Bulletin of the Iranian Mathematical Society (2019) 45:883–910
https://doi.org/10.1007/s41980-018-0171-1

ORIGINAL PAPER

An Optimal Subgradient Algorithm with Subspace Search for Costly Convex Optimization Problems

Masoud Ahookhosh[1] · **Arnold Neumaier**[1]

Received: 21 January 2018 / Revised: 25 September 2018 / Accepted: 28 September 2018 /
Published online: 10 October 2018

Abstract
This paper presents an acceleration of the optimal subgradient algorithm OSGA (Neumaier in Math Program 158(1–2):1–21, 2016) for solving structured convex optimization problems, where the objective function involves costly affine and cheap nonlinear terms. We combine OSGA with a multidimensional subspace search technique, which leads to a low-dimensional auxiliary problem that can be solved efficiently. Numerical results concerning some applications are reported. A software package implementing the new method is available.

Keywords Convex optimization · Nonsmooth optimization · Subgradient methods · Multidimensional subspace search · Optimal complexity · First-order black-box information · Costly linear operator

Mathematics Subject Classification 90C25 · 90C60 · 49M37 · 65K05 · 68Q25

1 Introduction

Over the past few decades, solving convex optimization with smooth or nonsmooth objectives has received much attention due to many applications in the fields of applied sciences and engineering, cf. [15,50]. For smooth problems, first- and second-order information is typically available and many first- and second-order methods exist, see [37,44]. However, for nonsmooth problems, usually only first-order information is available. Solving nonsmooth problems is commonly harder than solving smooth problems; however, there are many nonsmooth problems with nice structure such that

Communicated by Majid Soleimani-damaneh.

✉ Masoud Ahookhosh
masoud.ahookhosh@univie.ac.at

Arnold Neumaier
Arnold.Neumaier@univie.ac.at

[1] Faculty of Mathematics, University of Vienna, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

this structure can be used to design efficient methodologies for them. Because of the low memory requirement, first-order methods are especially important for problems with a large number of variables.

Subgradient methods constitute a class of first-order methods that have been developed since 1960 to solve convex nonsmooth optimization problems, see, e.g., [46,49]. In general, they only need function values and subgradients, have low memory requirement, and can be used for solving convex optimization problems with several millions of variables. However, too many iterations are needed to attain a very accurate solution. The low convergence speed of subgradient methods corresponds to their complexity (the number of iterations required to attain an ε-solution for a given $\varepsilon > 0$). In 1983, Nemirovski and Yudin [36] proved that the worst-case complexity bound to achieve an ε-solution of problems with a Lipschitz continuous convex nonsmooth objective by first-order methods is $\mathcal{O}(\varepsilon^{-2})$, while it is $\mathcal{O}(\varepsilon^{-1/2})$ for smooth problems with Lipschitz continuous gradients.

Algorithms attaining the optimal worst-case complexity bound for a class of problems are called optimal. Historically, optimal first-order methods for smooth convex optimization date back to Nesterov [38] in 1983. He later in [40,41] proposed two gradient-type methods for minimizing a sum of two functions (composite problems) with the optimal complexity, where, for the first method, the smooth part of the objective needs to have Lipschitz continuous gradients and, for the second one, the smooth part of the objective needs to have Hölder continuous gradients. Since 1983 many researchers have studied optimal first-order methods; see, e.g., Auslander and Teboulle [7], Beck and Teboulle [11], Devolder et al. [22], Gonzaga et al. [26,27], Lan [30], Lan et al. [31], Nesterov [37,39,40], and Tseng [52]. Moreover, Nemirovski and Yudin in [36] showed that the subgradient, subgradient projection, and mirror descent methods attain the complexity $\mathcal{O}(\varepsilon^{-2})$ for Lipschitz continuous nonsmooth objectives, so that they are optimal for this class of problems. Recently, Neumaier in [42] proposed a subgradient algorithm called OSGA, which attains both the optimal complexity $\mathcal{O}(\varepsilon^{-1/2})$ for smooth problems with Lipschitz continuous gradients and the optimal complexity $\mathcal{O}(\varepsilon^{-2})$ for Lipschitz continuous nonsmooth problems. It is notable that OSGA does not need to know about global information of objective functions such as Lipschitz constants and behaves well for problems arising in applications, see Ahookhosh and Neumaier [1,4–6].

A multidimensional subspace search scheme is a generalization of line search techniques, which are one-dimensional search schemes for finding a step-size along a specific direction. Hence, in multidimensional subspace search, one searches a vector of step-sizes allowing the best combination of several search directions for optimizing an objective function. Generally, subspace search techniques form a class of descent methods, where they can be used independently or employed as an accelerator inside of iterative schemes to attain a faster convergence. The pioneering work of subspace optimization was proposed in 1969 for smooth problems by Miele and Cantrell [33] and Cragg and Levy [21] who defined a memory gradient technique based on a subspace of the form $\mathcal{S} = \mathrm{span}\{-g_k, d_{k-1}\}$, where g_k denotes the gradient of the function at x_k and d_{k-1} is the last available direction. Since then, many subspace search schemes have been proposed by selecting various search directions, see, e.g., [18,20,51] and references therein. Depending on the selected search directions used for constructing a subspace, two classes of subspace methods are distinguished, namely, gradient-type techniques [21,23,34] and Newton-type schemes [28,32,53,54].

Content In this paper we propose an accelerated version of OSGA (called OSGA-S) for solving convex optimization problems involving costly linear operators and cheap nonlinear terms. Our new method is a two-stage method that solves unconstrained nonsmooth convex optimization problems of the form

$$\begin{array}{ll} \min & f(x) := \sum_{i=1}^{p} f_i(\mathcal{A}_i x) \\ \text{s.t.} & x \in \mathcal{V}, \end{array} \tag{1}$$

where for $i = 1, \ldots, p$ $(p \ll n)$, $f_i : \mathcal{U}_i \to \mathbb{R}$ is a (non)smooth, proper, and convex function, and $\mathcal{A}_i : \mathcal{V} \to \mathcal{U}_i$ is a linear operator, for real finite-dimensional vector spaces $\mathcal{V}$, $\mathcal{U}_i$. Solving (1) with OSGA involves two key steps, namely, providing the first-order information and solving an auxiliary high-dimensional subproblem (Eq. (5) below). Since the problem (1) is unconstrained, the exact solution of the corresponding auxiliary problem is given in a closed form, cf. [1,42]. In many applications involving overdetermined systems of equations and classification with support vector machine (see Sects. 4.1 and 4.2), the objective function has the form (1) involving costly affine but cheap nonlinear terms. Hence the most costly parts in computing function values and subgradients are related to applying forward and adjoint operators. We therefore try to improve in each iteration the current best point by an inner iteration solving a low-dimensional version of the original problem, using a subspace composed from the best point and the last few iterations. We emphasize that applying the subspace search involves no additional costly forward and adjoint operators. Therefore, the subspace search stage does not impose a significant cost to the outer scheme OSGA while improves its performance considerably. As proved in [42], this does not affect the worst-case complexity of the algorithm if done in the right place. However, our numerical results show that it successfully reduces the number of iterations and the running time needed in practice.

Similar to OSGA, OSGA-S needs to know about no global information except the strong convexity parameter μ ($\mu = 0$ if it is not available), and it only requires the first-order information; however, the main advantage of OSGA-S is being able to handle problems with complex structure of the form (1) involving composition of several functions and linear operators. Such structured problems have received much attention due to increase of interest in using mixed regularization terms, e.g., [8]. However, if f_i, $i = 1, \ldots, p$, are nonsmooth, then smooth solvers, Nesterov-type optimal methods [37,38,40], and proximal splitting methods [11,19] are not able to handle the problem. In this case subgradient methods [14] and Nesterov's universal gradient method with the level of smoothness parameter $\nu = 0$ [41] can deal with the problem. Note that the mirror descent methods [12,36] can only handle the constrained version of (1). On the other hand, to the best of our knowledge there is only little work involving subspace search techniques for nonsmooth optimization problems [23,35], where they are based on smoothing the objective functions so that can not be used in our numerical comparison.

For high-dimensional problems involving dense matrices, applying OSGA-S with a multidimensional subspace search results in a substantial gain in the running time, despite the extra effort needed for applying the subspace optimization. Indeed, the inner level runs OSGA-S only on a low-dimensional unconstrained auxiliary problem in an adaptive multidimensional subspace, and the associated solution is used to accelerate the outer level of OSGA-S iteration on the original problem. The multidimensional subspace uses some previously computed directions and results in a low-dimensional problem with typically at most 20 variables. Numerical experiments and comparison with subgradient methods and the universal gradient method show that the subspace search can significantly accelerate OSGA, especially when the objective involves costly linear operators.

The remainder of this paper is organized as follows. In the next section we briefly review the main idea of OSGA. Section 3 describes a combination of OSGA and a multidimensional subspace search. Numerical results are reported in Sect. 4, and some conclusions are given in Sect. 5.

Notations Let $\mathcal{V}$ be a real finite-dimensional vector space endowed with the norm $\|\cdot\|$, and $\mathcal{V}^*$ denotes its dual space, which is formed by all linear functional on $\mathcal{V}$ where the bilinear pairing $\langle g, x\rangle$ denotes the value of the functional $g \in \mathcal{V}^*$ at $x \in \mathcal{V}$. If $\mathcal{V} = \mathbb{R}^n$, then, for $1 \le p \le \infty$,

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

For a function $f : \mathcal{V} \to \overline{\mathbb{R}} = \mathbb{R} \cup \{\pm\infty\}$,

$$\operatorname{dom} f = \{x \in \mathcal{V} \mid f(x) < +\infty\}$$

denotes its effective domain, and f is called proper if $\operatorname{dom} f \neq \emptyset$ and $f(x) > -\infty$ for all $x \in \mathcal{V}$. The vector $g \in \mathcal{V}^*$ is called a subgradient of f at x if $f(x) \in \mathbb{R}$ and

$$f(y) \ge f(x) + \langle g, y - x\rangle \quad \text{for all } y \in \mathcal{V}.$$

The set of all subgradients is called the subdifferential of f at x denoted by $\partial f(x)$. We denote by f_x and g_x, the function value $f(x)$ and the subgradient g at $x \in C$, respectively.

2 A Review of OSGA

In this section we briefly review the main idea of the optimal subgradient algorithm (see Algorithm 1) proposed by Neumaier in [42] for solving the convex constrained minimization problem

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & x \in C, \end{array} \tag{2}$$

where $f : C \to \mathbb{R}$ is a proper and convex function defined on a nonempty, closed, and convex subset C of $\mathcal{V}$.

