Distributed Submodular Minimization via Block-Wise Updates and Communications

Andrea Testa, Francesco Farina, Giuseppe Notarstefano

Department of Electrical, Electronic, and Information Engineering,
 Alma Mater Studiorum - Università di Bologna, Bologna, Italy
{a.testa, franc.farina, giuseppe.notarstefano}@unibo.it

Abstract: In this paper we deal with a network of computing agents with local processing and neighboring communication capabilities that aim at solving (without any central unit) a submodular optimization problem. The cost function is the sum of many local submodular functions and each agent in the network has access to one function in the sum only. In this distributed set-up, in order to preserve their own privacy, agents communicate with neighbors but do not share their local cost functions. We propose a distributed algorithm in which agents resort to the Lovász extension of their local submodular functions and perform local updates and communications in terms of single blocks of the entire optimization variable. Updates are performed by means of a greedy algorithm which is run only until the selected block is computed, thus resulting in a reduced computational burden. The proposed algorithm is shown to converge in expected value to the optimal cost of the problem, and an approximate solution to the submodular problem is retrieved by a thresholding operation. As an application, we consider a distributed image segmentation problem in which each agent has access only to a portion of the entire image. While agents cannot segment the entire image on their own, they correctly complete the task by cooperating through the proposed distributed algorithm.

Keywords: Distributed optimization, Learning algorithms, Submodular minimization

1. INTRODUCTION

Many combinatorial problems in machine learning can be cast as the minimization of submodular functions (i.e., set functions that exhibit a diminishing marginal returns property). Applications include isotonic regression, image segmentation and reconstruction, and semi-supervised clustering (see, e.g., Bach et al. (2013)).

In this paper we consider the problem of minimizing in a distributed fashion (without any central unit) the sum of \( N \in \mathbb{N} \) submodular functions, i.e.,

\[
\min_{X \subseteq V} F(X) = \sum_{i=1}^{N} F_i(X)
\]

where \( V = \{1, \ldots, n\} \) is called the ground set and the functions \( F_i \) are submodular.

We consider a scenario in which problem (1) is to be solved by \( N \) peer agents communicating locally and performing local computations. The communication is modeled as a directed graph \( G = (V, E) \), where \( V = \{1, \ldots, N\} \) is the set of agents and \( E \subseteq V \times V \) is the set of directed edges in the graph. Each agent \( i \) receives information only from its in-neighbors, i.e., agents \( j \in N_i^{\text{in}} \triangleq \{ j \mid (j, i) \in E \} \cup \{i\} \), while it sends messages only to its out-neighbors \( j \in N_i^{\text{out}} \triangleq \{ j \mid (i, j) \in E \} \cup \{i\} \), where we have included

* This result is part of a project that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).
Submodular minimization problems can be mainly addressed in two ways. On the one hand, a number of combinatorial algorithms have been proposed (Iwata et al. (2001); Iwata and Orlin (2009)), some based on graph-cut algorithms Jegelka et al. (2011) or relying on problems with a particular structure Kolmogorov (2012). On the other hand, convex optimization techniques can be exploited to face submodular minimization problems by resorting to the so called Lovász extension. Many specialized algorithms have been developed in the last years by building on the particular properties of submodular functions (see Bach et al. (2013) and reference therein). In this paper we focus on the problem of minimizing the sum of many submodular functions, which has received attention in many works Stobbe and Krause (2010); Kolmogorov (2012); Jegelka et al. (2013); Fix et al. (2013); Nishihara et al. (2014). In particular, centralized algorithms have been proposed based on smoothed convex minimization Stobbe and Krause (2010) or alternating projections and splitting methods Jegelka et al. (2013), whose convergence rate is studied in Nishihara et al. (2014). This problem structure typically arises, for example, in Markov Random Fields (MRF) Maximum a-Posteriori (MAP) problems Shamu et al. (2016); Fix et al. (2013), a notable example of which is image segmentation.

