Noname manuscript No.
(will be inserted by the editor)

Distributed algorithms for convex problems with linear coupling constraints

Tommaso Colombo · **Simone Sagratella**

Received: date / Accepted: date

Abstract Distributed and parallel algorithms have been frequently investigated in the recent years, in particular in applications like machine learning. Nonetheless, only a small subclass of the optimization algorithms in the literature can be easily distributed, for the presence, e.g., of coupling constraints that make all the variables dependent from each other with respect to the feasible set. Augmented Lagrangian methods are among the most used techniques to get rid of the coupling constraints issue, namely by moving such constraints to the objective function in a structured, well-studied manner. Unfortunately, standard augmented Lagrangian methods need the solution of a nested problem by needing to (at least inexactly) solve a subproblem at each iteration, therefore leading to potential inefficiency of the algorithm. To fill this gap, we propose an augmented Lagrangian method to solve convex problems with linear coupling constraints that can be distributed and requires a single gradient projection step at every iteration. We give a formal convergence proof to at least ε-approximate solutions of the problem and a detailed analysis of how the parameters of the algorithm influence the value of the approximating parameter ε. Furthermore, we introduce a distributed version of the algorithm allowing to partition the data and perform the distribution of the computation in a parallel fashion.

Keywords Nonlinear optimization · Parallel algorithms · Distributed algorithms · Lagrangian methods

1 Introduction

Distributed and parallel algorithms to solve optimization problems have gained more and more attention in the last decades. These have acquired an even larger importance since the amount of data available has grown exponentially. Beyond the evident effectiveness from a computational time perspective, some of these also allow the distribution of the data. Partitioning the data in blocks and assigning those blocks to separate processes is key not only for speeding up the computation, but also e.g. for privacy reasons [7,11, 20,37–40].

Unfortunately, only a small subset of the optimization algorithms in the literature presents the desirable property of being distributable. Moreover, such distributed algorithms usually assume that the feasible set is representable as a cartesian product of feasible sets, so that the variables are block-separable and each

The work of the authors was partially supported by the grant: "Finanziamenti di ateneo per la ricerca scientifica 2018" n. RP11816432902D1E, Sapienza University of Rome

T. Colombo
Department of Computer, Control and Management Engineering Antonio Ruberti
Sapienza University of Rome
E-mail: colombo@diag.uniroma1.it

S. Sagratella
Department of Computer, Control and Management Engineering Antonio Ruberti
Sapienza University of Rome
E-mail: sagratella@diag.uniroma1.it

block has a private feasible set independent from the other blocks, even if the objective function depends on all the blocks [6,8,18]. However, these assumptions are not satisfied in many real big data applications like, e.g., support vector machines training. In particular, in the dual formulation of the training problem of support vector machines the presence of a linear equality constraint (i.e. a coupling constraint) over all the variables makes the distribution of the data uneasy [26,28,30,31,33].

A solution to the problem of the coupling constraints is to employ methods that allow including such constraints in the objective function, thus making the feasible set block-separable. The most well-known of such methods is probably the augmented Lagrangian method [5,12,13,27]. We must mention that there are other efficient decomposition methods, like column generation and Frank-Wolfe methods, which use fast subsolvers and have been effectively employed in machine learning applications [19,21,23,24,29,32].

Thanks to Lagrangian duality theory the coupling constraints can be included in the objective function and the problem solved by alternating a minimization over the primal variables and a maximization step over the Lagrangian multipliers [5,6]. Augmented Lagrangian methods, although much used in practice, need solving (at least inexactly) a subproblem over the primal variables at each iteration. Solving a subproblem by using an iterative scheme at each iteration may be really inefficient. In these regards, one may think to modify the standard version of the augmented Lagrangian method [5] by employing a single minimization step instead of performing a minimization with respect to the primal variables. We show that this is not viable in general and we provide a counterexample showing the ineffectiveness of such naive modification.

To fill this gap, we propose a modified augmented Lagrangian method to solve convex problems with linear coupling constraints. This new method turns out to be distributable and at every iteration performs a single gradient projection step with respect to the primal variables. We give a formal convergence proof to at least ε-approximate solutions of the problem, namely points that have an objective value better or equal than the optimal one and violate the equality constraints by a quantity lower or equal than ε. A detailed analysis of how the parameters of the algorithm influence the value of the approximation ε is also included. Furthermore, we sketch a distributed version of the algorithm, underlining how the data can be partitioned and assigned to separate processes that perform the computations independently, with minimal communication at the end of each iteration.

In Section 2 we detail the motivation for this work, in particular we define a general distributed scheme and we show how standard augmented Lagrangian methods can hardly be distributed; in Section 3 we present Algorithm 2 and we state the main convergence results in Theorem 1. In Section 4 we explore some technical results that are fundamental for the proof of Theorem 1. In particular we define a nonsmooth function ϕ whose properties are explored in Propositions 1 and 2; in Proposition 3 we show that any minimal point of ϕ is an ε-approximate solution of the problem; in Proposition 4 we show a strategy for updating the parameters of the algorithm leading to optimal solutions of the problem. In Section 5 we give the proof of Theorem 1. Finally, in Section 6 we sketch a distributed version of the algorithm and we discuss some conclusions in Section 7.

2 Motivation

We define the general formulation of the convex programming problem with linear coupling constraints:

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{s.t.} \quad & h(x) \triangleq Ax - b = 0 \\ & x \in X, \end{aligned} \tag{1}$$

where $f \in C^{1,1}$ is convex with Lipschitz constant of ∇f equal to $\overline{L}$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, and $X \subseteq \mathbb{R}^n$ is convex, compact and encompasses a structure that is separable in N blocks, namely

$$X \triangleq \prod_{\nu=1}^{N} X_\nu,$$

where $X_\nu \subseteq \mathbb{R}^{n_\nu}$, with $n \triangleq n_1 + \cdots + n_N$. For the sake of notational simplicity we denote the feasible set of Problem (1) with S. We observe that the set S is in general not separable due to the presence of the constraints h that may tie together variables of different blocks. For this reason we refer to h as the coupling constraints. We underline that the case with linear inequality constraints can be included in the above framework by simply adding slack variables.

We assume that S is non-empty. Therefore there exists an optimal solution $x^\star$ such that

$$x^\star \in S, \qquad f(x^\star) \leq f(x) \quad \forall\, x \in S.$$

To compute $x^\star$ one can resort to several numerical methods. We concentrate on (synchronous) distributed algorithms which structure is described in Figure 2. Namely, in a distributed algorithm at every iteration the computation is split into N independent blocks that can be performed by different parallel processes, and a final communication phase allows synchronization among the processes.

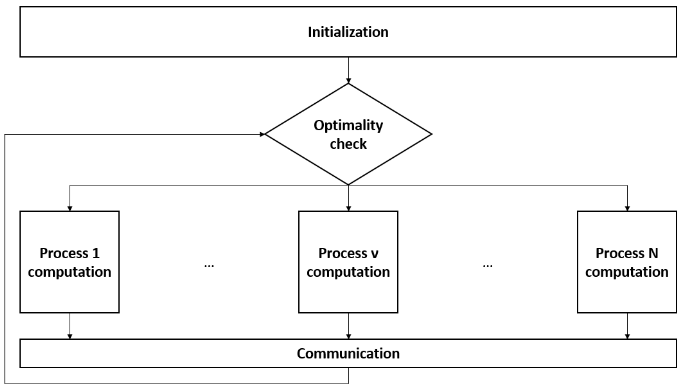


Fig. 1 General distributed algorithmic scheme

Such a distribution of the computation is even more relevant when it makes it possible to partition the data defining the problem into N blocks that need to be known only by the respective processes. This is the case when the data comes from several sources and can not be stored in a unique place, for example when there exist privacy issues or the dimension of the data is huge.

One of the most well-known methods to solve Problem (1) is the classical gradient projection algorithm, see e.g. [4]:

$$x^{k+1} = \mathscr{P}_S\left[x^k - \alpha_k \nabla f(x^k)\right], \tag{2}$$

where α_k is a positive stepsize and $\mathscr{P}_S(z)$ denotes the projection of z over the convex set S. To guarantee the convergence of the algorithm defined by iteration (2), one can compute α_k by employing a linesearch procedure, a diminishing stepsize rule, or by using a fixed, sufficiently small, stepsize $\alpha_k = \alpha \in \big(0, (2/\overline{L})\big)$ [2]. Unfortunately this method does not fall into the class of distributed algorithms because the projection on S, in general, cannot be decomposed with respect to the N blocks, see Appendix A.