OSGA is a subgradient algorithm for problem (2) that uses first-order information, i.e., function values and subgradients, to construct a sequence of iterations $\{x_k\} \in C$ whose sequence of function values $\{f(x_k)\}$ converge to the minimum $\widehat{f} = f(\widehat{x})$ with the optimal complexity. OSGA requires no information regarding global parameters such as Lipschitz constants of function values and gradients. In the unconstrained version relevant for the present work, we have $C = \mathcal{V}$, and we work with a quadratic prox-function $Q(z) := Q_0 + \frac{1}{2}\|z - x_0\|_2^2$, where $x_0 \in \mathcal{V}$ is a given starting point and Q_0 an appropriate positive constant. Let us denote by $g_Q(x)$ the gradient by Q at x. At each iteration, OSGA satisfies the bound

$$0 \le f(x_b) - \widehat{f} \le \eta Q(\widehat{x}) \tag{3}$$

on the currently best function value $f(x_b)$ with a monotonically decreasing error factor η that is guaranteed to converge to zero by an appropriate steplength selection strategy (see Procedure PUS). Note that $\widehat{x}$ is not known, thus the error bound is not fully constructive, but enough to guarantee the convergence of $f(x_b)$ to $\widehat{f}$ with a predictable worst-case complexity. To maintain (3), OSGA considers linear relaxations of f at z,

$$f(z) \ge \gamma + \langle h, z\rangle \qquad \text{for all } z \in C, \tag{4}$$

where $\gamma \in \mathbb{R}$ and $h \in \mathcal{V}^*$, updated using linear underestimators available from the subgradients evaluated (see Algorithm 1). For each such linear relaxation, OSGA solves a maximization problem of the form

$$\begin{array}{rll} E(\gamma, h) := & \max & E_{\gamma,h}(x) \\ & \text{s.t.} & x \in C, \end{array} \tag{5}$$

where

$$E_{\gamma,h}(x) := -\frac{\gamma + \langle h, x\rangle}{Q(x)}. \tag{6}$$

Let $\gamma_b := \gamma - f(x_b)$ and $u := U(\gamma_b, h) \in C$ be a solution of (5). From (4) and (6), we obtain

$$E(\gamma_b, h) \ge -\frac{\gamma - f(x_b) + \langle h, \widehat{x}\rangle}{Q(\widehat{x})} \ge \frac{f(x_b) - \widehat{f}}{Q(\widehat{x})} \ge 0. \tag{7}$$

Setting $\eta := E(\gamma_b, h)$ in (7) implies that (3) is valid. If x_b is not optimal then the right inequality in (7) is strict, and since $Q(z) \ge Q_0 > 0$, we conclude that the maximum η is positive.

In each step, OSGA uses the next scheme for updating the given parameters α, h, γ, η, and u, see [42] for more details.

Procedure PUS(parameters updating scheme)

```
   Input: $\delta,\ \alpha_{\max} \in\ ]0,1[,\ 0<\kappa' \le \kappa, \alpha, \eta, \bar{h}, \bar{\gamma}, \bar{\eta}, \bar{u}$;
   Output: $\alpha,\ h,\ \gamma,\ \eta,\ u$;
1  begin
2      $R \leftarrow (\eta - \overline{\eta})/(\delta\alpha\eta)$;
3      if $R < 1$ then
4          $\overline{\alpha} = \alpha e^{-\kappa}$;
5      else
6          $\overline{\alpha} \leftarrow \min(\alpha e^{\kappa'(R-1)}, \alpha_{\max})$;
7      end
8      $\alpha \leftarrow \overline{\alpha}$;
9      if $\overline{\eta} < \eta$ then
10         $h \leftarrow \overline{h};\ \gamma \leftarrow \overline{\gamma};\ \eta \leftarrow \overline{\eta};\ u \leftarrow \overline{u}$;
11     end
12 end
```

Algorithm 1: OSGA (optimal subgradient algorithm)

```
   Input: $\delta, \alpha_{\max} \in\ ]0,1[,\ 0<\kappa' \le \kappa$; local parameters: $x_0, \mu \ge 0$;
   Output: $x_b,\ f_{x_b}$;
1  begin
2      $x_b = x_0;\ h = g_{x_b} - \mu g_Q(x_b);\ \gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b \rangle$;
3      $\gamma_b = \gamma - f_{x_b};\ u = U(\gamma_b, h);\ \eta = E(\gamma_b, h) - \mu;\ \alpha \leftarrow \alpha_{\max}$;
4      while stopping criteria do not hold do
5          $x = x_b + \alpha(u - x_b);\ g = g_x - \mu g_Q(x);\ \overline{h} = h + \alpha(g - h)$;
6          $\overline{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x\rangle - \gamma)$;
           $x'_b = \operatorname{argmin}_{z\in\{x_b,x\}} f(z);\ f_{x'_b} = \min\{f_{x_b}, f_x\}$;
7          $\gamma'_b = \overline{\gamma} - f_{x'_b};\ u' = U(\gamma'_b, \overline{h}); x' = x_b + \alpha(u' - x_b)$;
8          choose $\overline{x}_b$ in such a way that $f_{\overline{x}_b} \le \min\{f_{x'_b}, f_{x'}\}$;
9          $\overline{\gamma}_b = \overline{\gamma} - f_{\overline{x}_b};\ \overline{u} = U(\overline{\gamma}_b, \overline{h});\ \overline{\eta} = E(\overline{\gamma}_b, \overline{h}) - \mu;\ x_b = \overline{x}_b;\ f_{x_b} = f_{\overline{x}_b}$;
10         update the parameters $\alpha, h, \gamma, \eta$ and $u$ using PUS;
11     end
12 end
```

In [42], it is shown that the number of iterations to achieve an ε-optimum is of the optimal order $\mathcal{O}\left(\varepsilon^{-1/2}\right)$ for a smooth f with Lipschitz continuous gradients and of the order $\mathcal{O}\left(\varepsilon^{-2}\right)$ for a Lipschitz continuous nonsmooth f. The algorithm has low memory requirements so that, if the subproblem (5) can be solved efficiently, OSGA is appropriate for solving large-scale problems. Numerical results reported by Ahookhosh in [1,3] for unconstrained problems, and by Ahookhosh and Neumaier in [4–6] for simply constrained problems show the good behavior of OSGA for solving practical problems.

Note that there is a flexibility in choosing $\overline{x}_b$ in Line 8 of OSGA ($\overline{x}_b \in \{x'_b, x'\}$). In the next section we give a two-stage scheme (called OSGA-S) that the outer stage is OSGA and the inner stage is a multidimensional subspace search used to produce a suitable point $\overline{x}_b$ in Line 8 of OSGA guaranteeing $f_{\overline{x}_b} \leq \min\{f_{x'_b}, f_{x'}\}$ without increasing a significant computational cost to the outer stage OSGA for solving the problem (1).

3 Structured Convex Optimization Problems

In this paper we consider the convex optimization problem (1), which appears in many applications such as signal and image processing, machine learning, statistics, data fitting, and inverse problems; see, e.g., [15,50].

In many applications, the objective function of (1) involves expensive linear mappings (equivalently matrix-vector products with dense matrices). To apply a first-order method for minimizing such problems, the first-order oracle (function values and subgradients) should be available, i.e.,

$$f_x = \sum_{i=1}^{p} f_i(\mathcal{A}_i x), \quad g_x \in \sum_{i=1}^{p} \mathcal{A}_i^* \partial f(\mathcal{A}_i \cdot)(x).$$

Hence, in each call of the first-order oracle, p forward operators $\mathcal{A}_i,\ i = 1, \ldots, p$, and p adjoint operators $\mathcal{A}_i^*,\ i = 1, \ldots, p$, must be applied requiring $\mathcal{O}(n^2)$ operations. This computationally leads to overall expensive function and subgradient evaluations such that the total cost of using a first-order method is dominated by the cost of applying forward and adjoint linear operators. This motivates the quest for developing an acceleration of OSGA using a multidimensional subspace search for solving such problems.

The primary idea of multidimensional subspace methods is to restrict the next iteration to a low-dimensional subspace by constructing a subproblem with a reduced dimension. Let us fix $M \ll n$, where n is the number of variables. Let the sequence $\{x_k\}_{k\geq 0}$ be generated by $x_{k+1} = x_k + d_k$, where d_k is a search direction. We suppose that $d_1, d_2, \ldots, d_M$ are M directions used to span the subspace

$$\mathcal{S} = \operatorname{span}\{d_1, d_2, \ldots, d_M\}. \tag{8}$$

In this case a direction d belongs to the subspace $\mathcal{S}$ if and only if there exist constants $t_1, t_2, \ldots, t_M$ such that

$$d = \sum_{i=1}^{M} t_i d_i = \widetilde{U} t, \tag{9}$$

where $\widetilde{U} := [d_1, d_2, \ldots, d_M]$ is a matrix constructed from the directions considered and $t = (t_1, t_2, \ldots, t_M)^T$ is a vector of coefficients. Afterwards, the M-dimensional minimization problem

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(\mathcal{A}_i(x + \widetilde{U}t)) = \sum_{i=1}^{p} f_i(v_i + V_i t) \\ \text{s.t.} & t \in \mathbb{R}^M \end{array} \tag{10}$$

is considered to determine the best possible vector of coefficients t, where $v_i := \mathcal{A}_i x$ and $V_i := \mathcal{A}_i \widetilde{U}$. The minimization problem (10) shows that the procedure of searching the best possible direction of the form (9) in the subspace (8) generalizes the idea of exact line search, see, e.g., [44], but it provides an approximate minimization. One can also construct the subspace

$$\mathcal{S} = \operatorname{span}\{x_{k-M+1}, x_{k-M+2}, \ldots, x_k\}, \tag{11}$$

and set $\widetilde{U} = (x_{k-M+1}, x_{k-M+2}, \ldots, x_k)$. Then the subspace minimization is defined by

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(\mathcal{A}_i(\widetilde{U}t)) = \sum_{i=1}^{p} f_i(V_i t) \\ \text{s.t.} & t \in \mathbb{R}^M. \end{array} \tag{12}$$

Since $M \ll n$, the minimization subproblems (10) and (12) are low-dimensional and can be solved efficiently by classical optimization methods. Hence subspace search techniques can be implemented extremely fast. This leads to suitable schemes for large-scale optimization as the number of variables of practical problems growing up. Moreover, using a multidimensional subspace search as an inner step of iterative schemes needs low memory, which may be considerably cheaper than performing one step of the algorithm in the full dimension. Further, many common ideas in nonlinear optimization can be considered as multidimensional subspace search techniques, namely conjugate gradient, limited memory quasi-Newton, and memory gradient methods; see, e.g., [18,23,54].