While a vast literature on distributed continuous optimization has been developed in the last years (see, e.g., Notarstefano et al. (2019)), distributed approaches for tackling (submodular) combinatorial optimization problems started to appear only recently. Submodular maximization problems have been treated and approximately solved in a distributed way in several works Kim et al. (2011); Mirzasoleiman et al. (2013); Bogunovic et al. (2017); Williams et al. (2017); Gharesifard and Smith (2017); Grimsman et al. (2017). In particular, distributed submodular maximization subject to matroid constraints is addressed in Williams et al. (2017); Gharesifard and Smith (2017), while in Grimsman et al. (2017), the authors handle the design of communication structures maximizing the worst case efficiency of the well-known greedy algorithm for submodular maximization when applied over networks. Regarding distributed algorithms for submodular minimization problems, they have not received much attention yet. In Jalal et al. (2018) a distributed subgradient method is proposed, while in Testa et al. (2018) a greedy column generation algorithm is given. All these approaches involve the communication/update of the entire decision variable at each time instant. This can be an issue when the decision variable is extremely large. Thus, block-wise approaches like those proposed in Notarincola et al. (2018) should be explored.

**Contribution and organization** The main contribution of this paper is the MiXing bloCKs and grEedy (MICKY) method, i.e., a distributed block-wise algorithm for solving problem (1). At any iteration, each agent computes a weighted average on local copies of neighbors solution estimates. Then, it selects a random block and performs an ad-hoc (block-wise) greedy algorithm (based on the one in (Bach et al., 2013, Section 3.2)) until the selected block is updated. Finally, based on the output of the greedy algorithm, the selected block of the local solution estimate is updated and broadcast to the out-neighbors. The proposed algorithm is shown to produce cost-optimal solutions in expected value by showing that it is an instance of the Distributed Block Proximal Method presented in Farina and Notarstefano (2019). In fact, the partial greedy algorithm performed on the local submodular cost function \( F_i \) is shown to compute a block of a subgradient of its Lovász extension.

A key property of this algorithm is that each agent is required to update and transmit only one block of its solution estimate. In fact, it is quite common for networks to have communication bandwidth restrictions. In these cases the entire state variable may not fit the communication channels and, thus, standard distributed optimization algorithms cannot be applied. Furthermore, the greedy algorithm can be very time consuming when an oracle for evaluating the submodular functions is not available and, hence, halting it earlier can reduce the computational load.

The paper is organized as follows. The distributed algorithm is presented and analyzed in Section 2, and it is tested on a distributed image segmentation problem in Section 3.

**Notation and definitions** Given a vector \( x \in \mathbb{R}^n \), we denote by \( x_\ell \) the \( \ell \)-th entry of \( x \). Let \( V \) be a finite, non-empty set with cardinality \( |V| \). We denote by \( 2^V \) the set of all its \( 2^{|V|} \) subsets. Given a set \( X \subseteq V \), we denote by \( 1_X \in \mathbb{R}^{|V|} \) its indicator vector, defined as \( 1_X = 1 \) if \( \ell \in X \), and 0 if \( \ell \notin X \). A set function \( F : 2^V \rightarrow \mathbb{R} \) is said to be submodular if it exhibits the diminishing marginal returns property, i.e., for all \( A, B \subseteq V \), \( A \subseteq B \) and for all \( j \in V \setminus B \), it holds that \( F(A \cup \{j\}) - F(A) \geq F(B \cup \{j\}) - F(B) \). In the following we assume \( F(X) < \infty \) for all \( X \subseteq V \) and, without loss of generality, \( F(\emptyset) = 0 \). Given a submodular function \( F : 2^V \rightarrow \mathbb{R} \), we define the associated base polyhedron as \( \mathcal{B}(F) := \{ w \in \mathbb{R}^n \mid \sum_{\ell \in X} w_\ell \leq F(X) \forall X \in 2^V, \sum_{j \in V} w_j = F(V) \} \) and by \( f(x) = \max_{w \in \mathcal{B}(F)} w^\top x \) the Lovász extension of \( F \).

2. DISTRIBUTED ALGORITHM

2.1 Algorithm description

In order to describe the proposed algorithm, let us introduce the following nonsmooth convex optimization problem

\[
\min_{x \in [0,1]^n} f(x) = \sum_{i=1}^N f_i(x)
\]
where \( f_i(x) : \mathbb{R}^n \rightarrow \mathbb{R} \) is the Lovász extension of \( F_i \) for all \( i \in \{1, \ldots, N\} \). It can be shown that solving problem (2) is equivalent to solving problem (1) (see, e.g., Lovász (1983) and (Bach et al., 2013, Proposition 3.7)). In fact, given a solution \( x^* \) to problem (1), a solution \( X^* \) to problem (1) can be retrieved by thresholding the components of \( x^* \) at an arbitrary \( \tau \in [0, 1] \) (see (Bach, 2019, Theorem 4), i.e.,
\[
X^* = \{ \ell \mid x^*_\ell > \tau \}.
\]
Notice that, given \( F_i \) in problem (1), each agent \( i \) in the network is able to compute \( f_i \); thus, in the considered distributed set-up, problem (2) can be addressed in place of problem (1). Moreover, since \( F_i \) is submodular for all \( i \), then \( f_i \) is a continuous, piece-wise affine, non-smooth convex function, see, e.g., Bach et al. (2013).