Note however that in the special case in which there are no coupling constraints (i.e. $m = 0$) this task is easier. In fact in this case the projection can be performed in a distributed fashion and this scheme is a proper distributed algorithm, since the projection on S turns out to be a projection on the N sets X_ν that are separate. With the additional assumption that the objective function $f(x) = \frac{1}{2}x^TQx + c^Tx$ is quadratic, then the situation is even more favourable since also the data can be partitioned. Specifically, referring to the general scheme in Figure 2, the vector of variables $x \in \mathbb{R}^n$ can be partitioned into N blocks: $x = \big(x_{(\nu)}\big)_{\nu=1}^N$, with $x_{(\nu)} \in \mathbb{R}^{n_\nu}$, $\nu = 1, \ldots, N$. Accordingly, $\nabla f(x) = \big(\nabla f(x)_{(\nu)}\big)_{\nu=1}^N$. In the situation cited above where $m = 0$ and f is quadratic, the generic process ν performs the following steps at any iteration k:

(i) compute $\nabla f(x^k)_{(\nu)} = \sum\limits_{\xi=1}^{N} a_{(\nu)}^{\xi,k} + c_{(\nu)}$ (where all the $a_{(\nu)}^{\xi,k}$ come from the communication phase at step (iii));

(ii) compute $x_{(\nu)}^{k+1} = \mathscr{P}_{X_\nu}\left[x_{(\nu)}^k - \alpha \nabla f(x^k)_{(\nu)}\right]$;
(iii) compute $a^{\nu,k+1} = Q_{*(\nu)}\, x_{(\nu)}^{k+1}$ and broadcast.

We observe that in this scheme the stepsize is fixed, and the computations up to process ν are independent from those of the other blocks. Process ν only needs to know the columns $Q_{*(\nu)}$ of the matrix Q corresponding to the ν-th block of variables, the ν-th block $c_{(\nu)}$ of vector c, and the set X_ν. Thus the data defining the problem can be partitioned and then distributed to the processes in the initialization phase. This plays a fundamental role in applications where either different pieces of information are available only to different processes or if the knowledge of such data requires significant computational burdens. The latter is the case of support vector machines training, because in its dual quadratic formulation a column of the matrix Q requires in general $\mathscr{O}(n^2)$ nonlinear calculations [28,30]. Nonetheless, we underline that the above scheme is not directly applicable to the support vector machines quadratic formulation due to the presence of a coupling linear constraint ($m = 1$) [31].

In the general case where $m > 0$, as explained above and in Appendix A, such simple distributed scheme with partitioned data cannot be employed due to the presence of the coupling constraints, which make S non-separable. One possible solution is to move the coupling constraints from the feasible set to the objective function, thus making the feasible set S separable since it reduces to X. This can be done by employing a Lagrangian method (see e.g. [4–6]). The well-known augmented Lagrangian function can be defined for Problem (1) as

$$\mathscr{L}_\rho(x,\mu) \triangleq f(x) + \mu^T h(x) + \frac{1}{2}\rho \|h(x)\|^2, \tag{3}$$

where $\mu \in \mathbb{R}^m$ is the vector of Lagrangian multipliers, $\rho > 0$ is the parameter of the penalty term, and $\|\cdot\|$ denotes the euclidean norm. We recall that, in our framework, assuming the second-order sufficient conditions to hold at the optimal point of the problem with multipliers μ^*, then a finite ρ^* exists such that, for all $\rho \geq \rho^*$, the augmented Lagrangian with $\mu = \mu^*$ turns out to be an exact penalty function, see [34]. In a general augmented Lagrangian method (see e.g. [5] for an efficient version of such method) a solution of Problem (1) can be found by using the scheme described in Algorithm 1.

Algorithm 1: Basic augmented Lagrangian method

```
  Data: $x^0 \in X$, $\mu^0 \in C$, $\rho_0 > 0$, $\{\beta_k\} > 0$;
1 for $k = 0, 1, \dots$ do
2     $x^{k+1} = \arg\min_{x \in X} \mathscr{L}_{\rho_k}(x, \mu^k)$;
3     $\mu^{k+1} = \mathscr{P}_C\left[\mu^k + \beta_k \nabla_\mu \mathscr{L}_{\rho_k}(x^{k+1}, \mu^k)\right]$;
4     $\rho_{k+1} \geq \rho_k$.
5 end
```

Algorithm 1 performs: at step 2 a minimization of the augmented Lagrangian function over X with respect to the variables x, at step 3 an ascent step for the augmented Lagrangian function with respect to μ using the stepsize β_k with a projection over a nonempty convex compact set C, and at step 4 a possible increase of the penalty parameter ρ. Convergence of such general scheme can be obtained by suitably choosing the sequence of positive steplengths β_k or by updating the penalty parameter ρ according to certain rules usually tied to the violation of the coupling constraints given by h. Actually, the minimization at step 2 can be also performed inexactly by computing x^{k+1} such that $\mathscr{L}_{\rho_k}(x^{k+1},\mu^k) \leq \mathscr{L}_{\rho_k}(x,\mu^k) + \varepsilon_k$, for all $x \in X$, with $\varepsilon_k \downarrow 0$, maintaining convergence properties.

We notice that any single (inexact) minimization at step 2 of Algorithm 1 could be distributed by employing a scheme similar to the one described above for the case with $m = 0$. In fact in this setting one can use e.g. a gradient projection method over X that is separable in N blocks. Therefore, the overall Algorithm 1, executing iteratively a distributed minimization over the variables x, can be viewed itself as a distributed algorithm (see e.g. [31,37]). Nevertheless, solving an (inexact) optimization problem at each iteration may be numerically inefficient. The alternating direction method of multipliers suffers from the same inefficiency since it relies on a similar scheme where an (inexact) minimization with respect to the primal variables x must be performed at each iteration (see e.g. [6,22]). One may try to substitute the (inexact) minimization at Step 2 of Algorithm 1 with one single gradient projection step and a stepsize updating rule such that the descent of the objective function at each iteration is guaranteed (a tipical choice is to set the stepsize equal to the inverse of the Lipschitz constant of the gradient with respect to x of the Lagrangian function). The following counterexample shows how this does not work in general.

Example 1 Let us consider the problem

$$\begin{aligned} &\text{minimize} \;\; x_1 \\ &\text{s.t.} \;\; h(x) = x_1 - x_2 = 0 \\ &\qquad x \in [-1,1]^2, \end{aligned}$$

which unique solution is $x^\star = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$.

It is proved in [5] that Algorithm 1, with $C = \{0\}$ and an updating rule for the penalty parameter $\rho_{k+1} = 2\rho_k$ if $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$, $\rho_{k+1} = \rho_k$ otherwise, where $\tau \in (0,1)$, globally converges to a solution of the problem. In this case μ vanishes and the Lagrangian function is

$$\mathscr{L}_{\rho_k}(x) = x_1 + \frac{\rho_k}{2}\|x_1 - x_2\|^2,$$

while its gradient can be written as

$$\nabla\mathscr{L}_{\rho_k}(x) = \begin{pmatrix} 1 + \rho_k(x_1 - x_2) \\ \rho_k(x_2 - x_1) \end{pmatrix}.$$

Let us see what happens by substituting step 2 with only one gradient projection step and a stepsize updating rule guaranteeing the descent of the objective function at each iteration. The modified version of the algorithm is then the following, for any iterate $k \geq 0$:

(i) $\alpha_k = \frac{1}{L + \rho_k\|A\|^2} = \frac{1}{2\rho_k}$;
(ii) $x^{k+1} = \mathscr{P}_{[-1,1]^2}\left[x^k - \alpha_k \nabla\mathscr{L}_{\rho_k}(x^k)\right]$;
(iii) $\rho_{k+1} = 2\rho_k$ if $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$; $\rho_{k+1} = \rho_k$ otherwise.

In particular we choose $\tau \in \left(0, \frac{1}{4}\right)$, $\rho_0 = 2$ and $x^0 = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$. By simple calculations, we have, for any k, that condition $\|h(x^k)\|^2 > \tau\|h(x^{k-1})\|^2$ is always satisfied and:

$$\begin{aligned} \rho_k &= 2\rho_{k-1} = 2^{k+1}, \\ \nabla\mathscr{L}_{\rho_k}(x^k) &= \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \\ x^k &= \begin{pmatrix} 0 \\ \frac{1}{2^{k+1}} \end{pmatrix}, \\ h(x^k) &= -x_2 = -\frac{1}{2^{k+1}}. \end{aligned}$$

But this implies

$$\lim_{k\to\infty} x^k = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \neq x^\star,$$

that is, this modified version of the algorithm does not converge to a solution of the problem. □

To fill this gap, in Section 3 we propose a nontrivial and globally convergent modification of Algorithm 1 to solve Problem (1) that is still distributable, but requires only a single gradient projection step with respect to the variables x at any iteration. The main idea underlying this new algorithm consists in avoiding the possible out of control growth of the penalty parameter ρ, that is evident in Example 1, by introducing an upper bound on this parameter. Thus, in case the penalty parameter reaches its upper limit, the algorithm computes an approximate solution of the problem by performing gradient-like minimization steps for a suitable merit function that is introduced in Section 4.