Motivated by the above-mentioned discussion, the multidimensional subspace search scheme can be outlined as follows:

Algorithm 2: MDSS (multidimensional subspace search)

```
   Input: $x_b \in \mathcal{V}$, $\widetilde{U}$, $v_i \in \mathcal{U}_i$, $V_i$ $(i = 1, \ldots, p)$;
   Output: $\overline{x}_b$, $f_{\overline{x}_b}$;
1  begin
2      solve the $M$-dimensional minimization problem (10) or (12) inexactly to find
       $t^*$;
3      set $\overline{x}_b = x_b + \widetilde{U}t^*$ for (10) or $\overline{x}_b = \widetilde{U}t^*$ for (12); $f_{\overline{x}_b} = f(\overline{x}_b)$;
4  end
```

To implement Algorithm 2 successfully, some factors are crucial: (i) the number of directions M controlling the computational cost of the scheme; (ii) choosing suitable directions to construct the subspaces; (iii) solving the minimization problem (10) or (12) efficiently. Indeed, for choosing the number of directions M, there is a trade-off between the total computational cost per iteration and the amount of possible decrease in function values.

We here use MDSS as an accelerator of OSGA for solving problems involving costly linear operators. More precisely, we save some previously computed points, construct a subspace of the form (8) and apply MDSS to find a point $\overline{x}_b$ in Line 8 of OSGA. This typically gives us a better point x_b in Line 9 of OSGA. In the next subsection, we will show how the subspace $\mathcal{S}$ is constructed and how the subproblem (12) can be solved efficiently at a reasonable cost.

3.1 Solving the Auxiliary Problem (12) by OSGA

In this section we show how one can construct a suitable subspace of the form (11) and how to solve the auxiliary problem (12) with OSGA. Without loss of generality, we here assume $\mathcal{A}_i : \mathbb{R}^n \rightarrow \mathbb{R}^m,\ i = 1, \ldots, p$. For $\widetilde{U}, V_i \in \mathbb{R}^{n\times(2M+1)},\ i = 1, \ldots, p$, $\widetilde{U}_{:j}$ and $(V_i)_{:j}$ denote the jth column of the matrices $\widetilde{U}$ and V_i, respectively. Let us consider a variant of OSGA using the multidimensional subspace search technique as follows:

Algorithm 3: OSGA-S (optimal subgradient algorithm with subspace search)

```
   Input: global parameters: $\delta, \alpha_{\max} \in ]0,1[,\ 0 < \kappa' \leq \kappa$; local parameters:
       $x_0, \mu \geq 0$;
   Output: $x_b,\ f_{x_b}$;
 1 begin
 2     $x_b = x_0;\ h = g_{x_b} - \mu g_Q(x_b);\ \gamma = f_{x_b} - \mu Q(x_b) - \langle h, x_b\rangle$;
 3     $\gamma_b = \gamma - f_{x_b};\ u = U(\gamma_b, h);\ \eta = E(\gamma_b, h) - \mu;\ \alpha \leftarrow \alpha_{\max};\ r = 0$; flag $= 1$;
 4     while stopping criteria do not hold do
 5         $x = x_b + \alpha(u - x_b);\ v_i^x = \mathcal{A}_i x, i = 1, \ldots, p; r = r + 1$,
           $\widetilde{U}_{:r} = x;\ (V_i)_{:r} = v_i^x,\ i = 1, \ldots, p$;
 6         $g = g_x - \mu g_Q(x);\ \overline{h} = h + \alpha(g - h)$;
           $\overline{\gamma} = \gamma + \alpha(f_x - \mu Q(x) - \langle g, x\rangle - \gamma)$;
 7         $x_b' = \mathrm{argmin}_{z \in \{x_b, x\}} f(z);\ f_{x_b'} = \min\{f_{x_b}, f_x\}$;
           $\gamma_b' = \overline{\gamma} - f_{x_b'};\ u' = U(\gamma_b', \overline{h})$;
 8         $x' = x_b + \alpha(u' - x_b);\ v_i^{x'} = \mathcal{A}_i(x'), i = 1, \ldots, p; r = r + 1, \widetilde{U}_{:r} = x$;
           $(V_i)_{:r} = v_i^{x'},\ i = 1, \ldots, p$;
 9         if $r = 2M$ then
10             $\widetilde{U}_{:2M+1} = x_b; (V_i)_{:2M+1} = v_i^b,\quad i = 1, \ldots, p$; flag $= 0; r = 1$;
11         end
12         if flag then
13             $f_{\overline{x}_b} = \min\{f_{x_b'}, f_{x'}\};\ \overline{x}_b = \mathrm{argmin}\{f_{x_b'}, f_{x'}\}$
14         else
15             solve (12) with $t \in \mathbb{R}^{2M+1}$ by MDSS routine to find $\overline{x}_b$ and $f_{\overline{x}_b}$;
16         end
17         $\overline{\gamma}_b = \overline{\gamma} - f_{\overline{x}_b}; \overline{u} = U(\overline{\gamma}_b, \overline{h}); \overline{\eta} = E(\overline{\gamma}_b, \overline{h}) - \mu; x_b = \overline{x}_b; f_{x_b} = f_{\overline{x}_b}$;
           $v_i^b = \mathcal{A}_i x_b,\ i = 1, \ldots, p$;
18         update the parameters $\alpha, h, \gamma, \eta$, and $u$ using PUS;
19     end
20 end
```

In OSGA-S if the number of iterations is less than M, we save the points x and x' and related vectors $v_i^x = \mathcal{A}_i x,\ v_i^{x'} = \mathcal{A}_i(x'),\ i = 1, \ldots, p$. These points and the best iteration so far (x_b) are used to construct the subspace

$$\mathcal{S} := \mathrm{span}\{x_{k-M+1}, x_{k-M+1}', \ldots, x_k, x_k', x_b\}. \tag{13}$$

If the number of iterations is larger than or equal to M, we use the subspace (13) and solve a subspace problem of the form (12) with $t \in \mathbb{R}^{2M+1}$, i.e.,

$$\begin{array}{ll} \min & \sum_{i=1}^{p} f_i(V_i t) \\ \text{s.t.} & t \in \mathbb{R}^{2M+1}. \end{array} \tag{14}$$

This possibly leads us to a better point $\overline{x}_b$ than that provided in Line 8 of OSGA. Note that if the number of iterations is bigger or equal than M, OSGA-S is a two-stage algorithm, where the outer stage is OSGA and the inner stage is a subspace search in Line 15. In the next result we show that $f_{\overline{x}_b} \leq \min\{f_{x_b'}, f_{x_k'}\}$ for all $k \geq 0$ of OSGA-S.

Theorem 1 *Let k be the iteration counter of OSGA-S, $k \geq M$, and $\mathcal{A}_i : \mathbb{R}^n \rightarrow \mathbb{R}^m$, $i = 1, \ldots, p$. Let also the points*

$$x_b,\ x_j,\ x_j' \quad \text{for } j = k - M + 1, \ldots, k$$

be generated by the former iterations of OSGA-S to construct the subspace (13). Then each step of OSGA applied to (14) in MDSS needs $4pm(2M+1)$ operations. In step k of OSGA-S, we have

$$f_{\overline{x}_b} \leq \min\{f_{x_b'}, f_{x_k'}\}.$$

Proof Let us define $\widetilde{U}_k \in \mathbb{R}^{n \times (2M+1)}$ by

$$\widetilde{U}_k := \big(x_{k-M+1}, x_{k-M+1}', \ldots, x_k, x_k', x_b\big) \tag{15}$$

and

$$v_i^b := \mathcal{A}_i x_b,\ v_i^j := \mathcal{A}_i x_j,\ (v_i^j)' := \mathcal{A}_i x_j' \text{ for } j = k - M + 1, \ldots, k,\ i = 1, \ldots, p.$$

Then we have

$$\begin{aligned} \mathcal{A}_i(\widetilde{U}_k t) &= (\mathcal{A}_i \widetilde{U}_k) t = \big(\mathcal{A}_i x_{k-M+1}, \mathcal{A}_i x_{k-M+1}', \ldots, \mathcal{A}_i x_k, \mathcal{A}_i x_k', \mathcal{A}_i x_b\big)\, t \\ &= (v_{k-M+1}^j, (v_{k-M+1}^j)', \ldots, v_k^j, (v_k^j)', v_i^b) t = V_{ik} t, \end{aligned} \tag{16}$$

for $i = 1, \ldots, p$, where $V_{ik} := (v_{k-M+1}^j, (v_{k-M+1}^j)', \ldots, v_k^j, (v_k^j)', v_i^b) \in \mathbb{R}^{m \times (2M+1)}$. This means that the construction of V_{ik}, $i = 1, \ldots, p$, has no extra cost if they have been saved in the outer scheme of OSGA-S. We now compute the first-order oracle at t by

$$f_t = \sum_{i=1}^p f_i(V_{ik} t), \quad g_t \in \sum_{i=1}^p V_{ik}^* \partial f_i(V_{ik} t). \tag{17}$$

Computing each of $V_{ik} t$ and $V_{ik}^* \partial f_i(V_{ik} t)$, $i = 1, \ldots, p$, needs $m(2M+1)$ operations. Therefore, apart from the cost of nonlinear terms, we need $2pm(2M+1)$ operations in each call of the first-order oracle for the problem (14). Since OSGA requires two calls of the first-order oracle in each iteration, we need $4pm(2M+1)$ operations in each iteration of MDSS.

