y_N - \frac{1}{L}g\right)$
    IF $f(x_N) > u(x_N)$, set $L = 2L$ and restart the iteration.
    Updates:
        Define $x \mapsto \phi_k(x) = \phi_k^* + \frac{\gamma_k}{2}\|x - v_k\|^2$
        Define $x \mapsto \ell(x) = f(y_k) + \nabla f(y_k)^T(x - y_k) + \frac{\mu}{2}\|x - y_k\|^2$
        Define $x \mapsto \phi_\alpha(x) = \alpha\ell(x) + (1-\alpha)\phi_k(x)$
    $\alpha_k = \alpha_N$, $y_k = y_N$, $x_{k+1} = x_N$
    $\gamma_{k+1} = \alpha_k\mu + (1-\alpha_k)\gamma_k$
    $v_{k+1} = \operatorname*{argmin}_{x\in\Omega} \phi_\alpha(x) = P_\Omega\left(v_k - \frac{\alpha_k}{\gamma_{k+1}}(g + \mu(v_k - y_k))\right)$
    $\phi_{k+1}^* = \phi_\alpha(v_{k+1})$
    $k = k + 1$.
```

2.1. Analysis of the algorithm. Let us describe the geometry of an iteration, and then analyze it. The iteration starts with two points $x_k, v_k \in \Omega$ (see Fig 2.1) and a simple quadratic function $\phi_k(x) = \phi_k^* + \dfrac{\gamma_k}{2}\|x - v_k\|^2$. A point $y_k = x_k + \theta(v_k - x_k)$ is chosen between x_k and v_k, and two things are done:

- A projected gradient step is taken at y_k, with step length $1/L$.
- A new simple quadratic $\phi_{k+1}(x) = \phi_{k+1}^* + \dfrac{\gamma_{k+1}}{2}\|x - v_{k+1}\|^2$ is generated by the following procedure: v_{k+1} is the minimizer in Ω of a combination $\phi_\alpha(\cdot)$ of $\phi_k(\cdot)$ and a lower approximation $\ell(\cdot)$ of $f(\cdot)$ around y_k: $\phi_\alpha(x) = \alpha\ell(x) + (1-\alpha)\phi_k(x)$. Both $\ell(\cdot)$ and $\phi_k(\cdot)$ are simple quadratics, and hence $\gamma_{k+1} I = \nabla^2\phi_\alpha(x) = (\alpha\mu + (1-\alpha)\gamma_k)\, I$.

The parameters: the most important feature in this scheme is the choice of the parameters α and θ at each iteration. The choice made in the basic algorithm is the same as the one devised by Nesterov in [7, Scheme (2.2.6)]. We basically show that his algorithm extends to the constrained case simply by projecting the vectors x_{k+1} and v_{k+1} that would be obtained for the unconstrained case. Instead of "discovering" the values for these parameters, we shall simply adopt them and show their properties. Instead of using Nesterov's analysis of the function ϕ_α, we develop a new way of showing his result as a step in the study of the constrained case.
Updating L: The parameter α depends on L. The only requirement needed in our analysis for the steepest descent step is that $f(x_{k+1}) \le u(x_N)$. This is trivially true if L is a Lipschitz constant for $\nabla f(\cdot)$ on Ω. The algorithm checks this condition and increases L whenever it is not satisfied. Since L never decreases, it is easy to see that if the algorithm starts with an estimate $L = L_0$, then number of updates of L is bounded by the first integer $p \ge (\log_2(L^*/L_0))$, because after p updates, $L = 2^p L_0$ and so $\log_2(L/L_0) \ge \log_2(L^*/L_0)$.

Note that if the initial value for L is known to be a Lipschitz constant for $\nabla f(\cdot)$, then no updates are needed, and the algorithm needs no function computations at all.

Features:

The algorithm uses local information about $f(\cdot)$ to compute x_{k+1} by a steepest descent step. It also uses global information given by the simple quadratics $\phi_k(\cdot)$: at all iterations, at an optimal solution x^*, $\phi_k(x^*) \ge f(x^*)$, and $\phi_k(\cdot)$ is "pushed down" by constructing $\phi_{k+1}(\cdot)$ as a linear combination of $\phi_k(\cdot)$ and a lower approximation of $f(\cdot)$ (see Fig. 2.1). The choice of the point y_k from where the steepest descent step is taken and the linear combination parameter are chosen to obtain a large reduction in the simple quadratic. The speed of convergence is given by the speed with which the simple quadratic is pushed down.

We shall prove that the choice of α at iteration k ensures that $\phi_{k+1}^* \ge f(x_{k+1})$, and hence the simple functions $\phi_k(\cdot)$ satisfy $\phi_k(x) \ge f^*$ for $x \in \Omega$, where f^* is the value of an optimal solution. These functions have two properties:
(i) $\gamma_{k+1} - \mu = (1-\alpha)(\gamma_k - \mu)$, and then (as we will show) $\gamma_k \to \mu$: the functions become flatter at each iteration.
(ii) $\phi_k(\cdot) - f(\cdot) \le \dfrac{\gamma_k - \mu}{\gamma_0 - \mu}(\phi_0(\cdot) - f(\cdot))$. In particular, at an optimal solution x^*, $\phi_k(x^*) \to f(x^*)$. As $\phi_k(x^*) \ge \phi_k^* \ge f(x_k)$, we conclude that $f(x_k) \to f(x^*)$ with the same speed as $\gamma_k \to \mu$.