3 A gradient projection method for the augmented Lagrangian reformulation

In this section we propose Algorithm 2, that is a gradient projection algorithm to solve the augmented Lagrangian reformulation of Problem (1). Algorithm 2 is characterized by two parameters, $\widehat{\mu} > 0$, used to define the compact set for the multipliers μ^k, and $\widehat{\rho} > 0$, the upper bound for ρ_k, which will play a key role in the effectiveness of computing a solution of Problem (1).

Algorithm 2: Gradient projection augmented Lagrangian method

```
   Data: $\widehat{\rho} > 0$, $\widehat{\mu} > 0$, $x^0 \in X$, $\mu^0 \in [-\widehat{\mu}, \widehat{\mu}]^m$, $\rho_0 \in (0, \widehat{\rho})$, $\gamma > 0$, $\delta > 0$, $\tau \in (0,1)$, $\widehat{k} = 0$;
1  for $k = 0, 1, \ldots$ do
2      $\alpha_k = \frac{1}{\overline{L} + \rho_k \|A\|^2 + \gamma(k - \widehat{k})}$ ;
3      $x^{k+1} = \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\rho_k}(x^k, \mu^k)]$ ;
4      if $\rho_k < \widehat{\rho}$ then
5          $\mu^{k+1} \in [-\widehat{\mu}, \widehat{\mu}]^m$;
6          $\widehat{k} = \widehat{k} + 1$ ;
7      else
8          $\mu_i^{k+1} = \begin{cases} -\widehat{\mu}, & i \,:\, h_i(x^{k+1}) < 0 \\ \widehat{\mu}, & \textit{otherwise} \end{cases}, \quad i = 1, \ldots, m;$
9      end
10     $\rho_{k+1} \in \begin{cases} [\min\{\rho_k + \delta, \widehat{\rho}\}, \widehat{\rho}], & \|h(x^{k+1})\| > \tau \|h(x^k)\| \\ \{\rho_k\}, & \textit{otherwise.} \end{cases}$
11 end
```

Algorithm 2 enjoys the following two nice properties:

- the computation can be distributed and, moreover, if f is quadratic the data can be distributed too (see Section 2);
- at each iteration k, only a gradient projection step over x is performed.

At step 2 the stepsize α_k is updated, exploiting the value of $\widehat{k}$, such that:

(i) on the one hand, whenever $\widehat{k} = k$ (i.e. $\rho_k < \widehat{\rho}$), it dynamically estimates the quantity $1/L_{\mathscr{L}}^k$, where $L_{\mathscr{L}}^k = \overline{L} + \rho_k \|A\|^2$ (with $\overline{L}$ being the Lipschitz constant of ∇f) is the Lipschitz constant of $\nabla_x \mathscr{L}_{\rho_k}(x^k, \mu^k)$ over X;
(ii) on the other hand, when $\widehat{k} < k$ (i.e. $\rho_k = \widehat{\rho}$), it ensures that $\alpha_k \downarrow 0$ and α_k is squared summable, but not summable, namely

$$\sum_{k=0}^{\infty} \alpha_k^2 < \infty, \quad \sum_{k=0}^{\infty} \alpha_k = \infty. \tag{4}$$

It will be clear by the following sections that the convergence of the algorithm is based on the stepsize updating rule property defined in (4).

At step 3 a single gradient projection step is performed to update x^k. The multipliers vector μ^k is updated either at step 5 if $\rho_k < \widehat{\rho}$, or at step 8 otherwise. We observe that in the former case μ^{k+1} can be any element in the compact set $M_{\widehat{\mu}} \triangleq [-\widehat{\mu}, \widehat{\mu}]^m$, i.e. μ^k can be updated like in Algorithm 1. In the latter case, μ^{k+1} is such that

$$\mu^{k+1} \in \arg\max_{\mu \in M_{\widehat{\mu}}} \mathscr{L}_{\rho_k}(x^{k+1}, \mu).$$

Throughout the paper we will equivalently refer to the set $M_{\widehat{\mu}}$ with the abbreviation M when the dependence on $\widehat{\mu}$ will not be of interest. Finally, at step 10 the penalty parameter ρ_k, bounded above by $\widehat{\rho} > 0$, is increased if a sufficient decrease in the violation of the equality constraints is not achieved.

We observe that such an iterative scheme implies the boundedness of the generated sequence $\{(x^k, \mu^k, \rho_k)\}$. In Theorem 1 we show that Algorithm 2 converges in the worst case to an ε-approximate solution of Problem (1), i.e. a point $\overline{x}$ such that

$$\overline{x} \in X, \; \|h(\overline{x})\| \leq \varepsilon, \qquad f(\overline{x}) \leq f(x) \quad \forall\, x \in S.$$

On the other hand, if $\widehat{\rho}$ is large enough, in the sense that the penalty parameter remains strictly below $\widehat{\rho}$, the algorithm provably converges to a solution $x^\star$ of Problem (1), that is a 0-approximate solution. We underline that $f(\bar{x}) \leq f(x^\star)$, namely the value of the objective function at $\bar{x}$ is a lower bound of the optimal value. Furthermore, we can theoretically show (see Theorem 1) that the value of the feasibility relaxation parameter ε can be controlled by suitably choosing the parameters $\widehat{\mu}$ and $\widehat{\rho}$. It is expedient to define the following finite quantities:

$$f^{min} \triangleq \min_{x\in X} f(x), \qquad f^{max} \triangleq \max_{x\in X} f(x).$$

Theorem 1 *Let $\{(x^k,\mu^k,\rho_k)\}$ be the sequence generated by Algorithm 2 and $(\bar{x},\bar{\mu},\bar{\rho})$ be any of its limit points. Then we have two cases:*

1. *if $\bar{\rho} < \widehat{\rho}$, then $\bar{x}$ is a solution of Problem* (1)*;*
2. *otherwise, $\bar{\rho} = \widehat{\rho}$ and $\bar{x}$ is an ε-approximate solution of Problem* (1) *with* $\varepsilon = \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$.

The proof of Theorem 1 is postponed to Section 5 since it requires some further theoretical developments, which are described in Section 4. In the light of Theorem 1, we observe that Algorithm 2 is in general inexact and the quality of the produced solution depends on the values of $\widehat{\mu}$ and $\widehat{\rho}$.

In Example 2 we show that the problem given in Example 1 is actually solved by employing Algorithm 2.

Example 2 Consider again the problem in Example 1. Let us assume again that the updating rule for the penalty parameter is $\rho_{k+1} = 2\rho_k$ if $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$, where $\tau \in \left(0,\frac{1}{4}\right)$, $\rho_{k+1} = \rho_k$ otherwise. In this case we employ Algorithm 2 and we choose $\widehat{\mu} = 0$, $\widehat{\rho} = 2$, $\gamma = \frac{1}{10}$, $\delta = 1$. The Lagrangian function and its gradient are the same of Example 1.

Algorithm 2 performs the following updates for $k \geq 0$:

(i) $\alpha_k = \frac{1}{\bar{L}+\rho_k\|A\|^2+\gamma(k-\widehat{k})} = \frac{1}{2\rho_k+\gamma(k-\widehat{k})}$;
(ii) $x^{k+1} = \mathscr{P}_{[-1,1]^2}\left[x^k - \alpha_k \nabla \mathscr{L}_{\rho_k}(x^k)\right]$;
(iii) $\rho_{k+1} = \min\{2\rho_k,\widehat{\rho}\}$ if $\|h(x^{k+1})\|^2 > \tau\|h(x^k)\|^2$, $\rho_{k+1} = \rho_k$ otherwise.

We choose again $\rho_0 = 2$ and $x^0 = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$, that is the same starting point of Example 1 and for which the naive modification of Algorithm 1 does not converge. Then, by simple calculations, we have $x^1 = \begin{pmatrix} 0 \\ \frac{1}{4} \end{pmatrix}$ and, for any $k \geq 1$, that $\rho_k = \widehat{\rho} = 2$, $\widehat{k} = 0$ and $\alpha_k = \frac{10}{40+k}$. The sequence produced by the algorithm for $k = 2,\dots,9$ is the following:

$$x^k = \begin{pmatrix} -\frac{1}{2}\sum_{i=1}^{k-1}\alpha_i \\ \frac{1}{4} - \frac{1}{2}\sum_{i=1}^{k-1}\alpha_i \end{pmatrix}.$$

In $k = 10$, x_1 reaches its lower bound:

$$x^{10} = \begin{pmatrix} -1 \\ \frac{1}{4} - \frac{1}{2}\sum_{i=1}^{9}\alpha_i \end{pmatrix}.$$

For $k \geq 11$,

$$x^k = \begin{pmatrix} -1 \\ \left(\frac{1}{4} - \frac{1}{2}\sum_{i=1}^{9}\alpha_i\right)\left(\prod_{i=10}^{k-1}(1-2\alpha_i)\right) - \sum_{i=10}^{k-1} 2\alpha_i \left(\prod_{j=i+1}^{k-1}(1-2\alpha_j)\right) \end{pmatrix}.$$

Such sequence $\{x^k\}$ converges (from above) to the optimal solution $x^\star = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$. □

We remark that the new method presented in this work is a subgradient-type scheme for which the convergence speed depends strongly on the stepsize. In particular, if the Lipschitz constant $\bar{L}$ or the parameter $\widehat{\rho}$ are large, the stepsize can be very small. E.g. in the presented Example 2 the convergence is slow.