By (15) and setting $t := (0, \ldots, 1, 0, 0)^T \in \mathbb{R}^{2M+1}$ (1 in its $(2M-1)$th component), $t' := (0, \ldots, 0, 1, 0)^T \in \mathbb{R}^{2M+1}$ (1 in its $(2M)$th component), and

$t_b := (0, \dots, 0, 0, 1)^T \in \mathbb{R}^{2M+1}$ (1 in its $(2M+1)$th component), we get

$$x_k = \widetilde{U}_k t, \quad x_k' = \widetilde{U}_k t', \quad x_b = \widetilde{U}_k t_b.$$

This implies that $x_k, x_k', x_b \in \mathcal{S}$, leading to

$$x_b' = \operatorname*{argmin}_{z \in \{x_b, x_k\}} f(z) \in \mathcal{S}. \tag{18}$$

Let $t^* \in \mathbb{R}^{2M+1}$ be the minimizer of the subspace problem (14) associated to the subspace (13). By (18) and setting $\overline{x}_b = \widetilde{U}_k t^*$, we can write

$$f_{\overline{x}_b} = \min_t f(\widetilde{U}_k t) = \min_{z \in \mathcal{S}} f(z) \le \min_{x \in \{x_b', x_k'\}} f(x) = \min\{f_{x_b'}, f_{x_k'}\},$$

giving the result. □

Note that if $\widetilde{U}_k$ and V_{ik}, for $i = 1, \dots, p$, are collected in the outer stage of OSGA-S, then no extra efforts for computing them are needed in applying the subspace search scheme MDSS (see Lines 5, 8, 10, and 17 of OSGA-S). Let us assume $m \approx n$. Then Theorem 1 implies that in each step of OSGA for solving (14) one needs $\mathcal{O}(n)$ operations. Therefore, applying n step of OSGA to (14) have the complexity the same as one call of the oracle for the full-dimensional problem by the outer scheme. Since we suppose n is a large number, the cost of applying n_0 ($n_0 \ll n$) steps of OSGA to (14) can be ignored in comparison to the cost of a single call of the first-order oracle in the full dimension. Hence MDSS can be applied efficiently to accelerate OSGA without imposing too much computational cost for large-scale objectives involving expensive linear operators and cheap nonlinear terms.

Theorem 1 implies that OSGA-S is a special case of OSGA obtained by specializing the choice of Line 8 in OSGA. Therefore, all theoretical feature of OSGA remains valid. Therefore, OSGA-S is optimal for smooth problems with Lipschitz continuous gradients, Lipschitz continuous nonsmooth problems, and strongly convex problems. We summarize this result in the next theorem that was proved in [42].

Theorem 2 *Let $f - \mu Q$ be a convex function. Then we have*

(i) *(Nonsmooth complexity bound) If f is Lipschitz continuous in $\mathcal{V}$, the total number of iterations needed by OSGA-S is at most $\mathcal{O}((\varepsilon^2 + \mu\varepsilon)^{-1})$. Thus the asymptotic worst-case complexity is $\mathcal{O}(\varepsilon^{-2})$ when $\mu = 0$ and $\mathcal{O}(\varepsilon^{-1})$ when $\mu > 0$.*

(ii) *(Smooth complexity bound) If f has Lipschitz continuous gradients with Lipschitz constant L, the total number of iterations needed by OSGA-S is at most $\mathcal{O}(\varepsilon^{-1/2})$ if $\mu = 0$, and $O(|\log \varepsilon|\sqrt{L/\mu})$ if $\mu > 0$.*

4 Numerical Experiments

A software package for solving unconstrained and simply constrained convex optimization problems with OSGA and the code of OSGA-S is publicly available at http://homepage.univie.ac.at/masoud.ahookhosh/.

The package is written in MATLAB. It uses the parameters

$$\delta = 0.9, \quad \alpha_{max} = 0.7, \quad \kappa = \kappa' = 0.5,$$

and the quadratic prox-function with $Q_0 = \frac{1}{2}\|x_0\|_2 + \epsilon$, where ϵ is the machine precision. A user manual [2] describes the design and use of the package. Some examples are included as illustrations. This section discusses numerical results and comparisons of OSGA-S with OSGA and some subgradient algorithms. All numerical experiments were executed on a PC Intel Core i7-3770 CPU 3.40GHz 8 GB RAM.

We compare OSGA-S with OSGA, SGA-1 (a non-summable diminishing steplength subgradient algorithm, cf. [14]), SGA-2 (a non-summable diminishing step-size subgradient algorithm, cf. [14]), and NESUN (Nesterov's universal gradient method, cf. [41]). In our implementation, SGA-1 and SGA-2 use the following step-sizes

$$\alpha_k^1 := \frac{\alpha_0}{\sqrt{k}\|g_k\|}, \quad \alpha_k^2 := \frac{\alpha_0}{\sqrt{k}},$$

respectively, where $\alpha_k^1, \alpha_k^2 > 0$.

4.1 Overdetermined Linear System of Equations

Consider the overdetermined linear system of equations

$$y = Ax + \nu, \tag{19}$$

where $x \in \mathbb{R}^n$ is an unknown vector, $A \in \mathbb{R}^{m\times n}$ with $m > n$, $y \in \mathbb{R}^m$ is an observation vector, and $\nu \in \mathbb{R}^m$ is unknown but small an additive noise. The objective is to recover x from y by solving (19). Such problems appear in many applications, see, e.g., [9,10,16]. They are of particular interest for robust fitting of linear models to data. In practice, this problem is typically ill-posed, cf. [43]. Therefore, x is usually computed by a minimization problem of the form (1) with one of the objective functions of Table 1.

Here, we set

$$\mathtt{A} = \mathtt{rand(m,n)} - 0.5, \quad \mathtt{y} = \mathtt{rand(m,1)} - 0.5, \quad \mathtt{x0} = \mathtt{rand(n,1)} - 0.5,$$

where $m = 50000$ and $n = 5000$. Since some of the problems given in Table 1 involve regularization terms that NESUN subproblem cannot be solved efficiently (e.g., $\|\cdot\|_\infty$), we will not consider it in this comparison. We therefore use SGA-1, SGA-2, OSGA, and OSGA-S for solving this overdetermined system of equations. We set $\alpha_0 = 8 \times 10^{-1}$ for SGA-1, use $\alpha_0 = 10^{-4}$ for SGA-2 if it applies to the problems L22R, L22L22R, L22L1R, L1R, L1L22R, and L1L1R, and exploit $\alpha_0 = 2 \times 10^{-2}$ for SGA-2 if it applies to the problems L2R, L2L22R, L2L1R, LIR, LIL22R, and LIL1R. Note that SGA-2 is very sensitive to the parameter α_0 for different problems, so we

Table 1 List of minimization problems for solving overdetermined systems of equations, where λ denotes the regularization parameter

Function	Name	Function	Name
$f_1(x) = \frac{1}{2}\|y - Ax\|_2^2$	L22R	$f_7(x) = \|y - Ax\|_1$	L1R
$f_2(x) = \frac{1}{2}\|y - Ax\|_2^2 + \frac{1}{2}\lambda\|x\|_2^2$	L22L22R	$f_8(x) = \|y - Ax\|_1 + \frac{1}{2}\lambda\|x\|_2^2$	L1L22R
$f_3(x) = \frac{1}{2}\|y - Ax\|_2^2 + \lambda\|x\|_1$	L22L1R	$f_9(x) = \|y - Ax\|_1 + \lambda\|x\|_1$	L1L1R
$f_4(x) = \|y - Ax\|_2$	L2R	$f_{10}(x) = \|y - Ax\|_\infty$	LIR
$f_5(x) = \|y - Ax\|_2 + \frac{1}{2}\lambda\|x\|_2^2$	L2L22R	$f_{11}(x) = \|y - Ax\|_\infty + \frac{1}{2}\lambda\|x\|_2^2$	LIL22R
$f_6(x) = \|y - Ax\|_2 + \lambda\|x\|_1$	L2L1R	$f_{12}(x) = \|y - Ax\|_\infty + \lambda\|x\|_1$	LIL1R

The objective functions are convex and contain a linear mapping A, which is typically a dense matrix and y is defined by (19)

Table 2 Summary of numerical results for all 12 problems of Table 1, where M_{best} denotes the best value of the parameter $M \in \{1, 2, \ldots, 20\}$, $N(M)$ stands for the number of iterations needed to achieve the function value less or equal than f_s, and $T(M)$ denotes the corresponding running time

Problem name	Iteration				Time (s)			
	M_{best}	$N(M_{\text{best}})$	$N(2)$	$N(20)$	M_{best}	$T(M_{\text{best}})$	$T(2)$	$T(20)$
L22R	11	17	22	33	7	8.31	11.32	25.64
L22L22R	8	22	43	42	8	10.18	16.88	27.82
L22L1R	2	14	14	20	4	6.92	7.41	10.22
L2R	5	23	27	39	5	9.76	11.22	23.74
L2L22R	1	18	19	39	1	6.94	7.62	18.61
L2L1R	1	27	36	31	1	10.56	14.72	73.12
L1R	1	93	94	103	1	36.90	40.80	73.66
L1L22R	3	95	100	104	3	40.22	41.77	82.40
L1L1R	19	59	85	114	2	35.46	35.46	65.66
LIR	1	2	2	93	1	0.67	0.69	3.57
LIL22R	10	32	64	88	10	16.60	26.33	61.95
LIL1R	1	3	8	92	1	1.12	3.19	67.75

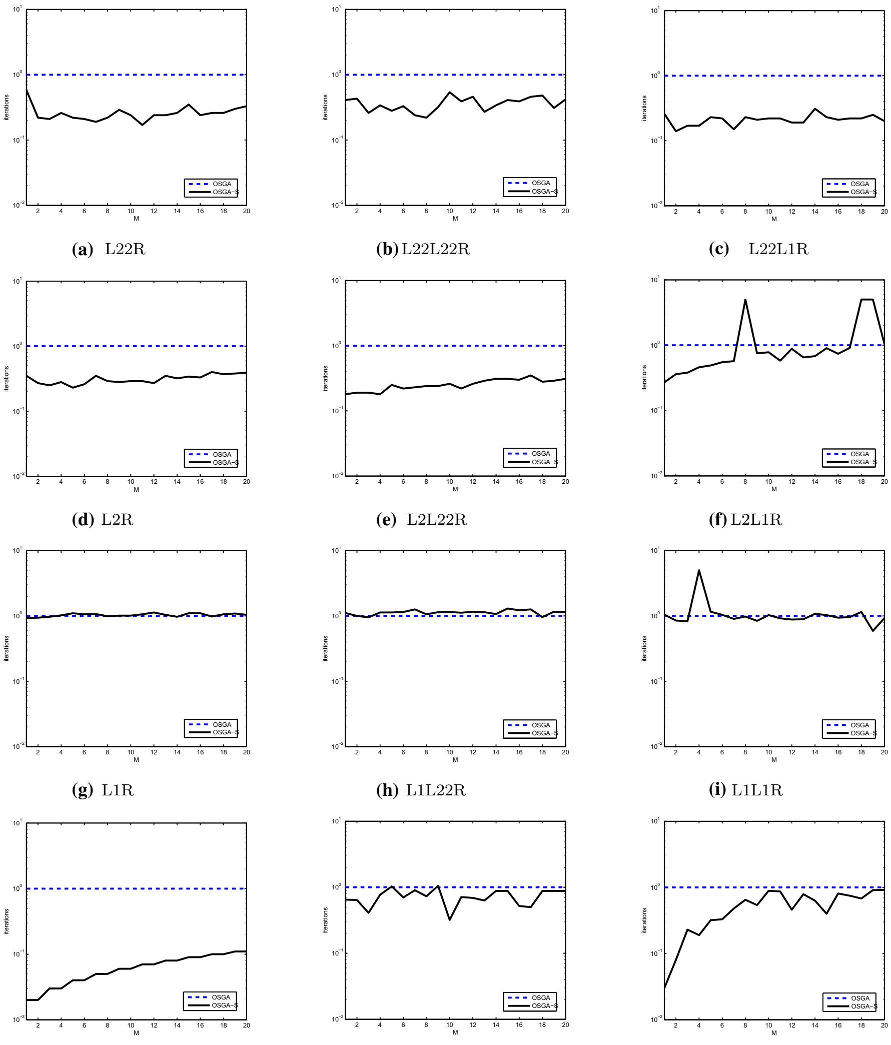
tunned α_0 to attain the best performance of SGA-2 for the considered set of problems. For all problems of Table 1, we set $\lambda = 1$.

