A Distributed Strategy for Computing Proximity Operators


Abstract—Various recent iterative optimization methods require to compute the proximity operator of a sum of functions. We address this problem by proposing a new distributed algorithm for a sum of non-necessarily smooth convex functions composed with arbitrary linear operators. In our approach, each function is associated with a node of a graph, which communicates with its neighbors. Our algorithm relies on a primal-dual splitting strategy that avoids to invert any linear operator, thus making it suitable for processing high-dimensional datasets. The proposed algorithm has a wide array of applications in signal/image processing and machine learning and its convergence is established.

I. INTRODUCTION

Proximal splitting approaches for finding the minimizer of a sum of convex non-necessarily smooth functions have encountered growing popularity and attracted a large interest in the last years [1], [2]. In these approaches, the functions are processed either via their gradient or their proximity operator depending on their differentiability properties. However, when these functions are very complex, a closed form expression of the latter proximity operator does not exist, so that one has to resort to iterative strategies in order to compute it. Primal-dual splitting methods are of prominent use when dealing with convex optimization problems where numerous linear operators are involved [3], [4], [5], [6]. Indeed, solving both primal and dual problems leads to algorithms that avoid the inversion of any involved linear operator, so making this class of methods very well suited to large-scale problems encountered in various application fields [7], [8]. Primal-dual techniques are based on several well-known strategies such as the Forward-Backward iteration [9], [10], the Douglas-Rachford algorithm [11], [12], or the Alternating Direction Method of Multipliers [13], [14]. Recently, primal-dual algorithms have been combined with block-coordinate approach, where at each iteration only a few blocks are activated following a specific selection rule, and the associated variables are updated [15], [16], [17]. Stochastic and deterministic versions of these algorithms have been used in numerous fields such as image processing and machine learning, where they are often known as dual ascent methods [18], [19]. New interesting algorithms have been produced with a high flexibility in the rule of selection of the blocks and a faster convergence speed, combined with a reduced memory requirement.

All the aforementioned algorithms are designed to be implemented in a centralized manner which may be suboptimal when dealing with massive datasets. Therefore, various asynchronous or distributed extensions have been proposed [13], [20], [14], [21], where each term is handled by a processing unit and an aggregate solution of the optimization problem is reached thanks to communications between those processing units.

In our paper, we propose a new distributed algorithm for computing the proximity operator of the following sum of functions:

\[ (\forall x \in \mathbb{R}^N) \quad H(x) = \sum_{j=1}^{J} h_j(A_j x), \]

where, for every \( j \in \{1, \ldots, J\} \), \( h_j : \mathbb{R}^{M_j} \to [-\infty, +\infty] \) are convex possibly nonsmooth functions and \( A_j \) is a matrix in \( \mathbb{R}^{M_j \times N} \).

The proposed algorithm extends the dual block preconditioned forward-backward algorithm that was recently proposed in [22] to a distributed asynchronous version. Then, each function \( h_j \) is now considered as locally related to a node of a connected hypergraph where communications are allowed between neighbors nodes. Moreover, our method takes advantage of variable metric techniques that have been shown to be efficient for accelerating the convergence speed of proximal approaches [23], [24], [25]. Finally, it takes advantage of all the benefits of primal-dual splitting strategies (handling a finite sum of convex functions without inverting none of the involved linear operators) and it offers convergence guarantees.

The reminder of this paper is organized as follows: in Section II we recall some fundamental background and we introduce a centralized dual forward-backward algorithm for computing the desired proximity operator. Section III presents a distributed version of this algorithm, furthermore a parallel variant of the distributed algorithm is proposed. Finally, some conclusions are given.

II. PROBLEM FORMULATION

A. Optimization Tools

Let \( \Gamma_0(\mathbb{R}^N) \) designate the class of proper lower-semicontinuous convex functions from \( \mathbb{R}^N \) to \( [-\infty, +\infty] \) and let \( B \in \mathbb{R}^{N \times N} \) be a symmetric positive definite matrix. The proximity operator of \( \psi \in \Gamma_0(\mathbb{R}^N) \) at \( \bar{x} \in \mathbb{R}^N \) relative to the metric induced by \( B \) is denoted by \( \text{prox}_{B, \psi}(\bar{x}) \) and defined
as the unique solution to the following minimization problem [1]:

$$\min_{x \in \mathbb{R}^N} \psi(x) + \frac{1}{2} \|x - \tilde{x}\|^2_2,$$

(2)

where the weighted norm \(\| \cdot \|_B\) is defined by \(\langle \cdot | B \cdot \rangle^{1/2}\) with \(\langle \cdot | \cdot \rangle\) the usual scalar product of \(\mathbb{R}^N\). When \(B\) is set to the identity matrix, note that the standard proximity operator is recovered.