4 Technical results

In order to prove part 2 of Theorem 1, we introduce and analyze a nonsmooth value function $\phi_{\widehat{\mu},\widehat{\rho}}$ that is related to the Lagrangian function (3) and to Problem (1):

$$\phi_{\widehat{\mu},\widehat{\rho}}(x) \triangleq f(x) + \widehat{\mu}\|h(x)\|_1 + \frac{\widehat{\rho}}{2}\|h(x)\|^2. \tag{5}$$

Notice that it depends on the same parameters $\widehat{\mu}$ and $\widehat{\rho}$ defined in Algorithm 2, and is useful for the subsequent analysis.

First of all we analyze the properties of the nonsmooth value function $\phi_{\widehat{\mu},\widehat{\rho}}$.

Proposition 1 *Let $\widehat{\mu},\widehat{\rho} \geq 0$. The function $\phi_{\widehat{\mu},\widehat{\rho}}$ is convex and locally Lipschitz continuous.*

Proof *It is well known that a sum of convex functions is convex. Thus the proof follows by noticing that f is convex by assumption, $\|h(x)\|_1 = \|Ax-b\|_1 = \sum_{i=1}^{m} |A_{i*}x - b_i| = \sum_{i=1}^{m} \max\{A_{i*}x-b_i, -A_{i*}x+b_i\}$ is convex, and $\|h(x)\|^2 = x^TA^TAx - 2b^TAx + b^Tb$ is convex. The proof of the local Lipschitz continuity follows by recalling that any convex function is locally Lipschitz continuous, see e.g. [10, Proposition 2.2.6].* □

Note that the role of the linearity of h is fundamental for the convexity of the nonsmooth function $\phi_{\widehat{\mu},\widehat{\rho}}$.

Proposition 2 *The subdifferential $\partial\phi_{\widehat{\mu},\widehat{\rho}}(x)$ is non-empty and*

$$\nabla f(x) + \widehat{\mu}\sum_{i=1}^{m}\xi_i + \widehat{\rho}\nabla h(x)h(x) \in \partial\phi_{\widehat{\mu},\widehat{\rho}}(x),$$

with

$$\xi_i \in \begin{cases} \{A_{i*}^T\}, & \text{if } A_{i*}x - b_i > 0,\\ \{-A_{i*}^T\}, & \text{if } A_{i*}x - b_i < 0,\\ conv\{-A_{i*}^T, A_{i*}^T\}, & \text{if } A_{i*}x - b_i = 0, \end{cases}$$

where $conv\{x,y\}$ denotes the convex hull of the vectors $x,y \in \mathbb{R}^n$.

Proof *From [10, Corollary 3], the subdifferential of the sum of convex functions equals the sum of the subdifferentials of the functions. Therefore we can write*

$$\partial\phi_{\widehat{\mu},\widehat{\rho}}(x) = \partial f(x) + \widehat{\mu}\partial\|h(x)\|_1 + \frac{\widehat{\rho}}{2}\partial\|h(x)\|^2,$$

where

$$\partial f(x) = \nabla f(x), \qquad \partial\|h(x)\|^2 = 2\nabla h(x)h(x) = 2A^T(Ax-b),$$

$$\partial\|h(x)\|_1 = \left(\sum_{i=1}^{m}\partial|A_{i*}x - b_i|\right) = \left(\sum_{i=1}^{m}\partial\max\{A_{i*}x-b_i, -A_{i*}x+b_i\}\right) \ni \sum_{i=1}^{m}\xi_i,$$

with

$$\xi_i \in \begin{cases} \{A_{i*}^T\}, & A_{i*}x - b_i > 0,\\ \{-A_{i*}^T\}, & A_{i*}x - b_i < 0,\\ conv\{-A_{i*}^T, A_{i*}^T\}, & A_{i*}x - b_i = 0, \end{cases}$$

(see e.g. [35, Exercise 8.31]). □

The following proposition proves that minimal points of $\phi_{\widehat{\mu},\widehat{\rho}}$ over X are ε-approximate solutions of Problem (1) with $\varepsilon = \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$.

Proposition 3 *Given $\widehat{\mu} > 0$ and $\widehat{\rho} > 0$, for any minimal point $\bar{x}$ of $\phi_{\widehat{\mu},\widehat{\rho}}$ over X, it holds that:*

1. $f(\bar{x}) \leq f(x^\star)$,

2. $\|h(\bar{x})\| \leq \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$,

where $x^\star$ is a solution of Problem (1), $f^{min} = \min_{x\in X} f(x)$ *and* $f^{max} = \max_{x\in X} f(x)$.

Proof *We first show the first inequality. It holds that*

$$f(\bar{x}) \leq f(\bar{x}) + \widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(x^\star) + \widehat{\mu}\|h(x^\star)\|_1 + \frac{\widehat{\rho}}{2}\|h(x^\star)\|^2 = f(x^\star),$$

by recalling that $\widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \geq 0$*, that* $x^\star \in X$ *and that* $h(x^\star) = 0$*. This proves the first assertion of this proposition.*

Let us now prove the second assertion. Let $\widetilde{x} \in X$ *such that* $h(\widetilde{x}) = 0$*, namely* $\widetilde{x}$ *is any feasible point of Problem* (1)*. Then we can write*

$$f^{min} \leq f(\bar{x}) \quad \textit{and} \quad f(\widetilde{x}) + \widehat{\mu}\|h(\widetilde{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\widetilde{x})\|^2 = f(\widetilde{x}) \leq f^{max}.$$

From the two inequalities above, we can write

$$f^{min} + \widehat{\mu}\|h(\bar{x})\|_1 \leq f(\bar{x}) + \widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(\widetilde{x}) + \widehat{\mu}\|h(\widetilde{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\widetilde{x})\|^2 \leq f^{max},$$

implying

$$\|h(\bar{x})\| \leq \|h(\bar{x})\|_1 \leq \frac{f^{max} - f^{min}}{\widehat{\mu}}.$$

On the other hand,

$$f^{min} + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(\bar{x}) + \widehat{\mu}\|h(\bar{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\bar{x})\|^2 \leq f(\widetilde{x}) + \widehat{\mu}\|h(\widetilde{x})\|_1 + \frac{\widehat{\rho}}{2}\|h(\widetilde{x})\|^2 = f(\widetilde{x}) \leq f^{max},$$

from which

$$\|h(\bar{x})\|^2 \leq \frac{f^{max} - f^{min}}{\widehat{\rho}},$$

which completes the proof. □

Furthermore, if a minimal point of $\phi_{\widehat{\mu},\widehat{\rho}}$ is also feasible for Problem (1), then it is a solution of Problem (1).

Corollary 1 *Let the same setting of Proposition 3 applies. Then, if* $\bar{x}$ *is such that* $\|h(\bar{x})\| = 0$, $\bar{x}$ *is a solution of Problem* (1).

Proof *From Proposition 3, it holds that*

$$f(\bar{x}) \leq f(x^*),$$

where x^* *is a solution of Problem* (1)*. The proof follows by recalling the hypothesis, from which we have* $h(\bar{x}) = 0$. □

We now want to provide a counterexample of how an ε-approximate solution of Problem (1) may not be a solution of $\phi_{\widehat{\mu},\widehat{\rho}}$ over X, that is, the vice versa of Proposition 3 does not hold in general.

Example 3 Let us consider the problem

$$\begin{aligned} &\text{minimize} \;\; x^2 \\ &\text{s.t.} \;\; h(x) = x - 1 = 0 \\ &\qquad x \in [0,2], \end{aligned}$$

which trivial solution is $x^\star = 1$. By definition, this is an ε-approximate solution for any $\varepsilon > 0$.

The corresponding $\phi_{\widehat{\mu},\widehat{\rho}}$ if we fix $\widehat{\mu} = 1, \widehat{\rho} = 0$ is

$$\phi_{1,0}(x) = x^2 + |x-1|,$$

which unique solution over $[0,2]$ is $\bar{x} = \frac{1}{2}$, that is different from $x^\star$.

Note that by letting $\widehat{\mu}$ grow to infinity, the solution of $\phi_{\widehat{\mu},\widehat{\rho}}$ gets closer and closer to the minimizer of $|x-1|$, which is the solution of the original problem. This will be better formalized in the next proposition. □

Proposition 3 shows that a minimizer of $\phi_{\widehat{\mu},\widehat{\rho}}$ is an ε-approximate solution of Problem (1). Specifically, the bigger the parameters $\widehat{\mu}$ and $\widehat{\rho}$ are, the smaller the value of the feasibility relaxation parameter ε is. The next proposition formally states that if one of the parameters goes to infinity, then ε goes to zero.