We first conduct an experiment on the parameter M to find an optimal range for this parameter. To this end, we consider the problems of Table 1, solve the problem by OSGA in 100 iterations, save the best function value f_s in each case, and run OSGA-S with $M = 1, 2, \ldots, 20$ to achieve f_s. The results of our experiment are summarized in Table 2 and Figs. 1 and 2. In Table 2, the best parameter M_{best} for each problem regarding the best number of iterations and the best running time, along with the results for M_{best}, $M = 2$, and $M = 20$, is reported. Figures 1 and 2 illustrate comparisons between OSGA and OSGA-S for $M = 1, 2, \ldots, 20$, where Fig. 1 shows the results for the number of iterations and Fig. 2 displays the results for the running time.

From the results of Table 2 and Figs. 1 and 2, it can be seen that M_{best} is varied for the considered problems; however, the interval $[1, 5]$ seems to be statistically reasonable for the parameter M. In addition, it is clear that the performance of OSGA-S depends on the parameter M, but if we set $M \in [1,\ 5]$, OSGA-S outperforms OSGA except for L1R, L1L22R, and L1L1R (see figures (g), (h), and (i) of Fig. 2).

We now solve the problems reported in Table 1 by SGA-1, SGA-2, OSGA, and OSGA-S, where we first solve these problems by OSGA in 100 iterations, save the best function value f_s and stop the others whenever they attain a function value less or equal than f_s or the number of iterations reaches to the maximum number of iterations, which is 500 here. We set $M = 2$ for OSGA-S. The results of implementation are summarized in Table 3 and Fig. 3.

In Table 3, N and T denote the number of iterations and the running time, respectively. The results of Table 3 show that OSGA and OSGA-S outperform SGA-1 and SGA-2 significantly regarding both the number of iterations and the running time; however, OSGA-S needs fewer iterations and less running time than OSGA. In Fig.



(a) L22R **(b)** L22L22R **(c)** L22L1R

(d) L2R **(e)** L2L22R **(f)** L2L1R

(g) L1R **(h)** L1L22R **(i)** L1L1R

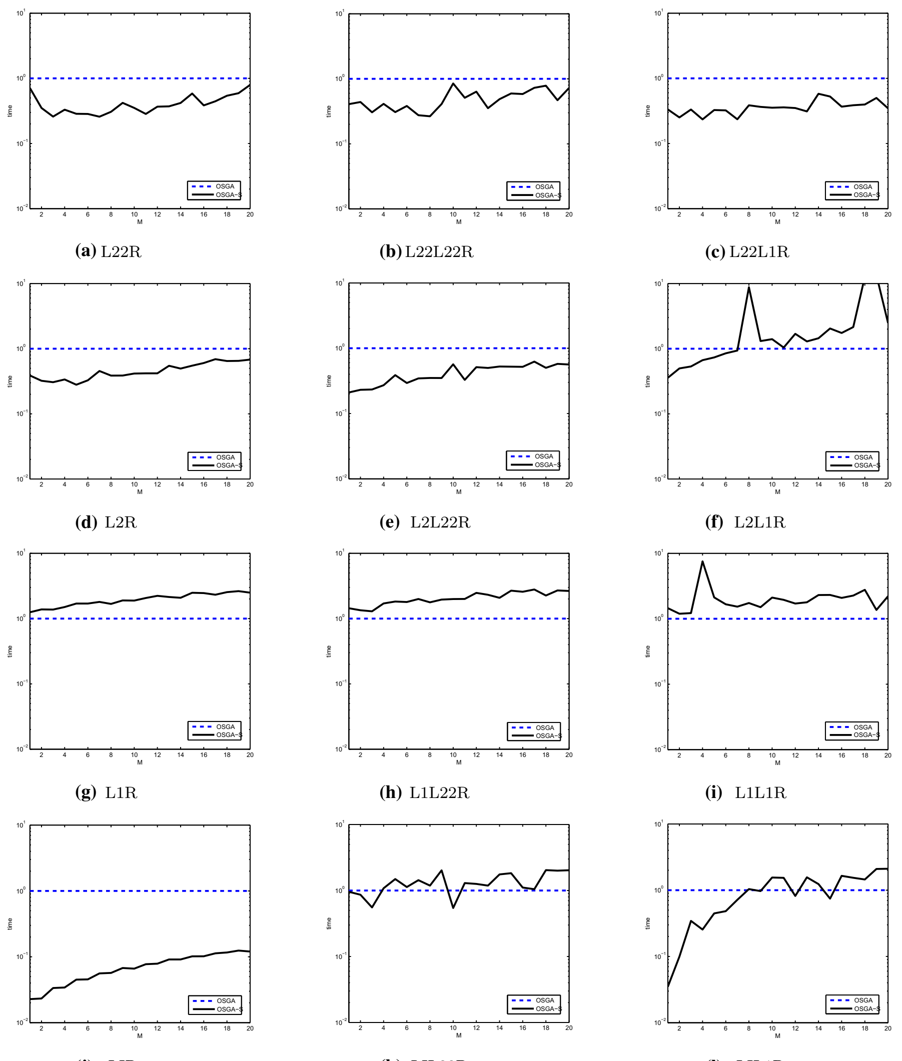
(j) LIR **(k)** LIL22R **(l)** LIL1R

Fig. 1 The relation $N(M)/N_{OSGA}$ against M for OSGA and OSGA-S for solving the problems in Table 1, where $N(M)$ denotes the total number of iterations

3, we illustrate the relative error of function values versus iterations, i.e.,

$$\delta_k := \frac{f_k - \widehat{f}}{f_0 - \widehat{f}}, \tag{20}$$

where f_0, f_k, and $\widehat{f}$ denote the function values at a starting point x_0, the current point x_k, and the minimizer $\widehat{x}$, respectively. The results of Fig. 3 show that in many cases OSGA-S get the same accuracy in fewer iterations and less running time; however, for



(a) L22R **(b)** L22L22R **(c)** L22L1R

(d) L2R **(e)** L2L22R **(f)** L2L1R

(g) L1R **(h)** L1L22R **(i)** L1L1R

(j) LIR **(k)** LIL22R **(l)** LIL1R

Fig. 2 The relation $T(M)/T_{OSGA}$ against M for OSGA and OSGA-S for solving the problems in Table 1, where $T(M)$ denotes the running time

some cases such as L1R, L22L1R, and L1L1R the difference between the number of iterations of OSGA-S and OSGA is not significant and worse running time are attained by OSGA-S. Moreover, the results of OSGA-S is much better than SGA-1, SGA-2, and OSGA for LIR and LIL1R that might be because of poor sparse subgradients of the infinity norm $\|\cdot\|_\infty$ for SGA-1, SGA-2, and OSGA, while the subspace minimization step of OSGA-S involves a combination of several former points resulting to better directions.

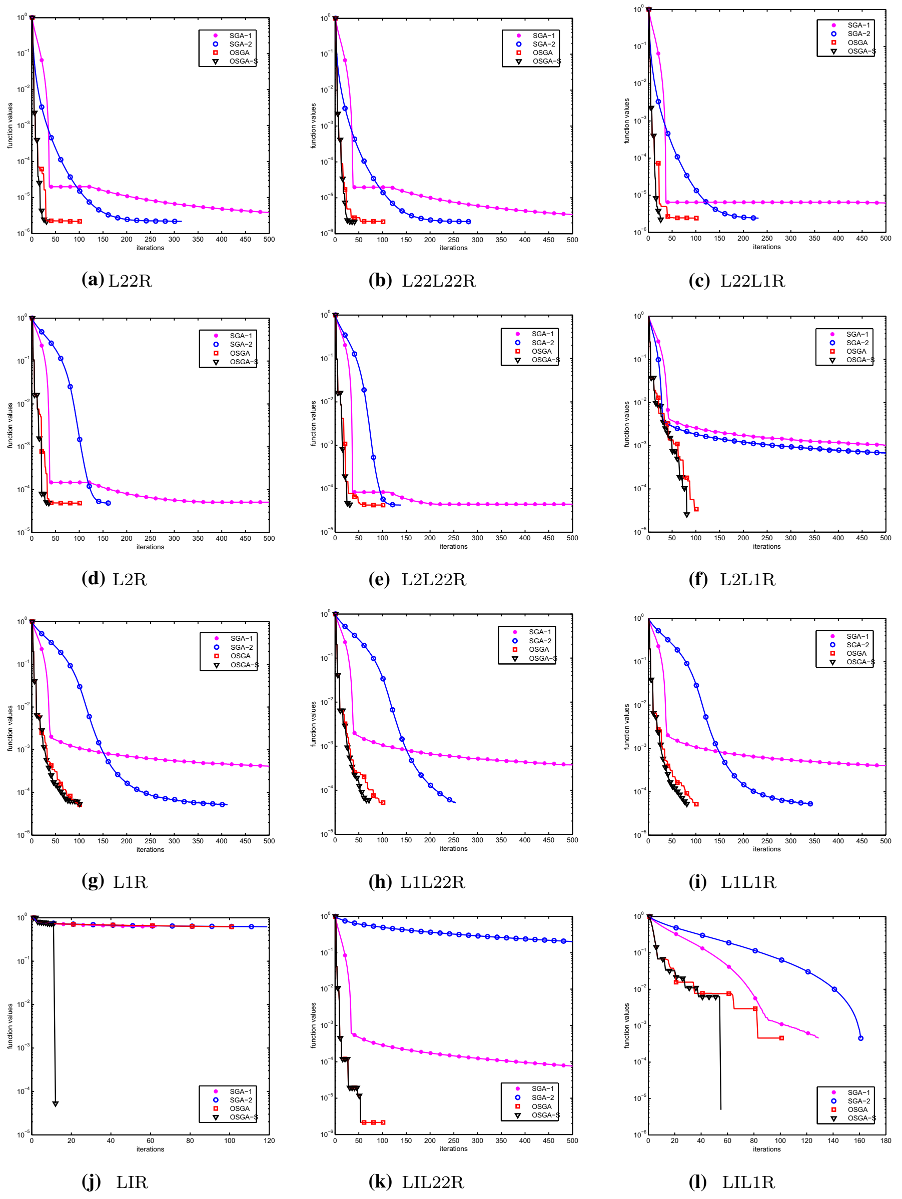


Fig. 3 The relative error of function values δ_k against iterations for SGA-1, SGA-2, OSGA, and OSGA-S for solving overdetermined systems of equations using the minimization problems presented in Table 1