We also define the conjugate of a function \(\psi \in \Gamma_0(\mathbb{R}^N)\) as

$$\psi^*: \mathbb{R}^N \to [-\infty, +\infty]: x \mapsto \sup_{\nu \in \mathbb{R}^N} \langle \nu | x \rangle - \psi(\nu).$$

(3)

According to the Moreau decomposition theorem [26]:

$$\text{prox}_{B, \psi^*}(x) = \text{Id} - B^{-1} \text{prox}_{B^{-1}, \psi}(B \cdot).$$

(4)

**B. Minimization Problem**

In this paper, we are interested in computing the proximity operator of \(H\) defined in (1) at \(\tilde{x} \in \mathbb{R}^N\), which amounts to find the solution to the following minimization problem:

Find \(\tilde{x} = \text{prox}_H(\tilde{x})

= \arg \min_{x \in \mathbb{R}^N} \sum_{j=1}^J h_j(A_jx) + \frac{1}{2} \|x - \tilde{x}\|^2_2.

(5)

where \(\bigcap_{j \in \{1, \ldots, J\}} \text{dom}(h_j \circ A_j) \neq \emptyset\). A number of primal-dual algorithms can be applied to solve Problem (5) by resorting to its dual formulation given by:

Find \(\tilde{y} = \arg \min_{y \in \mathbb{R}^M} \sum_{j=1}^J h_j^*(A_j^Ty) + \frac{1}{2} \|\tilde{x} - \sum_{j=1}^J A_j^Ty\|^2_2 + \frac{1}{2} \sum_{j=1}^J h_j^*(y^j),

(6)

where \((h_j^*)_{1 \leq j \leq J}\) are the conjugate functions of \((h_j)_{1 \leq j \leq J}\).

Among existing efficient primal-dual approaches, the Dual Block Preconditioned Forward-Backward algorithm was recently proposed in [22]:

**Algorithm 1** Dual Block Preconditioned Forward-Backward

**Initialization:**

- \(B_j \in \mathbb{R}^{M_j \times M_j}\) with \(B_j \succeq A_jA_j^T\), \(\forall j \in \{1, \ldots, J\}\)
- \(\epsilon \in [0, 1]\)
- \((y_0^j)_{1 \leq j \leq J} \in \mathbb{R}^{M_j}\)
- \(x_0 = \tilde{x} - \sum_{j=1}^J A_j^Ty_0^j\)

**Main loop:**

For \(n = 0, 1, \ldots\)

- \(\gamma_n \in [\epsilon, 2 - \epsilon]\)
- \(\bar{y}_n^j = y_0^j + \gamma_n B_j^{-1}A_jx_n\)
- \(y_{n+1}^j = \bar{y}_n^j - \gamma_n B_j^{-1} \text{prox}_{\gamma_n B_j^{-1}h_j}(\gamma_n^{-1}B_j\bar{y}_n^j)\)
- \(x_{n+1} = x_n - A_j^{-1}(y_{n+1}^j - y_n^j)\)

Algorithm 1 benefits from the acceleration provided by variable metric methods through the introduction of preconditioning matrices \((B_j)_{1 \leq j \leq J}\). Note that a non-preconditioned version is obtained by setting \(\forall j \in \{1, \ldots, J\}\ B_j = A_j^{-1}I_{M_j}\). Moreover, when all the dual variables \(y_n^j\) are updated in a parallel way followed by an update of the primal variable \(x_n\), one recovers the Parallel Dual Forward-Backward proposed in [27].

Results in terms of convergence speed reveal the effectiveness of the above algorithm compared to existing algorithms in the literature and convergence guarantees on both generated primal and dual sequences are supplied.