Proposition 4 *Let $\{\bar{x}^k\}$ be a sequence of minimizers of $\phi_{\widehat{\mu}^k,\widehat{\rho}^k}(x)$ over X. If either $\lim_{k\to\infty} \widehat{\mu}^k = \infty$ or $\lim_{k\to\infty} \widehat{\rho}^k = \infty$, then any accumulation point $\bar{x}$ of the sequence $\{\bar{x}^k\}$ is a solution of Problem* (1).

Proof *Let us assume, without loss of generality, that the first limit applies. By Proposition 3, it holds for any k that*

$$\bar{x}^k \in X, \quad f(\bar{x}^k) \leq f(x^*), \quad \|h(\bar{x}^k)\| \leq \frac{f^{max} - f^{min}}{\widehat{\mu}^k}.$$

By the continuity of $\|h(\bar{x}^k)\|$, we can write

$$\|h(\bar{x})\| = \lim_{k\to\infty} \|h(\bar{x}^k)\| \leq \lim_{k\to\infty} \frac{f^{max} - f^{min}}{\widehat{\mu}^k} = 0$$

and therefore $h(\bar{x}) = 0$. Similarly, by recalling that $x^\star$ is a solution of Problem (1), *we can write $f(\bar{x}) \leq f(x^\star)$.* □

We are now ready to give the proof of Theorem 1.

5 Proof of Theorem 1

In this section we give the proof of Theorem 1.

Proof We start by proving the first assertion of the theorem, namely that if $\bar{\rho} < \widehat{\rho}$ then $\bar{x}$ is a solution of Problem (1).
Recalling step 10 of Algorithm 2, if $\bar{\rho} < \widehat{\rho}$ then a $\bar{k} \geq 0$ exists such that for all $k \geq \bar{k}$

$$\|h(x^{k+1})\| \leq \tau \|h(x^k)\| < \|h(x^k)\|, \tag{6}$$

by recalling that $\tau \in (0,1)$. This yields, together with the positiveness of the sequence $\{\|h(x^k)\|\}$, that

$$\lim_{k\to\infty} \|h(x^k)\| = \bar{h} \geq 0.$$

Taking the limit for $k \to \infty$ in (6), we get

$$\bar{h} \leq \tau \bar{h},$$

which, since $\tau \in (0,1)$, is true only if $\bar{h} = 0$. Therefore, by the continuity of h, we have proved that

$$h(\bar{x}) = 0. \tag{7}$$

Moreover, recalling a well-known property of the projection operator, it holds for all $z \in \mathbb{R}^n$

$$(y - \mathscr{P}_X[z])^T (z - \mathscr{P}_X[z]) \leq 0 \quad \forall y \in X.$$

Therefore we can write

$$\left(x^k - \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\bar{\rho}}(x^k, \mu^k)]\right)^T \left(x^k - \alpha_k \nabla_x \mathscr{L}_{\bar{\rho}}(x^k, \mu^k) - \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\bar{\rho}}(x^k, \mu^k)]\right) \leq 0,$$

and by expanding the products

$$\left\|x^k - \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k,\mu^k)]\right\|^2 + \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k,\mu^k)^T \left(\mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k,\mu^k)] - x^k\right) \le 0,$$

from which, by recalling that $x^{k+1} = \mathscr{P}_X[x^k - \alpha_k \nabla_x \mathscr{L}_{\overline{\rho}}(x^k,\mu^k)]$, we get

$$\nabla_x \mathscr{L}_{\overline{\rho}}(x^k,\mu^k)^T \left(x^{k+1} - x^k\right) \le -\frac{1}{\alpha_k}\left\|x^{k+1} - x^k\right\|^2. \tag{8}$$

The descent lemma [4, Lemma 2.1] and (8) therefore yield

$$\begin{aligned}\mathscr{L}_{\overline{\rho}}(x^{k+1},\mu^k) - \mathscr{L}_{\overline{\rho}}(x^k,\mu^k) &\le \nabla_x \mathscr{L}_{\rho_k}(x^k,\mu^k)^T(x^{k+1}-x^k) + \frac{L^k_{\mathscr{L}}}{2}\|x^{k+1}-x^k\|^2 \\ &\le -\left(\frac{1}{\alpha_k} - \frac{L^k_{\mathscr{L}}}{2}\right)\|x^{k+1}-x^k\|^2\end{aligned}$$

and consequently

$$f(x^{k+1}) - f(x^k) + \mu^{k^T}(h(x^{k+1}) - h(x^k)) + \frac{\overline{\rho}}{2}(\|h(x^{k+1})\|^2 - \|h(x^k)\|^2) \le -\frac{\overline{L}+\overline{\rho}\|A\|^2}{2}\|x^{k+1}-x^k\|^2.$$

By taking the limit for $k \to \infty$ of the above inequality we get

$$\lim_{k\to\infty}(f(x^{k+1}) - f(x^k)) + \left(\lim_{k\to\infty}\mu^k\right)^T(\overline{h}-\overline{h}) + \frac{\overline{\rho}}{2}(\|\overline{h}\|^2 - \|\overline{h}\|^2) \le -\frac{\overline{L}+\overline{\rho}\|A\|^2}{2}\lim_{k\to\infty}\|x^{k+1}-x^k\|^2,$$

from which we can write

$$\lim_{k\to\infty}\left(f(x^{k+1}) - f(x^k)\right) \le 0.$$

This yields

$$\lim_{k\to\infty} f(x^k) = f(\overline{x}) = \overline{f},$$

because f is continuous and bounded from below over X. It follows that

$$\lim_{k\to\infty}\|x^{k+1}-x^k\|^2 = 0.$$

Focusing on step 3 of the algorithm, by the limit above and since $\alpha_k = \overline{\alpha} = \frac{1}{\overline{L}+\overline{\rho}\|A\|^2}$ for all $k \ge \overline{k}$, this yields

$$\nabla_x \mathscr{L}_{\overline{\rho}}(\overline{x},\overline{\mu})^T(x-\overline{x}) \ge 0 \qquad \forall\, x \in X. \tag{9}$$

Let us suppose by contradiction that $\overline{f}$ is greater than the optimal value of Problem (1). Therefore a point $\widetilde{x} \in X$ exists such that $h(\widetilde{x}) = 0$ and $f(\widetilde{x}) < \overline{f} = f(\overline{x})$. We can write

$$0 \le \left(\nabla f(\overline{x}) + A^T\overline{\mu}\right)^T(\widetilde{x}-\overline{x}) = \nabla f(\overline{x})^T(\widetilde{x}-\overline{x}) \le f(\widetilde{x}) - f(\overline{x}) < 0,$$

where the first inequality comes from (9), the equality is a consequence of the fact that $h(\widetilde{x}) = 0$ and (7), the second inequality is due to the convexity of f and the last inequality comes from the hypotheses that $\overline{x}$ is not optimal and $\widetilde{x}$ is optimal. But this is impossible. Finally, $\overline{x}$ is a solution of Problem (1) and this completes the proof of the first part.

We now prove the second assertion, i.e. if $\overline{\rho} = \widehat{\rho}$ then $\overline{x}$ is an ε-approximate solution of Problem (1) with $\varepsilon = \min\left\{\frac{f^{max}-f^{min}}{\widehat{\mu}}, \sqrt{\frac{f^{max}-f^{min}}{\widehat{\rho}}}\right\}$. First we need to prove that the updates at steps 3 and 8 of Algorithm 2 are eventually equivalent to employ a gradient projection method to minimize the nonsmooth function $\phi_{\widehat{\mu},\widehat{\rho}}(x)$, defined in (5), over X. In Proposition 2 we proved that the subdifferential of ϕ satisfies for all $x \in X$

$$\nabla f(x) + \widehat{\mu}\sum_{i=1}^m \xi_i + \widehat{\rho}\nabla h(x)h(x) \in \partial\phi_{\widehat{\mu},\widehat{\rho}}(x),$$

with

$$\xi_i \in \begin{cases} \{A_{i*}^T\}, & \text{if } A_{i*}x - b_i > 0, \\ \{-A_{i*}^T\}, & \text{if } A_{i*}x - b_i < 0, \\ conv\{-A_{i*}^T, A_{i*}^T\}, & \text{if } A_{i*}x - b_i = 0. \end{cases}$$