Table 3 Numerical results of SGA-1, SGA-2, OSGA, and OSGA-S, where the best time (in second) is displayed as bold

Problem name	SGA-1		SGA-2		OSGA		OSGA-S	
	N	T	N	T	N	T	N	T
L22R	500	125.33	316	77.50	100	40.39	30	**21.35**
L22L22R	500	125.52	288	70.73	100	37.96	43	**33.60**
L22L1R	500	102.36	232	62.50	100	38.30	25	**19.16**
L2R	500	137.48	162	45.34	100	45.43	38	**29.85**
L2L22R	500	144.80	139	43.22	100	44.53	31	**24.24**
L2L1R	500	138.72	500	117.80	100	42.20	80	**66.71**
L1R	500	124.84	412	107.57	100	**40.14**	100	87.10
L1L22R	500	126.45	254	63.46	100	**41.10**	72	62.28
L1L1R	500	125.46	346	84.62	100	**41.37**	81	68.63
LIR	63	15.67	119	31.36	100	41.76	11	**4.93**
LIL22R	500	136.61	500	133.70	100	45.67	53	**44.82**
LIL1R	129	**32.19**	161	42.56	100	42.26	45	46.74

4.2 Support Vector Machines

The learning with support vector machines (SVM) leads to several expensive convex optimization problems with large dense data set. Some of these problems have the form designed in this paper. Let us consider a binary classification, where a set of training data $(x_1, y_1), \ldots, (x_q, y_q)$ in which $x_i \in \mathbb{R}^n$ and $y_i \in \{-1, 1\}$ for $i = 1, \ldots, q$ is given. The aim is to find a classification rule using the training data, so that for a new point x one can assign a class $y \in \{-1, 1\}$ to x by the derived classification rule. The classification rule for SVM is given by the sign of $\langle x, w\rangle + w_0$, where w and w_0 may be determined by solving a penalized problem

$$\begin{array}{ll} \min & \sum_{i=1}^{q} [1 - y_i(\langle x_i, w\rangle + w_0)]_+ + \lambda\psi(w) \\ \text{s.t.} & w \in \mathbb{R}^n, \ w_0 \in \mathbb{R}, \end{array} \tag{21}$$

where $[z]_+ = \max\{z, 0\}$, and ψ can be $\|\cdot\|_1$ (SVML1R), $\|\cdot\|_2^2$ (SVML22R), and $\frac{1}{2}\|\cdot\|_2^2 + \|\cdot\|_1$ (SVML22L1R) (see, e.g., [13,48,55] and references therein). For $\langle x, w\rangle = w^T x$, let us define

$$X := \begin{pmatrix} y_1 x_1^T \\ \vdots \\ y_q x_q^T \end{pmatrix} \in \mathbb{R}^{q\times n}, \quad A := (X, y) \in \mathbb{R}^{q\times(n+1)}, \quad \widetilde{w} := \begin{pmatrix} w \\ w_0 \end{pmatrix} \in \mathbb{R}^{n+1}.$$

Then the problem (21) can be rewritten in the form

$$\begin{array}{ll} \min & \mathbf{1}^T\,[\mathbf{1} - A\widetilde{w}]_+ + \lambda\psi(w) \\ \text{s.t.} & \widetilde{w} \in \mathbb{R}^{n+1}, \end{array} \tag{22}$$

where $[\mathbf{1} - A\widetilde{w}]_+ = \max\{\mathbf{1} - A\widetilde{w}, 0\}$ and $\mathbf{1} \in \mathbb{R}^q$ is the vector of all ones. Typically A is a dense matrix constructed by data points x_i and y_i for $i = 1, \ldots, q$. It is clear that (22) is of the form (1), where an associated subgradient g is given by

$$g = -A^T\delta + \lambda \begin{pmatrix} \operatorname{sign}(w) \\ 0 \end{pmatrix},$$

in which

$$\delta_i = \begin{cases} 1 & \text{if } A_{i:}\widetilde{w} < 1, \\ 0 & \text{if } A_{i:}\widetilde{w} \geq 1, \end{cases}$$

for $i = 1, \ldots, q$.

In order to show the benefit of our subspace technique for this kind of problems, we apply SVML1R, SVML22R, and SVML22L1R to the leukemia data given by Golub et al. in [24], available in [25]. This dataset comes from a study of gene expression in two types of acute leukemias (acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL)) and it consists of 38 training data points and 34 test data points. We apply SVML1R, SVML22R, and SVML22L1R to the training data points ($q = 38$ and $n = 7129$) with six levels of regularization parameters. We first solve the problems by OSGA in 1000 iterations and save the best function value f_s in each case. We then run SGA-1, SGA-2, NESUN, OSGA, and OSGA-S, where they are stopped after 5000 iterations or after achieving a function value at least as good as f_s. The associated results are summarized in Table 4 and Figs. 4 and 5.

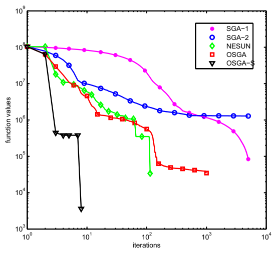
The results of Table 4 show that OSGA and NESUN are comparable but better than SGA-1 and SGA-2, and OSGA-S outperforms all others significantly with respect to the number of iterations (N) and the running time (T) (the best average is given by OSGA-S). In Figs. 4 and 5, we illustrate the function values versus iterations indicating that OSGA-S needs few iterations (typically less than 35 iterations) to get the accuracy that OSGA attains in 1000 iterations and SGA-1, and SGA-2 get in few thousands of iterations; however, this number of iterations is varied from about 100 to few thousands for NESUN for different problems. This shows a good potential of OSGA-S to be applied to machine learning problems.

We now consider the accuracy (the ratio of the number of correctly predicted data labels to the total number of data multiplied by 100) of OSGA-S for solving (22). Let us denote by OSGA-S-100, OSGA-S-50, OSGA-S-10, OSGA-S-1, OSGA-S-0.1, OSGA-S-0.01 the algorithm OSGA-S with the regularization parameter $\lambda = 100, 50, 10, 1, 0.1, 0.01$, respectively. We first investigate the relation between the accuracy and the number of iterations used to stop OSGA-S. To do so, we report the accuracy of OSGA-S-100, OSGA-S-50, OSGA-S-10, OSGA-S-1, OSGA-S-0.1, OSGA-S-0.01 when they are stopped after $10, 20, 30, 40, 50$ number of iterations.

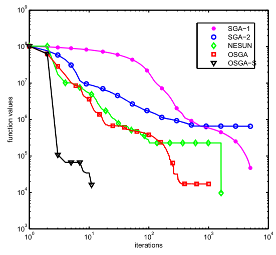
Table 4 Numerical results of SGA-1, SGA-2, NESUN, OSGA, and OSGA-S for a binary classification with linear support vector machines

Prob. name	Reg. par.	SGA-1			SGA-2			NESUN			OSGA			OSGA-S		
		N	T	f_b	N	T	f_b	N	T	f_b	N	T	f_s	N	T	f_b
SVML1R	$\lambda = 100$	5000	6.63	83576.85	5000	6.59	1282458.38	114	0.49	33818.37	1000	2.90	34577.96	**7**	**0.36**	**3614.44**
SVML1R	$\lambda = 50$	5000	6.77	46936.06	5000	6.69	653617.39	1619	6.86	**9648.03**	1000	2.87	17069.64	**10**	**0.51**	16277.52
SVML1R	$\lambda = 10$	5000	6.56	9874.56	5000	6.38	132705.86	1257	4.96	3876.52	1000	2.89	5285.64	**13**	**0.66**	**3607.89**
SVML1R	$\lambda = 1$	1387	1.97	10020.40	5000	6.79	13315.34	1228	4.83	6868.29	1000	2.85	10020.44	**10**	**0.51**	**2476.24**
SVML1R	$\lambda = 0.1$	1485	1.97	960.47	5000	6.56	1331.98	5000	15.88	1087.16	1000	2.90	960.70	**12**	**0.62**	**781.16**
SVML1R	$\lambda = 0.01$	1323	1.71	103.03	5000	6.33	133.20	178	**0.64**	93.31	1000	2.93	103.04	**33**	1.86	**77.37**
SVML22R	$\lambda = 100$	4282	5.72	983.42	5000	6.62	1122462.85	80	**0.30**	**646.13**	1000	2.92	996.61	**11**	0.54	741.47
SVML22R	$\lambda = 50$	4318	5.66	426.57	5000	6.66	602869.56	65	**0.25**	314.11	1000	2.92	430.96	**13**	0.63	**0.16**
SVML22R	$\lambda = 10$	4450	5.73	9.82	5000	6.34	127514.18	57	**0.20**	6.45	1000	2.80	10.14	**16**	0.78	**5.26**
SVML22R	$\lambda = 1$	3197	4.04	899.17	5000	6.33	12913.02	56	**0.19**	**630.80**	1000	2.77	900.06	**24**	1.17	879.25
SVML22R	$\lambda = 0.1$	1263	1.62	814.81	5000	6.18	10292.99	56	**0.19**	717.57	1000	2.85	960.70	**13**	0.63	**0.000038**
SVML22R	$\lambda = 0.01$	1118	1.44	92.56	5000	6.36	129.31	59	**0.21**	80.19	1000	2.81	92.61	**10**	0.48	**0.00038**
SVML22L1R	$\lambda = 100$	4540	6.42	3330.77	5000	6.81	2236212.30	391	1.58	1556.42	1000	2.95	3334.02	**4**	**0.21**	**309.67**
SVML22L1R	$\lambda = 50$	4499	6.10	29747.51	5000	6.79	1211931.69	640	2.61	10483.67	1000	2.80	29767.84	**15**	**0.79**	3720.47
SVML22L1R	$\lambda = 10$	4624	6.26	5331.55	5000	6.67	258375.25	130	**0.49**	2883.38	1000	2.87	5337.66	**10**	0.53	**3910.16**
SVML22L1R	$\lambda = 1$	2365	3.30	9061.50	5000	6.66	26209.49	896	3.51	4865.97	1000	2.90	9065.61	**10**	**0.53**	**1851.74**
SVML22L1R	$\lambda = 0.1$	3595	5.11	321.12	5000	6.96	2624.79	5000	14.85	1997.35	1000	2.95	321.45	**10**	**0.51**	**1.58**
SVML22L1R	$\lambda = 0.01$	1654	2.27	146.05	5000	6.85	262.51	5000	15.38	210.31	1000	2.83	146.05	**10**	**0.53**	**0.03**
Average		3283.33	4.40	12920.45	5000	6.58	427520.00	1212.55	4.07	10452.49	1000	2.87	6632.28	**12.83**	**0.65**	**2125.24**