In order to compute a single block of a subgradient of \( f_i \), each agent \( i \) is equipped with a local routine (reported next), that we call BLOCKGREEDY and that resembles a local (block-wise) version of the greedy algorithm in (Bach et al., 2013, Section 3.2). This routine takes as inputs a vector \( y \) and the required block \( \ell \), and returns the \( \ell \)-th block of a subgradient \( g_i \) of \( f_i \) at \( y \). For the sake of simplicity, suppose \( \ell \) is a single component block. Moreover, assume to have a routine PARTIALSORT that generates an ordering \( \{m_1, \ldots, m_p\} \) such that \( y_{m_1} \geq \cdots \geq y_{m_p} \), \( m_p = \ell \) and \( y_p \leq y_\ell \) for each \( r \in \{1, \ldots, n\} \setminus \{m_1, \ldots, m_p\} \). Then, the BLOCKGREEDY algorithm reads as follows.

\[
\textbf{Routine BLOCKGREEDY}(y, \ell) \quad \text{for agent } i
\]
\[\text{Input: } y, \ell \]
\[
\text{Obtain a partial order via } \\
\{m_1, \ldots, m_{p-\ell}, m_p = \ell\} = \text{PARTIALSORT}(y)
\]
\[\text{Evaluate } g_{i,m_p}, \text{ as } \\
g_{i,\ell} = \begin{cases} 
F_i(\{\ell\}), & \text{if } p = 1 \\
F_i(\{m_1, \ldots, m_{p-\ell}, \ell\}) - F_i(\{m_1, \ldots, m_{p-1}\}), & \text{otherwise}
\end{cases}
\]
\[\text{Output: } g_{i,\ell}
\]

The MICKY algorithm works as follows. Each agent stores a local solution estimate \( x^i_\ell \) of problem (2) and, for each in-neighbor \( j \in \mathcal{N}_i^{\text{in}} \), a local copy of the corresponding solution estimate \( x^j_\ell \). At the beginning, each node selects the initial condition \( x^i_0 \) at random in \([0, 1]^n\) and shares it with its out-neighbors. We associate to the communication graph \( G \) a weighted adjacency matrix \( W \in \mathbb{R}^{N \times N} \) and we denote with \( w_{ij} = |W|_{ij} \) the weight associated to the edge \((i, j)\). At each iteration \( k \), agent \( i \) performs three tasks:

(i) it computes a weighted average \( y^k_i = \sum_{j \in \mathcal{N}_i^{\text{in}}} w_{ij} x^j_\ell \),
(ii) it picks randomly (with arbitrary probabilities bounded away from 0) a block \( \ell^k \in \{1, \ldots, n\} \) and performs the BLOCKGREEDY \( (y^k_i, \ell^k) \);
(iii) it updates \( x^i_{\ell^k} \) according to (7), where \( \Pi_{[0,1]} \) is the projector on the set \([0, 1]\), and \( \alpha^k \in (0, 1) \), and broadcasts it to its out-neighbors \( j \in \mathcal{N}_i^{\text{out}} \).

Agents halt the algorithm after \( K > 0 \) iterations and recover the local estimates \( X^i_{\ell^k} \) of the set solution to problem (1) by thresholding the value of \( x^i_\ell \) as in (3). Notice that, in order to avoid to introduce additional notation, we have assumed each block of the optimization variable to be scalar (so that blocks are selected in \( \{1, \ldots, n\} \)). However, blocks of arbitrary sizes can be used (as shown in the subsequent analysis). A pseudocode of the proposed algorithm is reported in the next table.