We shall analyse one iteration of the algorithm. Our scope will be to prove two facts:

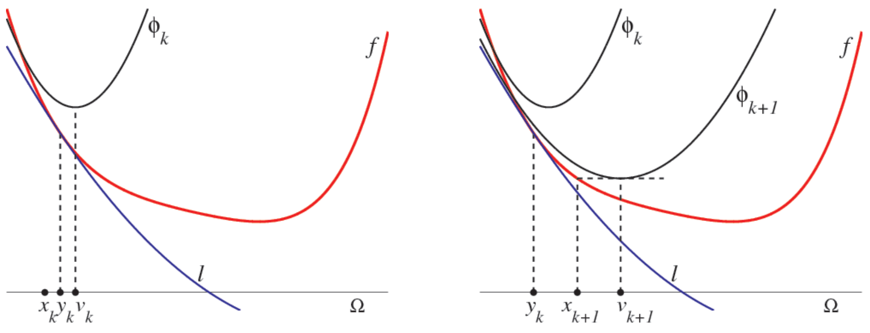

FIG. 2.1. *The mechanics of the algorithm.*

- if the iteration starts with $\phi_k^* \geq f(x_k)$, then it ends with $\phi_{k+1}^* \geq f(x_{k+1})$.
- For all $x \in \Omega$, $\phi_{k+1}(x) - f(x) \leq (1-\alpha_k)(\phi_k(x) - f(x))$.

These two facts will then easily lead to the desired complexity result.

Let us initially simplify the notation at iteration k; without loss of generality we assume that $x_k = 0$ and that $\|v_k - x_k\| = 1$. This can be obtained by a simple change of variables, unless in the case $x_k = v_k$. In this case, we assume that $x_k = v_k = y_k = 0$.

We also drop subscripts, so that all the points, functions and variables will be denoted as follows:

$$\begin{array}{rcl}(0, v, y, v_+, x_+, \alpha, \theta, \gamma, \gamma_+, \phi^*, \phi^*_+) & \equiv & (x_k, v_k, y_k, v_{k+1}, x_{k+1}, \alpha_k, \theta_k, \gamma_k, \gamma_{k+1}, \\ & & \phi_k^*, \phi_{k+1}^*).\end{array}$$

Given the point y, denote $g = \nabla f(y)$. Only two functions related to $f(\cdot)$ will play any role:

- lower approximation: $x \mapsto \ell(x) = f(y) + g^T(x-y) + \dfrac{\mu}{2}\|x-y\|^2$,
  $\nabla \ell(x) = g + \mu(x-y)$, $\nabla^2 \ell(x) = \mu I$.
- upper approximation: $x \mapsto u(x) = f(y) + g^T(x-y) + \dfrac{L}{2}\|x-y\|^2$,
  $\nabla u(x) = g + L(x-y)$, $\nabla^2 u(x) = LI$.

If L is a a Lipschitz constant for $\nabla f(\cdot)$, then for all $x \in \mathbb{R}^n$, $\ell(x) \leq f(x) \leq u(x)$. Also, by hypothesis, $\phi_k^* \geq f(0) \geq \ell(0)$.

In our algorithms L is not necessarily a Lipschitz constant. So, let us write clearly what are the hypotheses made for the treatment in this section concerning the quadratic approximations, using our simplified notation:
Let $x_N = P_\Omega(y - g/L)$.
(H1) $f(x_N) \leq u(x_N)$ (ensured by the algorithm).
(H2) $\phi^* \geq \ell(0)$.
Remark: The actual function $f(\cdot)$ plays no role in the analysis below. Only its upper and lower approximations are relevant, and need only to satisfy the hypotheses above.

During the iteration we construct the function

$$x \mapsto \phi_\alpha(x) = \alpha\ell(x) + (1-\alpha)\phi(x),$$

we have: $y = \theta v$, $v - y = (1-\theta)v$, and then

$$\begin{array}{rcl}\nabla\phi_\alpha(v) & = & \alpha(g + \mu(1-\theta)v) \\ \nabla^2\phi_\alpha(v) & = & (\alpha\mu + (1-\alpha)\gamma))I.\end{array}$$

Properties of the unconstrained problem

We begin by studying properties of the following points, shown in Fig. 2.2:

$$\begin{aligned}\tilde{v} &= \operatorname*{argmin}_{x\in\mathbb{R}^n} \phi_\alpha(x) = v - \frac{1}{\gamma_+}\nabla\phi_\alpha(v) = v - \frac{\alpha}{\gamma_+}(g+\mu(1-\theta)v)\\ \tilde{x} &= \operatorname*{argmin}_{x\in\mathbb{R}^n} u(x) = y - \frac{1}{L}g.\end{aligned}$$

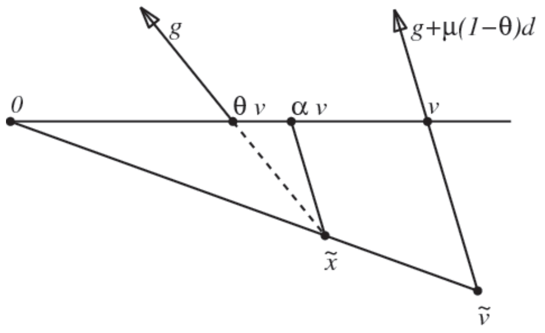

FIG. 2.2. *Geometric properties of the unconstrained case.*

The following expressions will be useful below:
By construction, $\theta = \dfrac{\gamma}{\gamma+\alpha\mu}\alpha$, $\alpha^2 = \gamma_+/L$ and $\gamma_+ = \alpha\mu + (1-\alpha)\gamma$.
By substitution, we get

$$1-\theta = \frac{\gamma_+}{\gamma+\alpha\mu} \tag{2.1}$$

$$\alpha-\theta = \frac{\alpha^2\mu}{\gamma+\alpha\mu} = \frac{\mu}{L}(1-\theta) \tag{2.2}$$

The first lemma shows our main finding about the geometry of these points. All the action happens in the two-dimensional space defined by $0, v, \tilde{v}$. Note the beautiful similarity of the various triangles in Fig 2.2.