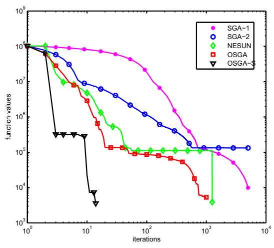
N, T, and f_b denote the number of the function values, the running time, and the best function value achieved by the associated algorithm. In each problems, the best iteration, time (in second), and function value are displayed as bold



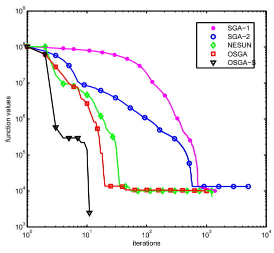
(a) SVML1R, λ =100



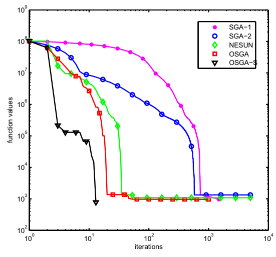
(b) SVML1R, λ =50



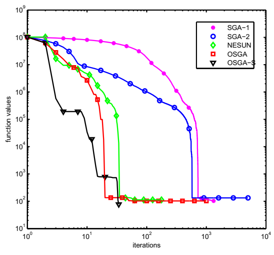
(c) SVML1R, λ =10



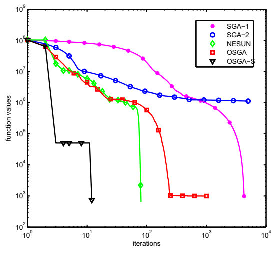
(d) SVML1R, λ =1



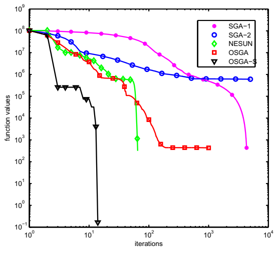
(e) SVML1R, λ =0 .1



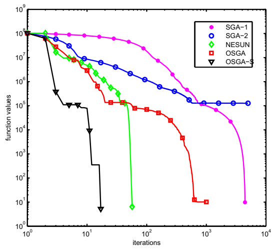
(f) SVML1R, λ =0 .01



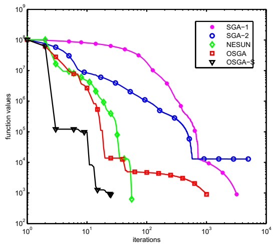
(g) SVML22R, λ =100



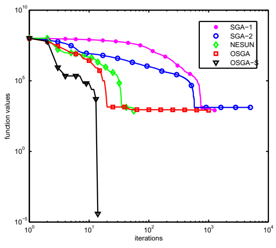
(h) SVML22R, λ =50



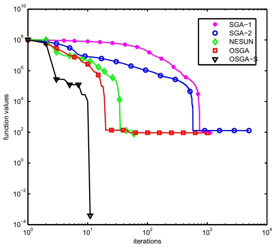
(i) SVML22R, λ =10



(j) SVML22R, λ =1

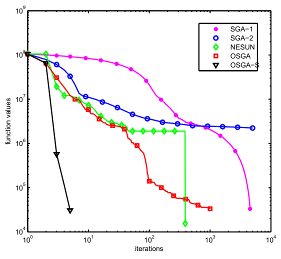


(k) SVML22R, λ =0 .1

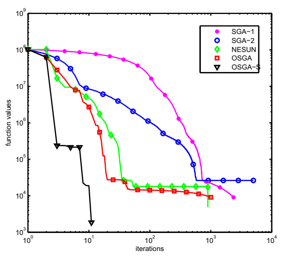


(l) SVML22R, λ =0 .01

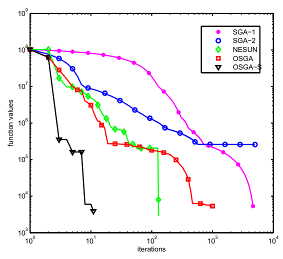
Fig. 4 The function values against iterations for the algorithms SGA-1, SGA-2, NESUN, OSGA, and OSGA-S for a binary classification with linear support vector machines (SVML1R, SVML22R)



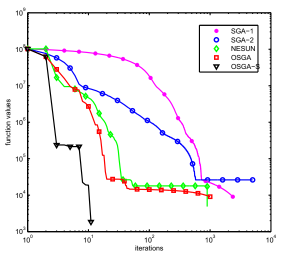
(a) SVML22L1R, $\lambda = 100$



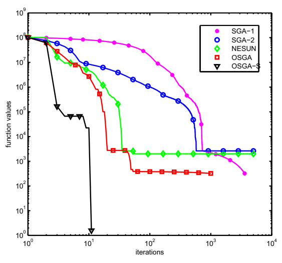
(b) SVML22L1R, $\lambda = 50$



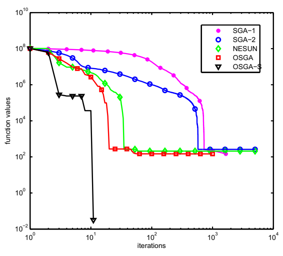
(c) SVML22L1R, $\lambda = 10$



(d) SVML22L1R, $\lambda = 1$



(e) SVML22L1R, $\lambda = 0.1$



(f) SVML22L1R, $\lambda = 0.01$

Fig. 5 The function values against iterations for the algorithms SGA-1, SGA-2, NESUN, OSGA, and OSGA-S for a binary classification with linear support vector machines (SVML22L1R)

The results are summarized in Table 5. From the results of this table, it is clear that in many cases the accuracy of OSGA-S is increased by considering a bigger number of iterations; however, it produces acceptable results after 50 iterations. In addition, it can be seen that the regularization parameter plays a crucial role in the accuracy of OSGA-S.

Among state-of-the-art SVM solvers, we here compare the accuracy of OSGA-S with LIBSVM [17], FITCSVM (MATLAB internal function), PEGASOS [47], SVMperf [29] with their default parameters to solve (22). In our implementation, LIBSVM, FITCSVM, SVMperf, and PEGASOS attain the accuracies 65.27, 79.17, 79.17, and 69.44, respectively. A comparison among these accuracies with those reported in Table 5 shows that the accuracy of OSGA-S is comparable or even better than LIBSVM, FITCSVM, SVMperf, and PEGASOS for the considered data set. Since the number of training and testing data for the leukemia data set [24,25] is small, we consider a comparison among OSGA-S and LIBSVM, FITCSVM, SVMperf, and PEGASOS for w1a–w8a data sets [45]. After training procedure, we consider a concatenation of the training and testing data and apply the derived classification functions, where the obtained accuracy for each solver is reported in Table 6. In this table, OSGA-S-1, OSGA-S-2, and OSGA-S-3 stand for OSGA-S for the problems SVML1R, SVML22R, and SVML22L1R, where we tune the regularization parameter to get the best performance of OSGA-S (see the numbers in parentheses of the last three columns of Table 6) and stop OSGA-S after 50 iterations. The results of Table 6 show that the accuracy of OSGA-S is almost comparable with those of state-of-the-art solvers LIBSVM, FITCSVM, SVMperf, and PEGASOS.

Table 5 The accuracy of OSGA-S for several levels of the regularization parameters after various number of iterations for solving SVML1R, SVML22R, and SVML22L1R

Problem name	Iterations	OSGA-S-100	OSGA-S-50	OSGA-S-10	OSGA-S-1	OSGA-S-0.1	OSGA-S-0.01
SVML1R	10	55.55	81.94	69.44	68.05	77.77	70.83
SVML1R	20	77.77	81.94	72.22	69.44	77.77	75.00
SVML1R	30	77.77	75.00	81.94	70.83	81.94	83.33
SVML1R	40	81.94	75.00	81.94	83.33	81.94	83.33
SVML1R	50	81.94	75.00	81.94	83.33	81.94	84.72
SVML22R	10	70.83	72.22	72.22	66.66	55.55	79.16
SVML22R	20	81.94	72.22	70.83	76.38	73.61	77.77
SVML22R	30	80.55	80.55	73.61	76.38	79.16	77.77
SVML22R	40	80.55	83.33	79.16	77.77	80.55	77.77
SVML22R	50	80.55	83.33	81.94	77.77	80.55	77.77
SVML22L1R	10	79.16	38.88	81.94	70.83	72.22	72.22
SVML22L1R	20	83.33	75.00	81.94	70.83	80.55	83.33
SVML22L1R	30	83.33	83.33	83.33	77.77	80.55	83.33
SVML22L1R	40	77.77	83.33	83.33	84.72	80.55	83.33
SVML22L1R	50	75.00	83.33	83.33	84.72	80.55	84.72

Table 6 The accuracy of LIBSVM, FITCSVM, SVMperf, PEGASOS, OSGA-S-1, OSGA-S-2, OSGA-S-3 for solving the problem (22). The number of features for all data is 300, and TrD and TeD denote the number of training and testing data, respectively. For OSGA-S-1, OSGA-S-2, OSGA-S-3, the number in parentheses stands for the regularization parameter

Data	TrD	TeD	LIBSVM	FITCSVM	SVMperf	PEGASOS	OSGA-S-1	OSGA-S-2	OSGA-S-3
w1a	2477	49749	97.35	97.81	97.59	97.02	97.03 (2000)	95.67 (720)	97.03 (1400)
w2a	3470	49749	98.17	98.12	97.99	97.02	97.03 (1050)	96.92 (500)	97.03 (2000)
w3a	4912	49749	97.97	98.36	97.95	97.02	97.03 (1000)	96.85 (500)	97.03 (1500)
w4a	7366	49749	97.33	98.47	97.93	97.02	95.84 (1700)	96.75 (1000)	97.03 (1500)
w5a	9888	49749	86.38	98.58	98.11	97.02	97.03 (2100)	96.50 (1100)	97.03 (2000)
w6a	17188	49749	81.44	98.67	98.15	97.02	96.72 (1000)	96.92 (2150)	97.03 (1500)
w7a	24692	49749	75.10	98.69	98.24	97.02	97.03 (1350)	95.72 (1800)	97.03 (1400)
w8a	49749	64700	63.95	98.72	97.39	97.01	97.01 (2250)	89.07 (2250)	97.01 (1400)

5 Conclusions

In this paper we give an iterative scheme for solving convex optimization problems involving costly linear operators with cheap nonlinear terms. More precisely, we combine OSGA with a multidimensional subspace search, which leads to solve a sequence of low-dimensional subproblems that can be solved efficiently by OSGA. Numerical results for overdetermined system of equations and support vector machines show the efficiency of the scheme proposed.