**III. DISTRIBUTED ALGORITHM**

In order to generate a distributed or decentralized solution to Problem (5), we will make use of a global consensus technique by rewriting the problem under the following form:

Find \(\tilde{x} = \underset{x = (x^j)_{1 \leq j \leq J} \in \Lambda_J}{\text{arg min}} \sum_{j=1}^J h_j(A_jx^j) + \frac{1}{2} \sum_{j=1}^J \omega_j \|x^j - \tilde{x}\|^2_2,$

(7)

where \((\omega_j)_{1 \leq j \leq J} \in [0, 1]^J\) are such that \(\sum_{j=1}^J \omega_j = 1\), and \(\Lambda_J\) is the vector subspace of \(\mathbb{R}^{NJ}\) that enforces the convergence toward the aggregate solution, defined as

\[ \Lambda_J = \{(x^1)^T, \ldots, (x^J)^T \in \mathbb{R}^{NJ} | x^1 = \ldots = x^J\}. \]

(8)

One can notice that the solution to Problem (7) is linked to the solution to Problem (5) when the variables \((x^j)_{1 \leq j \leq J}\) are all equal to \(\tilde{x}\), that is the value of the proximity operator of \(\sum_{j=1}^J h_j \circ A_j\) at \(\tilde{x}\) (see (5)).

**A. Local Form of Consensus**

For more flexibility, let us split the constraint set \(\Lambda_J\) into \(L\) local constraints \(\Lambda_{\kappa_\ell}\) with cardinalities \((\kappa_\ell)_{1 \leq \ell \leq L}\). Each constraint set \(\Lambda_{\kappa_\ell}\) handles a subset \((\mathcal{V}_\ell)_{1 \leq \ell \leq L}\) of \(\{1, \ldots, J\}\) such that, for every \(x = [(x^1)^T, \ldots, (x^J)^T]^T \in \mathbb{R}^{NJ}\),

\[ x \in \Lambda_J \iff (\forall \ell \in \{1, \ldots, L\}) (x^j)_{j \in \mathcal{V}_\ell} \in \Lambda_{\kappa_\ell}. \]

(9)

Fig. 1 represents an example of a connected hypergraph induced by the sets \((\mathcal{V}_\ell)_{1 \leq \ell \leq L}\). This hypergraph is composed of \(J = 7\) nodes associated to the functions \((h_j)_{1 \leq j \leq 7}\) and \(L = 4\) hyperedges represented by the sets \((\mathcal{V}_\ell)_{1 \leq \ell \leq 4}\) with cardinalities \(\kappa_1 = 3, \kappa_2 = 2, \kappa_3 = 2\) and \(\kappa_4 = 3\) respectively. Each function \(h_j\) is considered as local and processes its own private data. Moreover, each node \(j\) is allowed to communicate with nodes that belong to the same set \(\mathcal{V}_\ell\).

For instance, node 4 belongs to the set \(\mathcal{V}_4\) and communicates with node 5. Besides, node 3 belongs to \(\mathcal{V}_1\) and \(\mathcal{V}_4\) hence it is allowed to communicate with the nodes \(\{1, 2, 5, 7\}\). Note that the connectivity of the hypergraph is essential in order to ensure the convergence toward the consistent solution \(\tilde{x}\).

Let us define, for every \(\ell \in \{1, \ldots, L\}\), the matrix \(S_\ell \in \mathbb{R}^{N_{\kappa_\ell} \times NJ}\) associated to a constraint \(\Lambda_{\kappa_\ell}\) that extracts the vector \((x^j)_{j \in \mathcal{V}_\ell} \in \Lambda_{\kappa_\ell}\) from the concatenated vector \(x = [(x^1)^T, \ldots, (x^J)^T]^T \in \mathbb{R}^{NJ}\):

\[ (x^j)_{j \in \mathcal{V}_\ell} = [(x^{i(\ell, 1)})^T, \ldots, (x^{i(\ell, \kappa_\ell)})^T]^T = S_\ell x, \]

(10)
where \( i(\ell,1), \ldots, i(\ell,\kappa_{\ell}) \) denote the elements of \( \mathbb{V}_{\ell} \) ordered in an increasing manner. The transpose of matrices \( (S_{\ell})_{1 \leq \ell \leq L} \) is such that for every \( v^T = (v_{\ell,k})_{1 \leq \ell \leq L, k \in \kappa_{\ell}} \in \mathbb{R}^{N_{\kappa_{\ell}}}, \)
\[
x = [(x^1)^T, \ldots, (x^L)^T] = S_{\ell}^T v^T,
\]
where
\[
x^j = \begin{cases} v_{\ell,k} & \text{if } j = i(\ell,k) \text{ with } k \in \{1, \ldots, \kappa_{\ell}\} \\ 0 & \text{otherwise.} \end{cases}
\]

The above definitions allow us to propose the following alternative formulation of Problem (7):
\[
\text{Find } \hat{x} = \arg\min_{x=(x_j)_{1 \leq j \leq J} \in \mathbb{R}^{NJ}} \sum_{j=1}^{J} h_j(A_j x_j) + \sum_{\ell=1}^{L} \ell \lambda_{\ell} \ell(x_{\ell}) x_{\ell} + \frac{1}{2} \sum_{j=1}^{J} \omega_j \|x_j - \tilde{x}\|^2. \quad (13)
\]

The main difference between formulations (7) and (13) is the introduction of the term \( \sum_{\ell=1}^{L} \ell \lambda_{\ell} \ell(x_{\ell}) x_{\ell} \) that allows updates in an asynchronous fashion and ensures the convergence to the aggregate solution.