LEMMA 2.1. *For the construction above, $\tilde{x} = \alpha\tilde{v}$.*

Proof. By definition of $\tilde{v}$,

$$\alpha\tilde{v} = \alpha v\left(1 - \frac{\alpha\mu(1-\theta)}{\gamma_+}\right) - \frac{\alpha^2}{\gamma_+}g. \tag{2.3}$$

Using (2.1),

$$1 - \frac{\alpha\mu(1-\theta)}{\gamma_+} = 1 - \frac{\alpha\mu}{\gamma+\alpha\mu} = \frac{\gamma}{\gamma+\alpha\mu} = \frac{\theta}{\alpha}.$$

Substituting into (2.3),

$$\alpha\tilde{v} = \theta v - \frac{\alpha^2}{\gamma_+}g.$$

Remembering that $\alpha^2 = \gamma_+/L$ and that $y = \theta v$, we conclude that
$\alpha\tilde{v} = y - \dfrac{1}{L}g$, completing the proof. □

Now we can compare the values of $u(\tilde{x})$ and $\phi_+(\tilde{v})$. This will be done in two steps (observe the parallel lines in Fig 2.2).

Lemma 2.2. *Assume that (H2) holds for the construction above. Then* $u(\alpha v) \le \phi_\alpha(v)$.

Proof. On one hand, if $\|v\| = 1$,

$$\begin{aligned}\phi_\alpha(v) &= \alpha\ell(v) + (1-\alpha)\phi(v)\\ &= \alpha\left(f(y) + g^T(1-\theta)v + \frac{\mu}{2}(1-\theta)^2\right) + (1-\alpha)\phi^*.\end{aligned}$$

Using the assumption that $\phi^* \ge \ell(0) = f(y) - \theta g^T v + \theta^2\frac{\mu}{2}$ and rearranging,

$$\phi_\alpha(v) \ge f(y) + ((1-\theta)\alpha - \theta(1-\alpha))\, g^T v + \frac{\mu}{2}\left(\alpha(1-\theta)^2 + (1-\alpha)\theta^2\right),$$

and simplifying,

$$\phi_\alpha(v) \ge f(y) + (\alpha-\theta)g^T v + \frac{\mu}{2}(\alpha - 2\alpha\theta + \theta^2). \tag{2.4}$$

On the other hand

$$u(\alpha v) = f(y) + (\alpha-\theta)g^T v + \frac{L}{2}(\alpha-\theta)^2. \tag{2.5}$$

Let us simplify the last term, using (2.2),

$$\frac{L}{2}(\alpha-\theta)(\alpha-\theta) = \frac{\mu}{2}(1-\theta)(\alpha-\theta).$$

It follows that

$$u(\alpha v) = f(y) + (\alpha-\theta)g^T v + \frac{\mu}{2}(\alpha - \alpha\theta + \theta^2 - \theta).$$

Subtracting this from (2.4), we obtain immediately

$$\phi_\alpha(v) - u(\alpha v) \ge \frac{\mu}{2}\theta(1-\alpha) \ge 0$$

because $\theta, \alpha \in [0,1]$. If $x = y = v = 0$, then $\ell(0) = f(0) = u(0)$, and the result is straightforward, completing the proof. $\square$

Lemma 2.3. *Assume that (H2) holds for the construction above. Then* $u(\tilde{x}) \le \phi_\alpha(\tilde{v})$.

Proof. Since $\tilde{x}$ and $\tilde{v}$ are respectively global minimizers of $u(\cdot)$ and $\phi_\alpha(\cdot)$, we have for all $x \in \mathbb{R}^n$,

$$u(x) = u(\tilde{x}) + \frac{L}{2}\|x-\tilde{x}\|^2, \qquad \phi_\alpha(x) = \phi_\alpha(\tilde{v}) + \frac{\gamma_+}{2}\|x-\tilde{v}\|^2.$$

We already know from the last lemma that $u(\alpha v) \le \phi_\alpha(v)$. Now we only need to show that

$$u(\alpha v) - u(\tilde{x}) = \phi_\alpha(v) - \phi_\alpha(\tilde{v}).$$

The construction is shown in Fig. 2.2: since $\tilde{x} = \alpha\tilde{v}$,

$$\alpha v - \tilde{x} = \alpha(v - \tilde{v}).$$

$$u(\alpha v) - u(\tilde{x}) = \frac{L\alpha^2}{2}\|v - \tilde{v}\|^2 = \frac{\gamma_+}{2}\|v - \tilde{v}\|^2 = \phi_\alpha(v) - \phi_\alpha(\tilde{v}),$$

completing the proof. □

This completes the treatment of the unconstrained case: we know that $\phi_\alpha(\tilde{v}) \geq u(\tilde{x})$ and $\tilde{x} = \alpha\tilde{v}$. We are ready for the main result.

An iteration for the constrained problem:
The geometry of an iteration for the constrained problem is shown in Fig. 2.3 for the cases $\mu = 0$ (left) and $\mu > 0$ (right). The second figure also shows the position of the point z which will be used in the complexity analysis. This point corresponds to what would be used by the generalization of the Auslender-Teboulle method [1] done by Rossetto [9].

Remark: In an algorithm for the unconstrained problem, the new simple function at iteration $k+1$ will be $\phi_{k+1} = \phi_\alpha$. Now ϕ_α is minimized in Ω to obtain v_+ and $\phi_+^* = \phi_\alpha(v_+)$ by projecting $\tilde{v}$ into Ω. The new simple function will differ from ϕ_α: it will be the simple function with minimizer v_+ and Hessian $\gamma_+ I$. We shall prove that this simple function lays below ϕ_α in the set Ω, ensuring the desired decrease in the simple function.