Acknowledgements Open access funding provided by University of Vienna. We are very thankful to anonymous referees for careful reading and many useful suggestions, which improved the paper.

References

1. Ahookhosh, M.: Optimal subgradient methods: computational properties for large-scale inverse problems. Optim. Eng. (2018). https://doi.org/10.1007/s11081-018-9378-5
2. Ahookhosh, M.: User's manual for OSGA (Optimal SubGradient Algorithm) (2015). http://homepage.univie.ac.at/masoud.ahookhosh/uploads/User' s_manual_for_OSGA.pdf
3. Ahookhosh, M., Neumaier, A.: High-dimensional convex optimization via optimal affine subgradient algorithms, in ROKS workshop, 83–84 (2013)
4. Ahookhosh, M., Neumaier, A.: An optimal subgradient algorithm for large-scale bound-constrained convex optimization. Math. Methods Oper. Res. **86**(1), 123–147 (2017)
5. Ahookhosh, M., Neumaier, A.: Optimal subgradient algorithms for large-scale convex optimization in simple domains. Numer. Algorithms **76**(4), 1071–1097 (2017)
6. Ahookhosh, M., Neumaier, A.: Solving nonsmooth convex optimization with complexity $O(\varepsilon^{-1/2})$. TOP **26**(1), 110–145 (2018)
7. Auslender, A., Teboulle, M.: Interior gradient and proximal methods for convex and conic optimization. SIAM J. Optim. **16**, 697–725 (2006)
8. Bach, F., Jenatton, R., Mairal, J., Obozinski, G.: Convex optimization with sparsity-inducing norms. In: Optimization for Machine Learning. MIT press, Cambridge (2011)
9. Bartels, R.H., Conn, A.R., Li, Y.: Primal methods are better than dual methods for solving overdetermined linear systems in the l_∞ sense? SIAM J. Numer. Anal. **26**(3), 693–726 (1989)
10. Bartels, R.H., Conn, A.R., Sinclair, J.W.: Minimization techniques for piecewise differentiable functions: the l_1 solution to an overdetermined linear system. SIAM J. Numer. Anal. **15**, 224–241 (1978)
11. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**, 183–202 (2009)
12. Beck, A., Teboulle, M.: Mirror descent and nonlinear projected subgradient methods for convex optimization. Oper. Res. Lett. **31**(3), 167–175 (2003)
13. Bottou, L., Lin, C.J.: Support vector machine solvers. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines. MIT Press, Cambridge (2007)
14. Boyd, S., Xiao, L., Mutapcic, A.: Subgradient Methods, Notes for EE392o, Stanford University (2003). http://www.stanford.edu/class/ee392o/subgrad_method.pdf
15. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
16. Cadzow, J.A.: Minimum l_1, l_2, and l_∞ norm approximate solutions to an overdetermined system of linear equations. Digital Signal Process. **12**, 524–560 (2002)
17. Chang, C.C., Lin, C.J.: LIBSVM : a library for support vector machines. ACM Trans. Intell. Syst. Technol. 1–27 (2011). https://www.csie.ntu.edu.tw/~cjlin/libsvm/

18. Chouzenoux, E., Idier, J., Moussaoui, S.: A majorize-minimize strategy for subspace optimization applied to image restoration. IEEE Trans. Image Process. **20**(18), 1517–1528 (2011)
19. Combettes, P.L., Pesquet, J.C.: Proximal splitting methods in signal processing. In: Fixed-Point Algorithms for Inverse Problems in Science and Engineering. Springer, New York (2010)
20. Conn, A.R., Gould, N., Sartenaer, A., Toint, PhL: On iterated-subspace minimization methods for nonlinear optimization, Technical report 94/13, Department of Mathematics, Facultes Universitaires Notre Dame de la Paix, Namur, Belgium (1994)
21. Cragg, E.E., Levy, A.V.: Study on a supermemory gradient method for the minimization of functions. J. Optim. Theory Appl. **4**(3), 191–205 (1969)
22. Devolder, O., Glineur, F., Nesterov, Y.: First-order methods of smooth convex optimization with inexact oracle. Mathe. Program. **146**, 37–75 (2013)
23. Elad, M., Matalon, B., Zibulevsky, M.: Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. Appl. Comput. Harmon. Anal. **23**(3), 346–367 (2006)
24. Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science **286**, 531–536 (1999)
25. http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43
26. Gonzaga, C.C., Karas, E.W.: Fine tuning Nesterov's steepest descent algorithm for differentiable convex programming. Math. Program. **138**, 141–166 (2013)
27. Gonzaga, C.C., Karas, E.W., Rossetto, D.R.: An optimal algorithm for constrained differentiable convex optimization. SIAM J. Optim. **23**(4), 1939–1955 (2013)
28. Grapiglia, G.N., Yuan, J.Y., Yuan, Y.: A subspace version of the Powell-Yuan trust region algorithm for equality constrained optimization. J. Oper. Res. Soc. China **1**(4), 425–451 (2013)
29. Joachims, T., Finley, T., Yu, C.N.: Cutting-plane training of structural SVMs. Mach. Learn. J. **77**(1), 27–59 (2009). https://www.cs.cornell.edu/people/tj/svm_light/svm_perf.html and http://www.aic.uniovi.es/~oluaces/Oscars_Home_Page/Software/Software.html
30. Lan, G.: Bundle-level type methods uniformly optimal for smooth and non-smooth convex optimization. Math. Program. **149**, 1–45 (2015)
31. Lan, G., Lu, Z., Monteiro, R.D.C.: Primal-dual first-order methods with $O(1/\varepsilon)$ iteration-complexity for cone programming. Math. Program. **126**, 1–29 (2011)
32. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Math. Program. **45**(3), 503–528 (1989)
33. Miele, A., Cantrell, J.W.: Study on a memory gradient method for the minimization of functions. J. Optim. Theory Appl. **3**(6), 459–470 (1969)
34. Narkiss, G., Zibulevsky, M.: Sequential subspace optimization method for large-scale unconstrained problems, Technical report CCIT 559, EE Dept., Technion, Haifa, Israel (2005)
35. Narkiss, G., Zibulevsky, M.: Support vector machine via sequential subspace optimization, Technical Report CCIT 557, Technion Israel Institute of Technology, Faculty of Electrical Engineering (2005)
36. Nemirovsky, A.S., Yudin, D.B.: Problem Complexity and Method Efficiency in Optimization. Wiley, New York (1983)
37. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer, Dordrecht (2004)
38. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $O(1/k^2)$, Doklady AN SSSR (In Russian), **269**, 543–547 (1983). English translation: Soviet Math. Dokl. **27**, 372–376 (1983)
39. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. **103**, 127–152 (2005)
40. Nesterov, Y.: Gradient methods for minimizing composite objective function. Math. Program. **140**, 125–161 (2013)
41. Nesterov, Y.: Universal gradient methods for convex optimization problems. Math. Program. **152**, 381–404 (2014)
42. Neumaier, A.: OSGA: a fast subgradient algorithm with optimal complexity. Math. Program. **158**(1–2), 1–21 (2016)
43. Neumaier, A.: Solving ill-conditioned and singular linear systems: a tutorial on regularization. SIAM Rev. **40**(3), 636–666 (1998)
44. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (2006)

45. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, Bernhard, Burges, Christopher J.C., Smola, Alexander J. (eds.) Advances in Kernel Methods—Support Vector Learning. MIT Press, Cambridge (1998)
46. Polyak, B.: Introduction to Optimization. Optimization Software Inc., Publications Division, New York (1987)
47. Shalev-Shwartz, S., Singer, Y., Srebro, N.: Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In: IEEE International Conference on Machine Learning (ICML), (2007). https://www.mathworks.com/matlabcentral/fileexchange/31401-pegasos-primal-estimated-sub-gradient-solver-for-svm
48. Shawe-Taylor, J., Sun, S.: A review of optimization methodologies in support vector machines. Neurocomputing **74**, 3609–3618 (2011)
49. Shor, N.Z.: Minimization Methods for non-differentiable functions, Springer Series in Computational Mathematics. Springer, New York (1985)
50. Sra, S., Nowozin, S., Wright, S.J. (eds.): Optimization for Machine Learning. The MIT Press, Cambridge (2011)
51. Stoer, J., Yuan, Y.: A subspace study on conjugate gradient algorithms. J. Appl. Math. Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik **75**, 69–77 (1995)
52. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization, Technical report, Mathematics Department, University of Washington (2008). http://pages.cs.wisc.edu/~brecht/cs726docs/Tseng.APG.pdf
53. Wang, Z., Yuan, Y.: A subspace implementation of quasi-Newton trust region methods for unconstrained optimization. Numerische Mathematik **104**, 241–269 (2006)
54. Yuan, Y.: Subspace techniques for nonlinear optimization. In: R. Jeltsch, D.Q. Li, I. H. Sloan (eds.) Some Topics in Industrial and Applied Mathematics (Series in Contemporary Applied Mathematics CAM 8), pp. 206–218. Higher Education Press. Beijing (2007)
55. Zhu, J., Rosset, S., Hastie, T., Tibshirani, R.: 1-norm support vector machines. In: Proceedings of the 16th International Conference on Neural Information Processing Systems, Whistler, British Columbia, Canada, 9–11 December 2003