The gradient of $\mathscr{L}_{\widehat{\rho}}(x^k, \mu^k)$ with respect to x is

$$\nabla \mathscr{L}_{\widehat{\rho}}(x^k, \mu^k) = \nabla f(x^k) + A^T \mu^k + \widehat{\rho} \nabla h(x) h(x),$$

where, recalling the update rule of μ^k at step 8 of Algorithm 2, $A^T \mu^k$ can be rewritten as

$$A^T \mu^k = \sum_{i=1}^m \mu_i^k A_{i*}^T = \widehat{\mu} \left(\sum_{i:h_i(x^k) \geq 0} A_{i*}^T + \sum_{i:h_i(x^k) < 0} -A_{i*}^T \right) = \widehat{\mu} \sum_{i=1}^m \psi_i,$$

where

$$\psi_i = \begin{cases} A_{i*}^T, & \text{if } A_{i*}x - b_i \geq 0, \\ -A_{i*}^T, & \text{if } A_{i*}x - b_i < 0, \end{cases}$$

which yields

$$\nabla_x \mathscr{L}_{\widehat{\rho}}(x^k, \mu^k) \in \partial_x \phi_{\widehat{\mu}, \widehat{\rho}}(x^k), \qquad \forall\, k \geq \widehat{k}.$$

We just proved that step 3 of Algorithm 2 is eventually equivalent to employ a gradient projection method to minimize the nonsmooth function $\phi_{\widehat{\mu}, \widehat{\rho}}(x)$, which is a convex and Lipschitz-continuous function by Proposition 1. About the stepsize, since $\widehat{k}$ is fixed and finite, we get

$$\sum_{k=0}^{\infty} \alpha_k = \sum_{k=0}^{\widehat{k}} \frac{1}{\overline{L} + \rho_k \|A\|^2} + \frac{1}{\gamma} \sum_{k=1}^{\infty} \frac{1}{k + \left(\frac{\overline{L} + \widehat{\rho} \|A\|^2}{\gamma} \right)} \geq \frac{1}{\gamma} \sum_{k = \left\lceil \frac{\overline{L} + \widehat{\rho} \|A\|^2}{\gamma} \right\rceil + 1}^{\infty} \frac{1}{k} = \infty,$$

and

$$\sum_{k=0}^{\infty} \alpha_k^2 = \sum_{k=0}^{\widehat{k}} \left(\frac{1}{\overline{L} + \rho_k \|A\|^2} \right)^2 + \frac{1}{\gamma^2} \sum_{k=1}^{\infty} \left(\frac{1}{k + \left(\frac{\overline{L} + \widehat{\rho} \|A\|^2}{\gamma} \right)} \right)^2 \leq \left(\widehat{k} + 1 \right) \left(\frac{1}{\overline{L} + \rho_0 \|A\|^2} \right)^2 + \frac{1}{\gamma^2} \sum_{k=1}^{\infty} \frac{1}{k^2} < \infty.$$

Therefore the rule in (4) is fulfilled. This implies that $\bar{x}$ is a minimizer for $\phi_{\widehat{\mu}, \widehat{\rho}}$ over X, see e.g. [3, Theorem 3.2.6]. The proof follows by Proposition 3. □

We remark the following facts:

(i) the updating rule for the stepsize α_k defined in step 2 of Algorithm 2 can be modified, as long as it satisfies (4) when $\rho_k = \widehat{\rho}$, namely it must be squared summable, but not summable;
(ii) as long as $\rho_k < \widehat{\rho}$, the update rule for the multipliers at step 5 does not play any role and therefore any bounded μ_k is acceptable; this can be useful in the first phase of Algorithm 2 where different updating rules for μ_k can be developed;
(iii) Algorithm 2, with $\widehat{\mu} = 0$, can be viewed as a modified version of a sequential penalty algorithm and therefore the above theoretical analysis can be directly applied also to such framework.

6 A distributed version of Algorithm 2

In this section we want to sketch a distributed version of Algorithm 2 similar to the one described in Section 2. Our aim in particular is to consider the case of a quadratic objective $f(x) = \frac{1}{2}x^T Qx + c^T x$, which will allow distribution of the data too. Namely, we want to be able to perform Algorithm 2 in a distributed fashion where each process, indexed by $\nu = 1, \ldots, N$,

- acts only on a block of variables $x_{(\nu)} \in \mathbb{R}^{n_\nu}$,
- stores only its own data, i.e. the columns $Q_{*(\nu)} \in \mathbb{R}^{n \times n_\nu}$ of the matrix $Q \in \mathbb{R}^{n\times n}$, $c_{(\nu)}$, X_ν and $A_{*(\nu)}$, and does not need to know the data blocks of the other processes,
- communicates with the other processes in order to get convergence to a (ε-approximate) solution of Problem (1).

In this framework, we recall that we can write the gradient of the objective function f as $\nabla f(x) = \left(\nabla f(x)_{(\nu)}\right)_{\nu=1}^N$. A possible distributed implementation is the following, where the generic process ν performs the following steps at any iteration k:

(i) compute $\alpha_k = \frac{1}{\overline{L}+\rho_k\|A\|^2+\gamma(k-\widehat{k})}$

(ii) compute $h(x^k) = \sum\limits_{\xi=1}^{N} h^{\xi,k} - b$ (where all the $h^{\xi,k} = A_{*(\xi)}\, x^k_{(\xi)}$ come from the communication phase at step (ix))

(iii) compute $\mu_i^k = \max\left\{-\widehat{\mu}, \min\left\{\widehat{\mu}, \mu_i^{k-1} + \frac{1}{\|A\|}h_i(x^k)\right\}\right\}$ if $\rho_{k-1} < \widehat{\rho}$, and $\mu_i^k = \begin{cases} -\widehat{\mu}, & i\,:\, h_i(x^k) < 0, \\ \widehat{\mu}, & otherwise, \end{cases}$ otherwise, for all $i = 1, \ldots, m$;

(iv) compute $\rho_k = \begin{cases} \min\{\rho_{k-1}+\delta,\ \widehat{\rho}\}, & \|h(x^k)\| > \tau\|h(x^{k-1})\|, \\ \rho_{k-1}, & otherwise; \end{cases}$

(v) compute $\nabla \mathscr{L}_{\rho_k}(x^k,\mu^k)_{(\nu)} = \sum\limits_{\xi=1}^{N} a^{\xi,k}_{(\nu)} + c_{(\nu)} + A^T_{*(\nu)}\mu^k + \rho_k A^T_{*(\nu)}h(x^k)$ (where all the $a^{\xi,k} = Q_{*(\xi)}\, x^k_{(\xi)}$ come from the communication phase at step (viii));

(vi) compute $x^{k+1}_{(\nu)} = \mathscr{P}_{X_\nu}\left[x^k_{(\nu)} - \alpha_k \nabla \mathscr{L}_{\rho_k}(x^k,\mu^k)_{(\nu)}\right]$;

(vii) compute $\widehat{k} = \begin{cases} k+1, & \rho_k < \widehat{\rho}, \\ \widehat{k}, & otherwise; \end{cases}$

(viii) compute $a^{\nu,k+1} = Q_{*(\nu)}\, x^{k+1}_{(\nu)}$ and broadcast;

(ix) compute $h^{\nu,k+1} = A_{*(\nu)}\, x^{k+1}_{(\nu)}$ and broadcast.

We observe that the above algorithm needs the communication of the vectors $a^{\nu,k+1}$ and $h^{\nu,k+1}$ at Steps (viii) and (ix). This communication is not too heavy from a computational performance point of view, although it can slow down the algorithm if the number of processes N is big. Beyond this communication phase, any process ν works only on its block of variables $x_{(\nu)}$ and needs only to know its data block $Q_{*(\nu)}$, $A_{*(\nu)}$, $c_{(\nu)}$, X_ν and b. As explained in Section 2, such possible distribution of the data can be beneficial, and sometimes is necessary, in many applications like e.g. support vector machines training. At step (i), any process ν performs the computation of the stepsize $\alpha_k > 0$ in a way that, as described in Section 3, it is squared summable but not summable for all k such that $\rho_k = \widehat{\rho}$. Furthermore, such update allows that $\alpha_k = 1/L^k_{\mathscr{L}}$ for all $k:\ \rho_k < \widehat{\rho}$ (see Section 3). At step (iii) each process ν updates the multipliers vector μ^k as in Algorithm 1 if $\rho_{k-1} < \widehat{\rho}$, or, otherwise, such that it is the maximum of the function $\mathscr{L}_{\rho_k}(x^k,\mu)$ over $M_{\widehat{\mu}}$ with respect to μ. At step (iv) the penalty parameter is increased if a sufficient descent in the violation of the coupling constraints defined by h is not achieved. Finally, at steps (v-vi) every process ν updates its block of variables $x_{(\nu)}$ by a gradient projection step over X_ν. We want to stress that such update does not require any approximate solution of the subproblem in X_ν, like instead is required in Algorithm 1.