Consider an iteration of Algorithm 1, in the simplified setting made above. Define
$x_+ = \operatorname*{argmin}_{x\in\Omega} u(x) = P_\Omega(\tilde{x})$, $\quad v_+ = \operatorname*{argmin}_{x\in\Omega} \phi_\alpha(x) = P_\Omega(\tilde{v})$,
$\phi_+^* = \phi_\alpha(v_+)$, $\quad x \mapsto \phi_+(x) = \phi_+^* + \frac{\gamma_+}{2}\|x - v_+\|^2$.

THEOREM 2.4. *For the construction above,*
(i) $f(x_+) \leq \phi_+^*$.
(ii) For all $x \in \Omega$, $\phi_+(x) - f(x) \leq (1-\alpha)\,(\phi(x) - f(x))$, *or equivalently,* $\phi_+(x) - f(x) \leq \frac{\gamma_+ - \mu}{\gamma - \mu}\,(\phi(x) - f(x))$.

Proof. (i) By definition, $v_+ = P_\Omega(\tilde{v}) \in \Omega$. Define $z = \alpha v_+$. Then $z \in \Omega$, because $0 \in \Omega$ and $v_+ \in \Omega$. By definition, $u(x_+) \leq u(z)$. Then

$$f(x_+) \leq u(x_+) \leq u(\tilde{x}) + \frac{L}{2}\|z - \tilde{x}\|^2.$$

By Lemma 2.3, $u(\tilde{x}) \leq \phi_\alpha(\tilde{v})$ and by Lemma 2.1, $\tilde{x} = \alpha\tilde{v}$. It follows that

$$f(x_+) \leq \phi_\alpha(\tilde{v}) + \frac{L\alpha^2}{2}\|v_+ - \tilde{v}\|^2 = \phi_\alpha(\tilde{v}) + \frac{\gamma_+}{2}\|v_+ - \tilde{v}\|^2 = \phi_\alpha(v_+) = \phi_+^*,$$

proving (i).

(ii) By construction, for all $x \in \mathbb{R}^n$, $\phi_\alpha(x) = \alpha\ell(x) + (1-\alpha)\phi(x) \leq \alpha f(x) + (1-\alpha)\phi(x)$, or equivalently,

$$\phi_\alpha(x) - f(x) \leq (1-\alpha)(\phi(x) - f(x)). \tag{2.6}$$

By a direct application of Lemma 1.1, we conclude that for all $x \in \Omega$, $\phi_+(x) \leq \phi_\alpha(x)$. Introducing this in (2.6), we obtain

$$\phi_+(x) - f(x) \leq (1-\alpha)(\phi(x) - f(x)).$$

The equivalent formulation follows from the definition of $\gamma_+ = \alpha\mu + (1-\alpha)\gamma$, from which we get $\gamma_+ - \mu = (1-\alpha)(\gamma - \mu)$, completing the proof. □

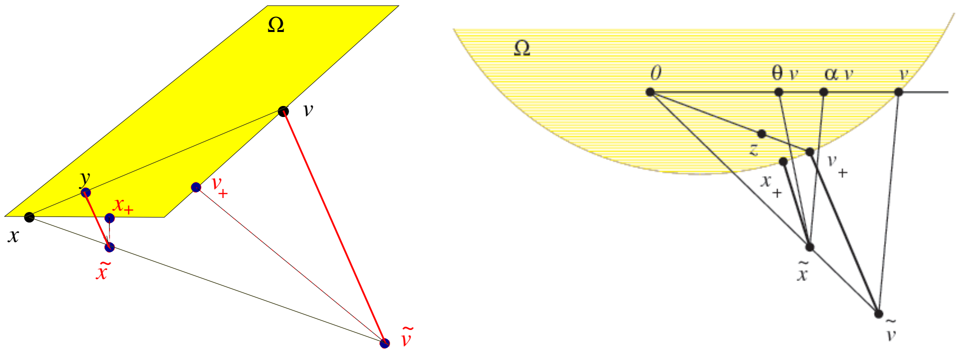

FIG. 2.3. *An iteration for the constrained problem with $\mu = 0$ and $\mu > 0$, respectively.*

2.2. Complexity. The following lemma was proved by Nesterov [7, p.77] with a different notation:

LEMMA 2.5. *Let $\zeta_0, \zeta_1, \dots$ be a sequence satisfying $\zeta_0 > 0$ and for $k \in \mathbb{N}$*

$$\zeta_{k+1} = (1 - \alpha_k)\zeta_k, \qquad L\alpha_k^2 \geq \zeta_{k+1}.$$

Then, for $k \in \mathbb{N}$, $\zeta_k \leq 4L/k^2$.

Proof. See [4, Lemma 10]. The original proof assumes that $L\alpha_k^2 = \zeta_{k+1}$, but the result remains trivially true with the inequality in our statement. □

THEOREM 2.6. *Consider the sequences generated by Algorithm 1 and assume that x^* is an optimal solution. Then for $k \in \mathbb{N}$,*

(i) $(\gamma_k - \mu) \leq \dfrac{4L}{k^2}$.

(ii) $f(x_k) - f(x^*) \leq \dfrac{4L}{(\gamma_0 - \mu)k^2}\left(f(x_0) - f(x^*) + \dfrac{\gamma_0}{2}\|x^* - x_0\|^2\right)$.