### Algorithm MICKY (Mixing Blocks and Greedy Method)

#### Initialization:
\[
x^0_i
\]
\[\textbf{for } k = 1, \ldots, K - 1 \text{ do } \\
\textbf{Update for all } j \in \mathcal{N}_i^{\text{in}}, \text{ } \ell \in \{1, \ldots, n\} \text{ do } \\
\text{Compute } \\
y^k_i = \sum_{j \in \mathcal{N}_i^{\text{in}}} w_{ij} x^j_\ell
\]
\[\textbf{Pick randomly a block } \ell^k \in \{1, \ldots, n\} \text{ do } \\
\textbf{Compute } \\
y^k_i = \text{BLOCKGREEDY}(y^k_i, \ell^k)
\]
\[\textbf{Update } \\
x^{k+1}_\ell = \begin{cases} 
\Pi_{[0,1]} \left[ x^k_\ell - \alpha^k y^k_i x^j_\ell \right] & \text{if } \ell = \ell^k \\
x^k_\ell & \text{otherwise}
\end{cases}
\]
\[\text{Broadcast } x^{k+1}_\ell \text{ to all } j \in \mathcal{N}_i^{\text{out}} \text{ do } \\
\text{Thresholding } \\
X^i_{\ell^k} = \{ \ell \mid x^{K}_\ell > \tau \}
\]

#### 2.2 Discussion

The proposed algorithm possesses many interesting features. Its distributed nature requires agents to communicate only with their direct neighbors, without resorting to multihop communications. Moreover, all the local computations involve locally defined quantities only. In fact, stepsize sequences and block drawing probabilities are locally defined at each node.

Regarding the block-wise updates and communications, they bring benefits in two areas. Communicating single blocks of the optimization variable, instead of the entire one, can significantly reduce the communication bandwidth required by each agent in broadcasting their local estimates. This makes the proposed algorithm implementable in networks with communication bandwidth restrictions. Moreover, the classical greedy algorithm requires to evaluate \( |V| \) times the submodular function in order to produce a subgradient. When \( |V| \) is very high and an oracle for evaluating functions \( F_i \) is not available, this can be a very time consuming task. For example, in the example application in Section 3, we will resort to the minimum graph cut problem. Evaluating the value of a cut for a graph in which \( E \subseteq V \times V \) is the set of arcs, requires a running-time \( O(|E|) \). In the BLOCKGREEDY routine, in contrast with what happens in the standard greedy routine, the sorting operation is (possibly) performed only on a part of the entire vector \( y \), i.e., until the \( \ell \)-th component has been sorted. Thus, our routine evaluates the \( \ell \)-th component of the subgradient in at most two evaluations of the submodular function.
2.3 Analysis

In order to state the convergence properties of the proposed algorithm, let us make the following two assumptions on the communication graph and the associated weighted adjacency matrix $W$.

**Assumption 1.** (Strongly connected graph). The digraph $G = (V, E, W)$ is strongly connected. □

**Assumption 2.** (Doubly stochastic weight matrix). For all $i, j \in V$, the weights $w_{ij}$ of the weight matrix $W$ satisfy

(i) if $i \neq j$, $w_{ij} > 0$ if and only if $j \in N_i^m$;
(ii) there exists a constant $\eta > 0$ such that $w_{ii} \geq \eta$ and if $w_{ij} > 0$, then $w_{ij} \geq \eta$;
(iii) $\sum_{j=1}^{N} w_{ij} = 1$ and $\sum_{i=1}^{N} w_{ij} = 1$. □

The above two assumptions are very common when designing distributed optimization algorithms. In particular, Assumption 1 guarantees that the information is spread through the entire network, while Assumption 2 assures that each agent gives sufficient weight to the information coming from its in-neighbors.

Let $\bar{x}_k \triangleq \frac{1}{K} \sum_{i=1}^{K} x_i^k$ be the average over the agents of the local solution estimates at iteration $k$ and define $f_{\text{best}}(x_i^k) \triangleq \min_{x_i \in \mathcal{X}} E[f(x_i^k)]$. Then, in the next result, we show that by cooperating through the proposed algorithm all the agents agree on a common solution and the produced sequences $\{x_i^k\}$ are asymptotically cost optimal in expected value when $K \to \infty$.

**Theorem 1.** Let Assumptions 1 and 2 hold and let $\{x_i^k\}_{k \geq 0}$ be the sequences generated through the MICKY algorithm. Then, if the sequences $\{\alpha_i^k\}$ satisfy

$$\sum_{k=0}^{\infty} \alpha_i^k = \infty, \quad \sum_{k=0}^{\infty} (\alpha_i^k)^2 < \infty, \quad \alpha_i^{k+1} \leq \alpha_i^k \quad (9)$$

for all $k$ and all $i \in V$, it holds that,

$$\lim_{k \to \infty} E[\|x_i^k - \bar{x}^k\|] = 0, \quad (10)$$

and

$$\lim_{k \to \infty} f_{\text{best}}(x_i^k) = f(\bar{x}^k), \quad (11)$$

being $\bar{x}^*$ the optimal solution to (2).