In order to solve Problem (13) using Algorithm 1, it is necessary to set the following parameters:
- \( J' = J + L \),
- \( (\forall \ell \in \{1, \ldots, L\}) \quad M_{J+\ell} = N \kappa_{\ell}, \)
- \( M = \sum_{\ell=1}^{J} M_{\ell}, \)
- \( (\forall j \in \{1, \ldots, J\}) \quad A_j = [0_{1 \times \omega_{j}^{-1/2}} A_j 0_{1 \times \omega_{j}^{-1/2}}], \)
- \( D = \text{Diag}(\omega_{j}^{-1/2} I_{N}, \ldots, \omega_{j}^{-1/2} I_{N}), \)
- \( (\forall j \in \{1, \ldots, J\}) \quad h_j(A_j x_j) = \ell(x_{\ell}) \quad \text{and} \quad A_{J+\ell} = S_{\ell} D. \)

Then, Problem (13) is recast as:

\[
\text{Find } \hat{x} = D \hat{x}' \quad \text{such that } \quad 
\hat{x}' = \arg\min_{x' \in \mathbb{R}^{N_{J}}} \sum_{j=1}^{J} h_j(A_j x') + \frac{1}{2} ||x' - \tilde{x}||^2, \quad (14)
\]

where \( \tilde{x} = [x_{\ell}^{1/2} x^T, \ldots, x_{\ell}^{-1/2} x_{\ell}^T]^T \in \mathbb{R}^{N_{J}}. \)

\section{Proposed Algorithm}

Using Algorithm 1 to solve Problem (14) leads to the following distributed algorithm after some reindexing and simplifications:

\begin{algorithm}
\caption{Distributed Preconditioned Dual Forward-Backward}
\begin{algorithmic}
\State \textbf{Initialization:} \quad (\omega_j)_{1 \leq j \leq J} \in [0, 1]^J \text{ such that } \sum_{j=1}^{J} \omega_j = 1 \quad \text{and } B_j \in \mathbb{R}^{M_j \times M_j} \text{ with } B_j \succeq A_j A_j^T, \quad j \in \{1, \ldots, J\} \quad \text{and } \rho \in \mathbb{R} \text{, } A \in \{0, 1\}
\State \text{Main loop:} \quad \text{For } n = 0, 1, \ldots
\State \text{If } j_n \not\in J \quad \text{else}
\State \text{Local optimization:} \quad y_{n+1} = y_n + \gamma_n B_j^{-1} \text{prox}_{\gamma_n \omega_j} (y_n - B_j^{-1} h_n (\gamma_n^{-1} B_j^{-1} \tilde{y}_n)) \quad j \in \{1, \ldots, J\} \quad \text{and } y_{n+1} = y_n, \quad j \in \{1, \ldots, J\} \quad \text{and } j \notin \{1, \ldots, J\} \quad \text{else}
\State \text{Synchronization:} \quad \ell_n = j_n - J \quad \text{and } j_{n+1} = y_{n+1}, \quad j \in \{1, \ldots, J\}
\State \ell_n = j_n - J \quad \text{and } j_{n+1} = y_{n+1}, \quad j \in \{1, \ldots, J\}
\State \text{For } k = 1, \ldots, \kappa_{\ell_n} \quad \text{and } j \notin \{1, \ldots, J\}
\State \quad \ell_{n+1}^{\ell_n, k} = \ell_n^{\ell_n, k}, \quad \text{and } j \notin \{1, \ldots, J\}
\end{algorithmic}
\end{algorithm}

Fig. 1. Connected hypergraph of 7 nodes and 4 hyperedges.
One can notice that Algorithm 2 is composed of two main parts:

- First a local optimization part which is reminiscent of the Dual Block Forward-Backward algorithm where, at each iteration, a block \( j_n \) is selected and the associated dual and primal variables \( y_n^j \) and \( x_n^j \) respectively are updated. Note that the difference between the proposed algorithm and Algorithm 1 lies in the fact that each block \( j_n \) is now associated to a primal variable \( x_n^j \) whereas, in Algorithm 1 \( x_n \) was a shared variable.