We are ready to report some preliminary tests on the proposed distributed method. We consider the following problem setting: $n = 1000$, $m = 100$, Q positive definite and $\|Q\| = 1$ (i.e. $\overline{L} = 1$), c equal to the vector of all ones, $\|A\| = 1$, $b = 0$, $X = [-10,10]^n$. We focus on the following setting for Algorithm 2: $\gamma = 1$, $\delta = 0.5$, $\tau = 0.9$, $x^0 = 0$, $\mu^0 = 0$, $\rho_0 = 1$. We generated 4 experiments by considering different values for $\widehat{\mu}$ and $\widehat{\rho}$: exp1 with $\widehat{\mu}$ =1e-1 and $\widehat{\rho}$ =1e3, exp2 with $\widehat{\mu}$ =1e1 and $\widehat{\rho}$ =1e3, exp3 with $\widehat{\mu}$ =1e-1 and $\widehat{\rho}$ =1e4, exp4 with $\widehat{\mu}$ =1e1 and $\widehat{\rho}$ =1e4. In our tests we used: PC Windows with CPU Intel Core i7-8650U - 4 core (base frequency 1.9 GHz, turboboost up to 4.2 GHz) and RAM 16 GB, Python 3.6.3, and MPI 3.0.

In Figure 2, for all the 4 experiments, we report: with a full line the objective relative error (i.e. $\max\{$1e-4$,(f(x^k)-f^*)/f^*\}$, where f^* is the optimal value of the problem), and with a dotted line the coupling constraints violation (i.e. $\|h(x^k)\|/\sqrt{n}$), versus iterations (we report the first 120k iterations). In all the experiments the algorithm returns an ε-approximate solution of the problem, i.e. a point $\bar{x} \in X$ such that $f(\bar{x}) \leq f(x^*)$ and $\|h(\bar{x})\| \leq \varepsilon$. As theoretically observed in Theorem 1, Figure 2 confirms that the larger $\widehat{\mu}$ and $\widehat{\rho}$, the smaller ε. Moreover, our experiments make it evident that Algorithm 2 is faster in the first phase (when $\rho_k < \widehat{\rho}$) than in the second one. The slower convergence in exp1 and exp2 with respect to the one in exp3 and exp4 is due to the fact that we get $\rho_k = \widehat{\rho}$ quite early in exp1 and exp2 (around iteration 2k), while in exp3 and exp4 the algorithm switches to phase 2 later (around iteration 20k). For this reason it is important to set carefully the parameters $\widehat{\rho}$, δ, and τ to grant many iterations to phase 1 of the algorithm: while $\widehat{\rho}$ must be reasonably large, δ should be small, and τ close to 1. Another key issue is the value of $\widehat{\mu}$: large values of $\widehat{\mu}$ on the one hand yield small values of the approximation ε, as stated in Theorem 1, but on the other hand can make phase 2 of the algorithm really different from phase 1 thus spoiling the joint convergence of the objective relative error and the constraints violation that is obtained in phase 1. In our experiments we considered reasonable values for $\widehat{\mu}$ (up to 10), and we get good convergence behaviours. However in the figure of both exp3 and exp4 it is possible to see that, when the algorithm switches from phase 1 to phase 2 (around iteration 20k), the lines make a small jump.

We tested the algorithm with $N = 1$, 2, and 4 parallel processes, every process takes care of $1000/N$ variables. While the iterations clearly remain exactly the same, we observed a speedup in terms of CPU time that is almost ideal. Specifically, to execute 200k iterations we notice around 700 seconds with $N = 1$, around 360 seconds with $N = 2$, and around 190 seconds with $N = 4$, for all the experiments.

7 Conclusions and directions for future research

In this paper we propose an augmented Lagrangian method to solve convex problems with linear coupling constraints. The proposed method employs only one gradient projection step at any iteration and therefore both the data and the computation can be easily distributed. This can be beneficial (or necessary) in many practical situations, like e.g. when privacy concerns exist.

Convergence to (at least ε-approximate) solutions of the problem and a detailed analysis on the influence of the parameters to the effectiveness of the method are given. Furthermore, a parallel, distributed implementation is presented in the case where the function f is quadratic. Such case is of particular interest for many applications, e.g. support vector machines training.

In future work, we aim at testing such method on real applications like support vector machines training, in order to find out how to determine the parameters such that the quality of the solution and the convergence speed are acceptable. In such kind of problems the computation of the kernel matrix Q represents a large part of the computational burden [9]. Computing such matrix in a distributed fashion by the parallel processes may lead to substantial computational savings, especially if the available memory of each process is enough to allow pre-computation of the entire matrix offline.

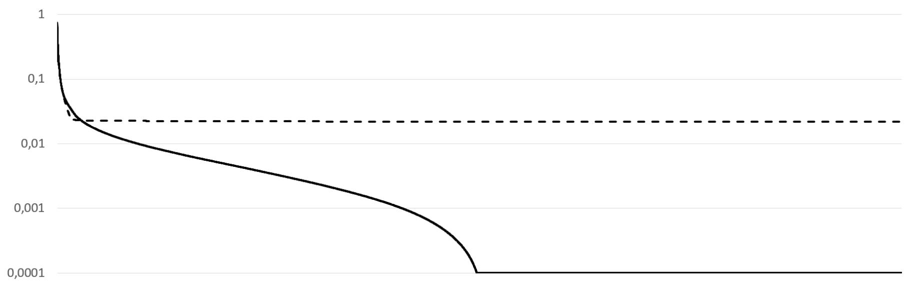
A The classical gradient projection algorithm defined in (2) is not distributable

Consider the general case in which S is not separable due to the presence of the constraints h that couple the different blocks of variables $x_{(\nu)}$, i.e. $m > 0$. To solve Problem (1), one could think to employ the following naive parallel version of the classical gradient projection algorithm (whose original generic iteration is defined in (2)):

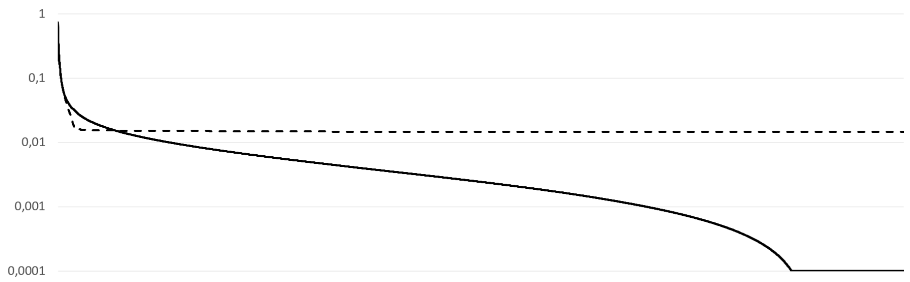
$$x_{(\nu)}^{k+1} = \mathscr{P}_{S_\nu(x_{(\nu)}^k)}\left[x_{(\nu)}^k - \alpha_k \nabla f(x^k)_{(\nu)}\right], \qquad \nu = 1,\ldots,N,$$

where $x^k \in S$ and the decomposed subsets S_ν are defined in the following way

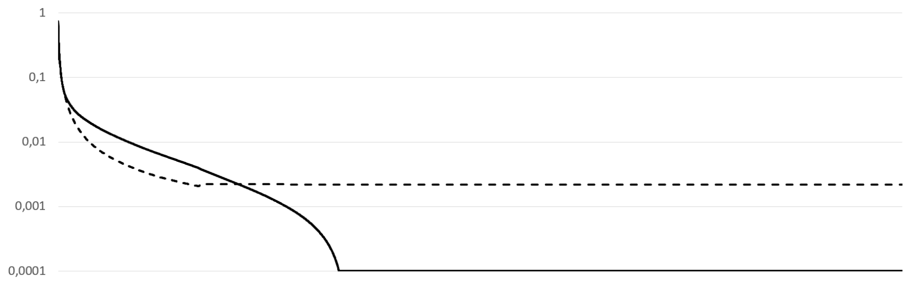
$$S_\nu(x_{(\nu)}^k) \triangleq \left\{x_{(\nu)} \in X_\nu : A_{*(\nu)} x_{(\nu)} = A_{*(\nu)} x_{(\nu)}^k\right\} \subseteq \Re^{n_\nu}, \qquad \nu = 1,\ldots,N.$$



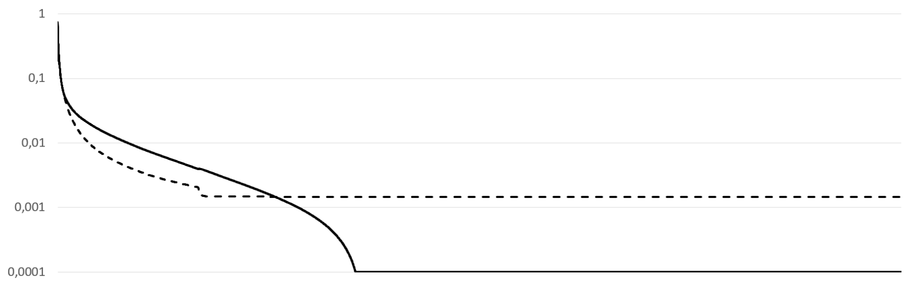
(a) exp1



(b) exp2



(c) exp3



(d) exp4

Fig. 2 Objective relative error (full line) and constraints violation (dotted line) Vs. iterations