**Proof.** By using the same arguments used in (Farina and Notarstefano, 2019, Lemma 3.1), it can be shown that $x_i^k \geq x_j^k$ for all $k$ and all $i, j \in V$. Then (10) follows from (Farina and Notarstefano, 2019, Lemma 5.11). Moreover, as anticipated, it can be shown that $x_i^k$ is the $i$-th block of a subgradient of the function $f_j(x)$ in problem (2) (see, e.g., (Bach et al., 2013, Section 3.2)). In fact, being $f_j$ defined as the support function of the base polyhedron $B(F_j)$, i.e., $f_j(x) = \max_{w \in B(F_j)} w^T x$, the greedy algorithm (Bach et al., 2013, Section 3.2) iteratively computes a subgradient of $f_j \alpha_i^k$ component by component. Moreover, subgradients of $f_j$ are bounded by some constant $G < \infty$, since every component of a subgradient of $f_j$ is computed as the difference of $F_j$ over two different subsets of $V$. Given that, the proposed algorithm can be seen as a special instance of the Distributed Block Proximal Method in Farina and Notarstefano (2019). Thus, since Assumptions 1 and 2 hold, it inherits all the convergence properties of the Distributed Block Proximal Method and under the assumption of diminishing stepsizes (9) respectively, the result in (11) follows (see (Farina and Notarstefano, 2019, Theorem 5.15)). □

Notice that the result in Theorem 1 does not say anything about the convergence of the sequences $\{x_i^k\}$, but only states that if diminishing stepsizes are employed, asymptotically these sequences are consensual and cost optimal in expected value.

Despite that, from a practical point of view, two facts typically happen. First, agents approach consensus, i.e., for all $i \in \{1, \ldots, N\}$, the value $\|x_i^k - \bar{x}^k\|$ becomes small, extremely fast, so that they all agree on a common solution. Second, if the number of iterations $K$ in the algorithm is sufficiently large, the value of $x_i^K$ is a good solution to problem (2). Then, given $x_i^K$, each agent can reconstruct a set solution to problem (1) by using (8) and, in order to obtain the same solution for all the agents, we consider a unique threshold value, known to all the agents, $\tau \in [0, 1]$. Recomorphic 2.1. Notice that, by resorting to classical arguments, it can be easily shown from the analysis in Farina and Notarstefano (2019) that the convergence rate of $f_{\text{best}}$ in Theorem 1 is sublinear (with explicit rate depending on the actual stepsize sequence). Moreover, if constant stepsizes are employed, convergence of $f_{\text{best}}$ to the optimal solution is attained in expected value with a constant error with rate $O(1/k)$ (Farina and Notarstefano, 2019, Theorem 2).

3. COOPERATIVE IMAGE SEGMENTATION

Submodular minimization has been widely applied to computer vision problems as image classification, segmentation and reconstruction, see, e.g., Stobbe and Krause (2010); Jegelka et al. (2013); Greig et al. (1989). In this section, we consider a binary image segmentation problem in which $N = 8$ agents have to cooperate in order to separate an object from the background in an image of size $D \times D$ pixels (with $D = 64$). Each agent has access only to a portion of the entire image, see Figure 2, and can communicate according to the graph reported in the figure.

Before giving the details of the distributed experimental setup let us introduce how such a problem is usually treated in a centralized way, i.e., by casting it into a $s$-$t$ minimum cut problem.

3.1 $s$-$t$ minimum cut problem

Assume the entire $D \times D$ image be available for segmentation, and denote as $V = \{1, \ldots, D^2\}$ the set of pixels. As shown, e.g., in Greig et al. (1989); Boykov and Funka-Lea (2006) this problem can be reduced to an equivalent $s$-$t$ minimum cut problem, which can be approached by submodular minimization techniques.

More in detail, this approach is based on the construction of a weighted digraph $G_{s-t} = (V_{s-t}, E_{s-t}, A_{s-t})$, where $V_{s-t} = \{1, \ldots, D^2, s, t\}$ is the set of nodes, $E_{s-t} \subseteq V_{s-t} \times V_{s-t}$ is the edge set and $A_{s-t}$ is a positive weighted adjacency matrix. There are two sets of directed edges $(s, p)$ and $(p, t)$, with positive weights $a_{s,p}$ and $a_{p,t}$ respectively, for all $p \in V$. Moreover, there is an undirected edge $(p, q)$ between any two neighboring pixels with weight $a_{p,q}$. The
weights $a_{s,p}$ and $a_{p,t}$ represent individual penalties for assigning pixel $p$ to the object and to the background respectively. On the other hand, given two pixels $p$ and $q$, the weight $a_{p,q}$ can be interpreted as a penalty for a discontinuity between their intensities.