Proof. (i) The algorithm sets at iteration k, $\gamma_{k+1} - \mu = (1 - \alpha_k)(\gamma_k - \mu)$. Define for $k \in \mathbb{N}$, $\zeta_k = \gamma_k - \mu$. Then $\zeta_{k+1} = (1 - \alpha_k)\zeta_k$. Also by construction $L\alpha_k^2 = \gamma_{k+1} \geq \gamma_{k+1} - \mu = \zeta_{k+1}$. The result follows directly from Lemma 2.5.
(ii) From Theorem 2.4, for $k \in \mathbb{N}$, $x \in \Omega$,

$$\phi_{k+1}(x) - f(x) \leq \frac{\gamma_{k+1} - \mu}{\gamma_k - \mu}(\phi_k(x) - f(x)).$$

By recursion, we get

$$\phi_k(x) - f(x) \leq \frac{\gamma_k - \mu}{\gamma_0 - \mu}(\phi_0(x) - f(x)). \tag{2.7}$$

Substituting $x = x^*$ in (2.7), noting that $f(x_k) \leq \phi_k^* \leq \phi_k(x^*)$ and using (i),

$$\begin{aligned} f(x_k) - f(x^*) &\leq \frac{4L}{(\gamma_0 - \mu)k^2}(\phi_0(x^*) - f(x^*)) \\ &= \frac{4L}{(\gamma_0 - \mu)k^2}\left(f(x_0) + \frac{\gamma_0}{2}\|x^* - x_0\|^2 - f(x^*)\right), \end{aligned}$$

completing the proof. □

Remark: a good choice for the first step is $\gamma_0 = L$ or $\gamma_0 = L+\mu$. If one chooses $\gamma_0 > L$, it is easy to see that in the first iteration, $\gamma_1 \leq L$. In fact; $(\gamma_1-\mu) = (1-\alpha)(\gamma_0-\mu)$, $\alpha^2 = \dfrac{\gamma_1}{L}$. If $\gamma_1 > L$, $(\gamma_1 - \mu) < 0$, which is impossible. With the choice $\gamma_0 = L + \mu$, the result becomes

$$f(x_k) - f(x^*) \leq \frac{4}{k^2}\left(f(x_0) - f(x^*) + \frac{L+\mu}{2}\|x^* - x_0\|^2\right).$$

Remark: The term $f(x_0) - f(x^*)$ might be removed by using the inequality

$$f(x_0) \leq f(x^*) + \nabla f(x^*)^T(x_0 - x^*) + \frac{L}{2}\|x^* - x_0\|^2,$$

but for constrained problems $\nabla f(x^*) \neq 0$ in general. In the unconstrained case, of course, $\nabla f(x^*) = 0$, which simplifies the expression.

COROLLARY 2.7. *With the hypotheses of Theorem 2.6, it is also true that*

$$f(x_k) - f(x^*) \leq \left(1 - \sqrt{\frac{\mu}{L}}\right)^k \left(f(x_0) - f(x^*) + \frac{\gamma_0}{2}\|x^* - x_0\|^2\right).$$

Proof. We have

$$\gamma_{k+1} - \mu = (1 - \alpha_k)(\gamma_k - \mu).$$

As $\alpha_k^2 = \dfrac{\gamma_{k+1}}{L} \geq \dfrac{\mu}{L}$, we conclude that

$$\gamma_k - \mu \leq \left(1 - \sqrt{\frac{\mu}{L}}\right)^k (\gamma_0 - \mu).$$

The result follows as in the theorem by substitution of this into (2.7), completing the proof. □

This concludes the complexity analysis of the basic method, showing that it is an optimal algorithm. We hope to have unveiled the geometrical aspects of each iteration. All the action happens in a three-dimensional affine space determined by the points x_k, v_k, the gradient vector g (which define the two-dimensional space in which the unconstrained iteration would work), and v_{k+1}. Only Lipschitz and strong convexity constants for the function restricted to this affine subspace would be needed, and this fact will be useful in the next section.

3. Enhanced algorithms.

The basic algorithm presented above is a short steps method. Each gradient step uses the guaranteed descent step length $1/L$, and the constant γ_k is updated by $(\gamma_{k+1} - \mu) = (1 - \alpha_N)(\gamma_k - \mu)$, where α_N satisfies $L\alpha_N^2 = \gamma_{k+1}$, according to the theory.

A more efficient algorithm may be obtained by trying to improve these parameters: the step length may be increased in the gradient step by a line search, and this may be followed by an increase in the constant α, also by a line search. Increasing α increases the speed with which $\gamma_k \to \mu$, which is directly related to the speed of convergence. Also, y_k may be moved along the direction $v_k - x_k$ whenever this is a descent direction, and this also leads to a possible increase in α.

Our algorithm will use these improvements. We state the algorithm and then comment on each of the enhancements. Let L^* and μ be respectively a Lipschitz constant for $\nabla f(\cdot)$ and a strong convexity constant for f, both unknown.