Let $\{x^k\}$ be the sequence produced by this algorithm. The sets S_ν are fixed during the iterations and depend only on the starting point x^0:

$$S_\nu(x^{\nu,k}) = S_\nu(x^0_{(\nu)}) = S^0_\nu, \qquad \forall k \geq 0, \qquad \nu = 1,\ldots,N.$$

A fixed point $\bar{x}$ for $\{x^k\}$ is therefore a solution of the following variational inequality problem (see e.g. [1,15,16,25])

$$\bar{x} \in X,\ \ h(\bar{x}) = 0, \qquad \nabla f(\bar{x})^T(x-\bar{x}) \geq 0, \qquad \forall x \in \prod_{\nu=1}^{N} S^0_\nu. \tag{10}$$

On the other hand, computing a solution of Problem (1), being a fixed point for the iterations defined in (2), turns out to be a solution x^* of this different variational inequality

$$x^* \in X,\ \ h(x^*) = 0, \qquad \nabla f(x^*)^T(x-x^*) \geq 0, \qquad \forall x \in S. \tag{11}$$

Notice that the point $\bar{x}$, solution of the variational inequality (10), could not be a solution of the variational inequality (11), and therefore of Problem (1). This is due to the fact that, for any feasible starting guess $x^0 \in S$, we obtain only

$$\prod_{\nu=1}^{N} S_\nu^0 \subseteq S,$$

but not the other inclusion in general. Actually the fixed point $\bar{x}$ of $\{x^k\}$ is only an equilibrium of the (potential) generalized Nash equilibrium problem (see e.g. [14,17,36]) whose generic player $\nu \in \{1,\dots,N\}$ solves the following optimization problem that is parametric with respect to all the blocks of variables $x_{(\mu)}$ of the other players $\mu \neq \nu$:

$$\begin{aligned} \underset{x_{(\nu)}}{\text{minimize}} \quad & f(x) \\ \text{s.t.} \quad & h(x) = 0, \;\; x_{(\nu)} \in X_\nu. \end{aligned}$$

References

1. Aussel, D., Sagratella, S.: Sufficient conditions to compute any solution of a quasivariational inequality via a variational inequality. Mathematical Methods of Operations Research **85**(1), 3–18 (2017)
2. Bertsekas, D.P.: Nonlinear programming (1999)
3. Bertsekas, D.P., Scientific, A.: Convex optimization algorithms
4. Bertsekas, D.P., Tsitsiklis, J.N.: Parallel and distributed computation: numerical methods, vol. 23. Prentice hall Englewood Cliffs, NJ (1989)
5. Birgin, E.G., Martinez, J.M.: Practical augmented Lagrangian methods for constrained optimization, vol. 10. SIAM (2014)
6. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine learning **3**(1), 1–122 (2011)
7. Cannelli, L., Facchinei, F., Scutari, G.: Multi-agent asynchronous nonconvex large-scale optimization. In: 2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pp. 1–5. IEEE (2017)
8. Cassioli, A., Di Lorenzo, D., Sciandrone, M.: On the convergence of inexact block coordinate descent methods for constrained optimization. European Journal of Operational Research **231**(2), 274–281 (2013)
9. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM transactions on intelligent systems and technology (TIST) **2**(3), 27 (2011)
10. Clarke, F.H.: Optimization and nonsmooth analysis, vol. 5. Siam (1990)
11. Daneshmand, A., Sun, Y., Scutari, G., Facchinei, F., Sadler, B.M.: Decentralized dictionary learning over time-varying digraphs. arXiv preprint arXiv:1808.05933 (2018)
12. Di Pillo, G., Lucidi, S.: On exact augmented lagrangian functions in nonlinear programming. In: Nonlinear Optimization and Applications, pp. 85–100. Springer (1996)
13. Di Pillo, G., Lucidi, S.: An augmented lagrangian function with improved exactness properties. SIAM Journal on Optimization **12**(2), 376–406 (2002)
14. Facchinei, F., Kanzow, C.: Generalized Nash equilibrium problems. 4OR **5**(3), 173–210 (2007)
15. Facchinei, F., Kanzow, C., Karl, S., Sagratella, S.: The semismooth Newton method for the solution of quasi-variational inequalities. Computational Optimization and Applications **62**(1), 85–109 (2015)
16. Facchinei, F., Pang, J.S.: Finite-dimensional variational inequalities and complementarity problems. Springer Science & Business Media (2007)
17. Facchinei, F., Sagratella, S.: On the computation of all solutions of jointly convex generalized Nash equilibrium problems. Optimization Letters **5**(3), 531–547 (2011)
18. Facchinei, F., Scutari, G., Sagratella, S.: Parallel selective algorithms for nonconvex big data optimization. IEEE Transactions on Signal Processing **63**(7), 1874–1889
19. García, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, i: Convergence analysis. Optimization **52**(2), 171–200 (2003)
20. Gondzio, J., Grothey, A.: Exploiting structure in parallel implementation of interior point methods for optimization. Computational Management Science **6**(2), 135–160 (2009)
21. Harchaoui, Z., Juditsky, A., Nemirovski, A.: Conditional gradient algorithms for machine learning. In: NIPS Workshop on Optimization for ML, vol. 3, pp. 3–2 (2012)
22. Hong, M., Luo, Z.Q.: On the linear convergence of the alternating direction method of multipliers. Mathematical Programming **162**(1-2), 165–199 (2017)
23. Jaggi, M.: Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In: ICML (1), pp. 427–435 (2013)
24. Lacoste-Julien, S., Jaggi, M.: On the global linear convergence of Frank-Wolfe optimization variants. In: Advances in Neural Information Processing Systems, pp. 496–504 (2015)
25. Latorre, V., Sagratella, S.: A canonical duality approach for the solution of affine quasi-variational inequalities. Journal of Global Optimization **64**(3), 433–449 (2016)
26. Lin, C.J., Lucidi, S., Palagi, L., Risi, A., Sciandrone, M.: Decomposition algorithm model for singly linearly-constrained problems subject to lower and upper bounds. Journal of Optimization Theory and Applications **141**(1), 107–126 (2009)
27. Lucidi, S.: New results on a class of exact augmented lagrangians. Journal of Optimization Theory and Applications **58**(2), 259–282 (1988)
28. Lucidi, S., Palagi, L., Risi, A., Sciandrone, M.: A convergent decomposition algorithm for support vector machines. Computational Optimization and Applications **38**(2), 217–234 (2007)

29. Mangasarian, O.: Machine learning via polyhedral concave minimization. In: Applied Mathematics and Parallel Computing, pp. 175–188. Springer (1996)
30. Manno, A., Palagi, L., Sagratella, S.: Parallel decomposition methods for linearly constrained problems subject to simple bound with application to the svms training. Computational Optimization and Applications pp. 1–31
31. Manno, A., Sagratella, S., Livi, L.: A convergent and fully distributable svms training algorithm. In: Neural Networks (IJCNN), 2016 International Joint Conference on, pp. 3076–3080. IEEE (2016)
32. Ouyang, H., Gray, A.: Fast stochastic Frank-Wolfe algorithms for nonlinear SVMs. In: Proceedings of the 2010 SIAM International Conference on Data Mining, pp. 245–256. SIAM (2010)
33. Piccialli, V., Sciandrone, M.: Nonlinear optimization and support vector machines. 4OR **16**(2), 111–149 (2018). DOI 10.1007/s10288-018-0378-2
34. Rockafellar, R.T.: Augmented Lagrange multiplier functions and duality in nonconvex programming. SIAM Journal on Control **12**(2), 268–285 (1974)
35. Rockafellar, R.T., Wets, R.J.B.: Variational analysis, vol. 317. Springer Science & Business Media (2009)
36. Sagratella, S.: Algorithms for generalized potential games with mixed-integer variables. Computational Optimization and Applications **68**(3), 689–717 (2017)
37. Scutari, G., Facchinei, F., Lampariello, L.: Parallel and distributed methods for constrained nonconvex optimizationpart i: Theory. IEEE Transactions on Signal Processing **65**(8), 1929–1944
38. Scutari, G., Facchinei, F., Lampariello, L., Sardellitti, S., Song, P.: Parallel and distributed methods for constrained nonconvex optimization-part ii: applications in communications and machine learning. IEEE Transactions on Signal Processing **65**(8), 1945–1960
39. Scutari, G., Facchinei, F., Lampariello, L., Song, P.: Parallel and distributed methods for nonconvex optimization. In: Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pp. 840–844. IEEE (2014)
40. Woodsend, K., Gondzio, J.: Hybrid mpi/openmp parallel linear support vector machine training. Journal of Machine Learning Research **10**(Aug), 1937–1953 (2009)