In order to quantify the weights defined above, let us denote by $I_p \in [0,1]$ the intensity of pixel $p$. Then, see, e.g., Boykov and Funka-Lea (2006), $a_{p,q}$ is computed as

$$a_{p,q} = e^{-\frac{(I_p-I_q)^2}{2\sigma^2}},$$

where $\sigma$ is a constant modeling, e.g., the variance of the camera noise. Moreover, weights $a_{s,p}$ and $a_{p,t}$ are respectively computed as

$$a_{s,p} = -\lambda \log P(x_p = 1),$$
$$a_{p,t} = -\lambda \log P(x_p = 0),$$

where $\lambda > 0$ is a constant and $P(x_p = 1)$ (respectively $P(x_p = 0)$) denotes the probability of pixel $p$ to belong to the foreground (respectively background).

The goal of the $s$-$t$ minimum cut problem is to find a subset $X \subseteq V$ of pixels such that the sum of the weights of the edges from $X \cup \{s\}$ to $\{t\} \cup V \setminus X$ is minimized.

### 3.2 Distributed set-up

In the considered distributed set-up, $N = 8$ agents are connected according to a strongly-connected Erdős-Rényi random digraph and each of them has access only to a portion of the image (see Figure 2). In this set-up, clearly, each agent can assign weights only to some edges in $E_{s-t}$ so that, it cannot segment the entire image on its own.

Let $V_i \subseteq V$ be the set of pixels seen by agent $i$. Each node $i$ assigns a local intensity $I^i_p$ to each pixel $p \in V_i$. Then, it computes its local weights as

$$a^i_{p,q} = \begin{cases} 
\frac{e^{-\frac{(I^i_p-I^i_q)^2}{2\sigma^2}}}{x_p}, & \text{if } p, q \in V_i \\
0, & \text{otherwise}
\end{cases}$$
$$a^i_{s,p} = \begin{cases} 
-\lambda \log P(x^i_p = 1), & \text{if } p \in V_i \\
0, & \text{otherwise}
\end{cases}$$
$$a^i_{p,t} = \begin{cases} 
-\lambda \log P(x^i_p = 0), & \text{if } p \in V_i \\
0, & \text{otherwise}
\end{cases}$$

Given the above locally defined weights, each agent $i$ construct its private submodular function $F^i$ as

$$F^i(X) = \sum_{p \in X} a^i_{p,q} + \sum_{q \in V \setminus X} a^i_{s,q} + \sum_{p \in X} a^i_{p,t} - \sum_{q \in V} a^i_{s,q}. \quad (12)$$

We applied the MICKY distributed algorithm to this set-up and we split the optimization variable in 40 blocks. In order to mimic possible errors in the construction of the local weights, we added some random noise to the image. We implemented the MICKY algorithm by using the Python package DISROPT (Farina et al. 2019) and we ran it for $K = 1000$ iterations. A graphical representation of the results is reported in Figure 3. Each row is associated to one network agent while each column is associated to a different time stamp. More in detail, we show the initial condition at time $k = 0$ and the candidate (continuous) solution at $k \in \{100, 200, 300, 400, 500, 600\}$ iterations. The last column represents the solution $X^k_{\text{final}}$ of each agent obtained by thresholding $x^k$ with $k = 1000$ and $\tau = 0.5$. As appearing in Figure 3, the local solution set estimates $X^k_{\text{final}}$ are almost identical. Moreover, the connectivity structure of the network clearly affects the evolution of the local estimates. Finally, the evolution of the cost error is depicted in Figure 4, where $X^k_i \triangleq \{i \mid x^k_i > \tau\}$.

### 4. CONCLUSIONS

In this paper we presented MICKY, a distributed algorithm for solving submodular problems involving the minimization of the sum of many submodular functions without any central unit. It involves random block updates and communications, thus requiring a reduced local computational load and allowing its deployment on networks with low communication bandwidth (since it requires a small amount of information to be transmitted at each iteration). Its convergence in expected value has been shown under mild assumptions. The MICKY algorithm has been tested...
on a cooperative image segmentation problem in which each agent has access to only a portion of the entire image.

REFERENCES