Algorithm 2. *Enhanced Optimal Algorithm*

```
Data: $x_0 \in \Omega$, $v_0 = x_0$, $\mu > 0$ (convexity constant),
    $L > \mu$ (estimated Lipschitz constant), $\gamma_0 = L$, $\phi_0^* = f(x_0)$
$k = 0$
REPEAT
    $d = v_k - x_k$
    Compute $\alpha_N$ as the largest root of $L\alpha^2 = \alpha\mu + (1-\alpha)\gamma_k$
    Set $\theta_N = \frac{\gamma_k}{\gamma_k + \mu\alpha_N}\alpha_N$
    IF $f(x_k + \theta_N d) < f(x_k)$
        Compute $\theta \in [\theta_N, 1]$ such that $f(x_k + \theta d) \leq f(x_k)$
        Set $x_k = x_k + \frac{\theta - \theta_N}{1 - \theta_N} d.$
        $\theta_N = \theta$
        $d = v_k - x_k$
    $y_N = x_k + \theta_N d$, $g = \nabla f(y_N)$
    Define $u(x) = f(y_N) + g^T(x - y_N) + \frac{L}{2}\|x - y_N\|^2$
    Compute $x_N = P_\Omega\left(y_N - \frac{1}{L}g\right)$
    IF $f(x_N) > u(x_N)$, set $L = 2L$ and restart the iteration
    Projected line search
    Compute $\lambda \geq \frac{1}{L}$ such that $f\left(P_\Omega(y_N - \lambda g)\right) \leq f(x_N)$
    Set $x_{k+1} = P_\Omega(y_N - \lambda g)$
    Updating $\phi(\cdot)$
        Define
        $\ell(x) = f(y_N) + \nabla f(y_N)^T(x - y_N) + \mu\|x - y_N\|^2/2$,
        $\phi_\alpha(x) = \alpha\ell(x) + (1-\alpha)\phi_k(x)$.
    Compute a value $\alpha \geq 0$, with $\alpha \geq \alpha_N$, such that
        $\min_{x\in\Omega} \phi_\alpha(x) \doteq \phi_\alpha\left(P_\Omega\left(v_k - \frac{\alpha}{\alpha\mu + (1-\alpha)\gamma_k}\nabla\ell(v_k)\right)\right) \geq f(x_{k+1})$
    Updates
        $\alpha_k = \alpha$
        $\gamma_{k+1} = \alpha_k\mu + (1-\alpha_k)\gamma_k$
        $v_{k+1} = P_\Omega\left(v_k - \frac{\alpha_k}{\gamma_{k+1}}(g + \mu(v_k - y_N))\right)$
        $\phi_{k+1}^* = \phi_\alpha(v_{k+1})$
        $k = k + 1$.
```

Discussion

First enhancement: the gradient step. It is clear that by choosing x_{k+1} such that $f(x_{k+1}) \leq f(x_N)$ the analysis made in Sec. 2.2 keeps unchanged. A step length $\lambda \neq 1/L$ may be computed by a projected gradient search of the Armijo type, following Bertsekas [2].

Since x_N has already been computed, one can start a search method with the step length $\lambda = 1/L$ and then increase it. Instead of this, we started the projected Armijo search with a larger step and backtracked as usual, setting $\lambda = \beta\lambda$, with $\beta \in (0,1)$. The number of steps may be limited to a number q by setting the initial step so that $\lambda\beta^q = \dfrac{1}{L}$.

Second enhancement:

The second enhancement consists in moving the point x_k along the direction $v_k - x_k$, when this is a descent direction. If $f(v_k) \le f(x_k)$, then we set $x_k = v_k$, reducing the objective function and simplifying the iteration. This is equivalent to a reinitialization of the method at v_k, with the initial value of γ equal to γ_k.

Otherwise, we try to move x_k so that the resulting y_N satisfies $f(y_N) \le f(x_k)$.

This is justified as follows: let $\theta \in [\theta_N, 1]$ be such that $f(x_k + \theta d) \le f(x_k)$, and assume that $f(x_k + \theta_N d) \le f(x_k)$, where $d = (v_k - x_k)$. By convexity, this inequality holds for any $z = x_k + \xi d$, $\xi \in [0, \theta_N]$, and we may choose ξ such that

$$\frac{\theta - \xi}{1 - \xi} = \theta_N \qquad \text{or} \qquad \xi = \frac{\theta - \theta_N}{1 - \theta_N}.$$

By starting the iteration at z instead of x_k (which is justified because $f(z) \le f(x_k)$), we obtain $y = x_k + \theta d = z + \theta_N(v_k - z)$, and the analysis in Sec. 2 holds. The virtual change $x_k = x_k + \xi(v_k - x_k)$ does not have to be implemented (unless in the iterations in which L increases), because x_k plays no role in the iteration of the algorithm.

Third enhancement: computation of α.

Note that for a given value of α,

$$\operatorname*{argmin}_{x \in \Omega} \phi_\alpha(x) = P_\Omega\left(v_k - \left(\nabla^2 \phi_\alpha(v)\right)^{-1} \nabla \phi_\alpha(v)\right),$$

with $\nabla^2 \phi_\alpha(v) = (\alpha\mu + (1-\alpha)\gamma_k)\, I$ and $\nabla\phi_\alpha(v) = \alpha \nabla \ell(v)$. Then

$$\operatorname*{argmin}_{x \in \Omega} \phi_\alpha(x) = P_\Omega\left(v_k - \frac{\alpha}{\alpha\mu + (1-\alpha)\gamma_k} \nabla \ell(v_k)\right) = P_\Omega(v_k - \zeta \nabla \ell(v_k)),$$

with $\zeta = \dfrac{\alpha}{\alpha\mu + (1-\alpha)\gamma_k}$.

So, what we need to choose α is a projected line search like the one made in the gradient step. Our ideal goal would be find α such that $\phi_{k+1}(v_{k+1}) = f(x_{k+1})$. This can be computed exactly by a second order equation in the unconstrained case [4, Lemma 6], but here it must be done by a search.

The search may start by setting $\alpha = \alpha_N$, and then trying to increase this value.

Routine 1. *For computing* $\alpha \in [0,1]$.

```
$\alpha = \alpha_N$, $\nabla \ell(v_k) = \nabla f(y_k) + \mu(1-\theta)(v_k - x_k)$, $\beta \in (0,1)$,
$\delta \in (0,1)$ (say, $\beta = 0.5$, $\delta = 0.2$).