- The second part of Algorithm 2 is a synchronization step in which a set \( V_{\ell_n} \) is selected and all the variables \( (x^{\ell_n})_{j_n \in V_{\ell_n}} \) are updated by computing the average over the selected set \( V_{\ell_n} \).

In Algorithm 2 all computation steps only involve local variables, so considerable flexibility is allowed by the quasicyclic rule for choosing the indices \( j_n \) and \( \ell_n \) at each iteration \( n \). This distributed algorithm inherits all the advantages of primal-dual methods, in particular it requires no inversion of the matrices \((A_j)_{1 \leq j \leq J}\), which is of main interest when these matrices do not have a simple structure and are of very large size. Note that the proposed approach is quite different from the ones developed in [20] since it does not implement a random sweeping rule and the convergence results do not rely on nonexpansiveness properties of some stochastic operators.

Remark that the case of a graph topology is encompassed by Algorithm 2, by setting for every \( \ell \in \{1, \ldots, L\} \) the cardinality of the sets \( V_{\ell} \) to \( \kappa_{\ell} = 2 \), and requiring after each optimization step on block \( j_n \), a synchronization step on all the sets \( V_{\ell} \) to which \( j_n \) belongs.

C. Special Case when \( L = 1 \)

An interesting instance of Algorithm 2 is obtained by setting \( L = 1 \), \( V_1 = \{1, \ldots, J\} \), and by performing at each iteration an update of a subset of the dual variables followed by a global averaging step. After some reindexing of the involved variables, this yields Algorithm 3.

This algorithm allows to solve Problem (7) by computing the dual variables in a parallel manner. At iteration \( n \in \mathbb{N} \), it allows to activate only a subset \( (y_n^j)_{j \in J_n} \) of them. It should be emphasized that even in the case when all the dual variables are updated iteratively (i.e. \( (\forall n \in \mathbb{N}) \ J_n = \{1, \ldots, J\} \) ), Algorithm 3 exhibits a different structure from the Parallel Dual Forward-Backward Algorithm in [27].

### Algorithm 3 Parallel Preconditioned Dual Forward-Backward

**Initialization:**

\[
(\omega_j)_{1 \leq j \leq J} \in [0, 1]^J \text{ such that } \sum_{j=1}^{J} \omega_j = 1
\]

\[
B_j \in \mathbb{R}^{M_j \times M_j} \text{ with } B_j \succeq A_j A_j^\top, \quad j \in \{1, \ldots, J\}
\]

\[
\vartheta \in [0, 1]
\]

\[
y_0^j \in \mathbb{R}^{M_j}, \quad x_0^j = \tilde{x} - A_j^\top y_0^j, \quad j \in \{1, \ldots, J\}.
\]

**Main loop:**

For \( n = 0, 1, \ldots \)

\[
\gamma_n \in [\vartheta, 2 - \vartheta]
\]

\[
J_n \subset \{1, \ldots, J\}
\]

For \( j \in J_n \)

\[
\gamma_n^j = y_n^j + \gamma_n B_j^{-1} A_j x_n^j
\]

\[
y_{n+1}^j = \gamma_n^j - \gamma_n B_j^{-1} \text{prox}_{\gamma_n \omega_j B_j^{-1} A_j^\top} (\gamma_n^{-1} B_j \gamma_n^j)
\]

\[
x_{n+1/2}^j = x_n^j - A_j^\top (y_{n+1}^j - y_n^j)
\]

For \( j \in \{1, \ldots, J\} \setminus J_n \)

\[
y_{n+1}^j = y_n^j
\]

\[
x_{n+1/2}^j = x_n^j
\]

\[
\pi_n = \frac{1}{J} \sum_{j=1}^{J} x_{n+1/2}^j
\]

For \( j = 1, \ldots, J \)

\[
x_{n+1}^j = x_{n+1/2}^j + \gamma_n \vartheta \omega_j^{-1}(\pi_n - x_{n+1/2}^j).
\]

### IV. Conclusion

We have proposed a new asynchronous primal-dual algorithm for computing the proximity operator of a sum of convex functions composed with arbitrary linear operators, which is a frequently encountered problem in various application fields. The proposed algorithm benefits from the flexibility offered by variable metric techniques that can significantly improve its convergence speed. The convergence properties of this algorithms will be discussed in a forthcoming paper.

As future work, we also intend to apply the proposed algorithm to large-scale inverse problems, especially those encountered in video restoration [28].
REFERENCES