Compute $v_+ = P_\Omega\left(v_k - \frac{\alpha}{\alpha\mu + (1-\alpha)\gamma_k}\nabla \ell(v_k)\right)$, $\phi_+^* = \phi_\alpha(v_+)$
$\tilde\alpha = \alpha$
WHILE $\phi_+^* \ge f(x_{k+1})$
    $\phi_{k+1}^* = \phi_+^*$, $v_{k+1} = v_+$, $\alpha = \tilde\alpha$
    $\tilde\alpha = \alpha + \delta(1-\alpha)$
    $v_+ = P_\Omega\left(v_k - \frac{\tilde\alpha}{\tilde\alpha\mu + (1-\tilde\alpha)\gamma_k}\nabla \ell(v_k)\right)$, $\phi_+^* = \phi_{\tilde\alpha}(v_+)$
END
```

At the end of this algorithm, we always have $\phi^*_{k+1} \geq f(x_{k+1})$.

Complexity of the enhanced algorithm: The enhancements in iteration k either decrease the values of $f(x_k)$ or $f(x_{k+1})$, or increase the value of α_k, keeping the property $\phi^*_k \geq f(x_k)$, and hence the complexity analysis made in Sec. 2 holds unchanged.

Remarks: The actual value of L does not have much influence on the number of iterations of the algorithm, due to the line searches and the displacement of y_k. If L is low, the value of α_N increases, and time may be saved in all searches, but this does not seem to be relevant. So we never decrease L.

The possible displacement of x_k at each iteration also profits from a reduction of f in the direction $(v_k - x_k)$.

A practical algorithm: The optimality of the algorithms above could only be proved if each iteration takes the steepest descent step from a point $y = x_k + \theta(v_k - x_k)$ with $\theta \geq \theta_N$. This frequently leads to iterations in which the objective function increases, and we noted that the practical performance of the method improves (for our limited set of tests) if we choose θ as an approximate minimizer of $f(x_k + \theta(v_k - x_k))$ in $[0, 1]$. With this choice we loose optimality (see comments below), and both line searches for the computation of x_{k+1} and α in iteration k must allow for steps respectively smaller than $1/L$ and α_N.

Algorithm 3. *Practical Algorithm*

```
Data: $x_0 \in \Omega$, $v_0 = x_0$, $\mu > 0$ (convexity constant),
    $L > \mu$ (estimated Lipschitz constant), $\gamma_0 = L$, $\phi^*_0 = f(x_0)$
$k = 0$
REPEAT
    $d = v_k - x_k$
    Compute $\theta \in [0,1]$ by an approximate minimization of $f(x_k + \rho(v_k - x_k))$
    for $\rho \in [0,1]$.
    $y_k = x_k + \theta d$, $g = \nabla f(y_k)$
    **Projected Armijo search:** find $\lambda \in [0,1]$ such that
        $f(P_\Omega(y_k - \lambda g)) \leq f(y_k)$
    $x_{k+1} = P_\Omega(y_k - \lambda g)$
    **Updating** $\phi(\cdot)$
        *Define*
        $\ell(x) = f(y_k) + \nabla f(y_k)^T(x - y_k) + \mu\|x - y_k\|^2/2$,
        $u(x) = f(y_k) + \nabla f(y_k)^T(x - y_k) + L\|x - y_k\|^2/2$,
        $\phi_\alpha(x) = \alpha\ell(x) + (1-\alpha)\phi_k(x)$.
    Compute an approximate maximizer of $\alpha \in (0,1)$ such that
        $\min_{x\in\Omega} \phi_\alpha(x) \doteq \phi_\alpha\left(P_\Omega\left(v_k - \frac{\alpha}{\alpha\mu + (1-\alpha)\gamma_k}\nabla\ell(v_k)\right)\right) \geq f(x_{k+1})$
    IF $f(x_{k+1}) > u(x_{k+1})$, set $L = 10L$
    **Updates**
        $\alpha_k = \alpha$
        $\gamma_{k+1} = \alpha_k\mu + (1-\alpha_k)\gamma_k$
        $v_{k+1} = P_\Omega\left(v_k - \frac{\alpha_k}{\gamma_{k+1}}(g + \mu(v_k - y_k)\right)$
        $\phi^*_{k+1} = \phi_\alpha(v_{k+1})$
        $k = k + 1$.
```

Comments: Although the constant L is not used by this algorithm, we included its update. The method is not proved to be optimal, but it can be made optimal by one of two procedures: first, by alternating iterations of Algorithms 2 and 3; second by introducing a safeguard consisting in the computation of an iteration of the basic Algorithm 1, and choosing the best between the steps generated by the practical and the basic algorithm.

The safeguard doubles the number of gradient computations, and did not improve significantly the total number of iterations in our tests. Note that the safeguard is only relevant in the iterations in which $\theta < \theta_N$, because otherwise Algorithms 2 and 3 do essentially the same thing.

4. Numerical Results. In this section, we report the results of a limited set of computational experiments.

The codes are written in MATLAB and we used as stopping criterion the norm of the projected gradient. Our comparisons will be based on the number of iterations. We compare the performance of the following algorithms:

• [Nesterov]	Algorithm proposed by Nesterov in [8].
• [Basic]	Algorithm 1.
• [Enhanced]	Algorithm 2.
• [Practical]	Algorithm 3.

The first three are the only optimal methods that in our knowledge are capable of treating problems with unknown L.

We also compared the methods with the scheme proposed by Nesterov in [7, p. 76] and with a generalization of the interior point method by Auslender and Teboulle [1], made by Rossetto [9]. Both these methods require the knowledge of L, and have a performance similar to the basic algorithm (also using a fixed value for L). We do not include these comparisons here.

We tested the problem of minimizing a quadratic function in a box:

$$\begin{array}{rl} \text{minimize} & f(x) = (g^*)^T(x - x^*) + \frac{1}{2}(x - x^*)^T Q(x - x^*) \\ \text{subject to} & x \in \Omega = \{x \in \mathbb{R}^n \mid 0 \le x \le U\} \end{array}$$

The box is constructed by generating the vector U with random values in $[0, 1]$. Then we generate a random vector $z \in \mathbb{R}^n$ and define the optimal solution as $x^* = P_\Omega z$.

For the function, n, $\mu = 1$ and $L > \mu$ are given. Using MATLAB routines, we generate a random $n \times n$ matrix Q with eigenvalues in the interval $[\mu, L]$ and a random gradient vector g^* at x^*, satisfying the optimality condition $P_\Omega(x^* + g^*) = x^*$.

For each experiment, we generate the problem as above and a random initial point $x_0 \in \Omega$.

Figure 4.1 shows the function values (left) and the projected gradient norms (right) for the problem above with $L = 10000$, $\mu = 1$ and $n = 5000$.

We generated 120 instances of problems with n assuming values from 5 to 6000; L/μ from 100 to 10000 which were solved by all methods. The work per iteration is in the average the following:
Algorithm 1 computes 1 gradient and 2 function values.
Algorithm 2 computes 2 gradients and 7 function values.
Algorithm 3 computes 1 gradient and 6.8 function values.
Nesterov's algorithm computes 4 gradients and no function values.

Figure 4.2 shows the performance profiles [3] of the methods for the number of iterations (left) and for a laboriousness measure defined by the number of function evaluations plus three times the number of gradient evaluations (right).

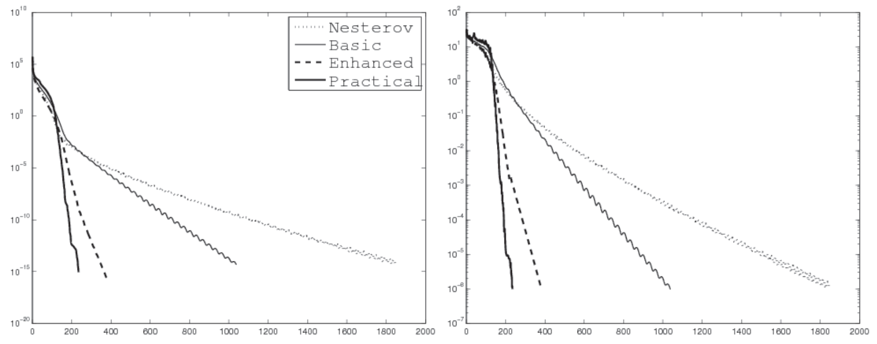

FIG. 4.1. *Variation of the function values (left) and of the norm of the projected gradient (right).*

The Practical Algorithm performs very well. It is the best for 62% of the problems and it solves all problems using no more than three times the number of the iterations of the best. The Basic Algorithm solves 99% problems using no more than eight times the number of iterations of the best. As its iterations are cheaper than for the enhanced algorithms, Basic Algorithm solves 90% of the problems using no more than twice the laboriousness of the best.

Our algorithms perform clearly better than Nesterov's method. Although this method computes no function values, it computes in the average 4 gradients per iteration.

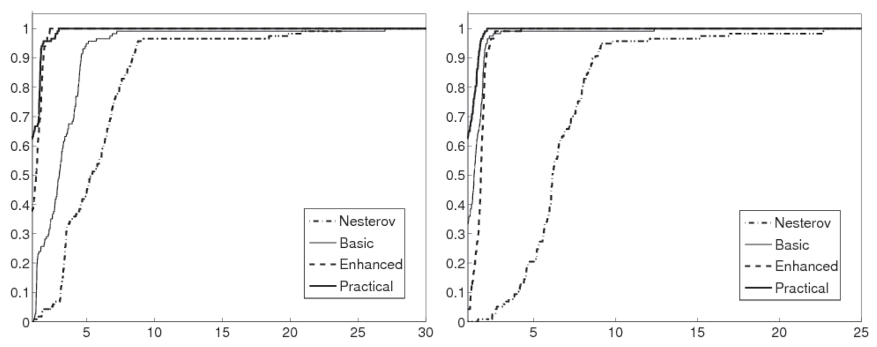

FIG. 4.2. *Performance profile for number of iterations (left) and evaluations (right).*

REFERENCES

[1] A. Auslender and M. Teboulle. Interior gradient and proximal methods for convex and conic optimization. *SIAM Journal on Optimization*, 16(3):697–725, 2006.
[2] D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, USA, 1995.
[3] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.
[4] C. C. Gonzaga and E. W. Karas. Optimal steepest descent algorithms for unconstrained convex problems: Fine tuning Nesterov's method. Technical report, Dep. Mathematics, Federal University of Paraná, Brazil, 2008.
[5] G. Lan, Z. Lu, and R.D.C. Monteiro. Primal-dual first-order methods with $O(1/\epsilon)$ iteration-complexity for cone programming. Technical report, School of ISyE, Georgia Tech, USA, 2006. Accepted in Mathematical Programming.

[6] A. S. Nemirovski and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley, New York, 1983.
[7] Y. Nesterov. *Introductory Lectures on Convex Optimization. A basic course.* Kluwer Academic Publishers, Boston, 2004.
[8] Y. Nesterov. Gradient methods for minimizing composite objective function. Discussion paper 76, CORE, UCL, Belgium, 2007.
[9] D. R. Rossetto. *Tópicos em métodos ótimos para otimização convexa.* PhD thesis, Department of Applied Mathematics, University of São Paulo, São Paulo, Brazil, 2011. In portuguese.