Distributed Trust-Region Method With the First Order Models

Aleksandar Armacki*, Dusan Jakovetić*, Nataša Krejić*, Nataša Krklec Jerinkić*
*Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad
{aleksandar.armacki, dusan.jakovetic, natasa, natasa.krklec}@dmi.uns.ac.rs

***Abstract*—In this paper, we introduce the trust region concept for distributed optimization. A large class of globally convergent methods of this type is used efficiently in centralized optimization, both constrained and unconstrained. The methods of this class are built on the idea of modeling the objective function at each iteration and taking the new iteration as the minimizer of the model in a certain area, called the trust region. The trust region size, the minimization method and the model function depend on the properties of the objective function. In this paper we propose a general framework and concentrate on the first order methods, i.e. the gradient methods. Using the trust-region mechanism for generating the step size we end up with a fully distributed method with node varying step sizes. Numerical results presented in the paper demonstrate the efficiency of the proposed approach.**

***Index Terms*—Distributed optimization, Trust region method, multi-agent network**

I. Introduction

We consider a network of N agents, each of which has access to a local (convex) function $f_i : \mathbb{R}^d \mapsto \mathbb{R}$. Agents cooperate locally, with the view of minimizing the aggregate objective function defined as:

$$f(x) = \sum_{i=1}^{N} f_i(x) \tag{1}$$

Problems of this form attract a lot of interest, as they are applicable in many areas, like distributed inference in sensor networks [9], and large-scale machine learning [14], [2], [3].

A number of iterative methods for solving (1) in distributed environment is proposed and analysed in the literature. A large class of gradient methods is based on the idea of two step procedure: each node makes an update in the direction of (local) negative gradient using a certain step size and after that all nodes update their iteration through a consensus step. Important questions in all of these methods are the direction of the update at each step and the step size. A number of methods with constant step size [11], diminishing step sizes [8] and so on, are analysed in the literature. Furthermore, the convergence towards the exact solution of (1) can be obtained with a suitable direction that is a local (node specific) approximation of the global gradient, for details see [7] and references therein. Second order methods are also successfully applied in distributive environment for solving (1), see [1], [10].

Aleksandar Armacki is supported by the H2020 project Ibidaas, Grant Agreement No 780787, while the other authors are supported by Ibidaas and Serbian Ministry of Education, Science and Technological Development, grant no. 174030.

Trust region framework is widely used in centralized optimization as a global solver for both constrained and unconstrained problems. A comprehensive theory with implementation details is available in [5]. A gradient method with adapted step sizes based on the trust region approach is recently analysed in [6]. The main ingredients of a trust region method are the model function, the trust region size and a solution of the model in the trust region area, and all three ingredients are updated in each iteration. The trust region approach can be adopted to nonconvex case, adding a regularization term to the second order approximation model, [5] and to derivative free environment [4].

In this paper we introduce the distributed trust region framework suitable for solving (1). The key modification aims to make the standard method fully distributed and is achieved through a consensus part. Each node works with a local function and its' model, and local trust region size as well. The general trust region rule for accepting an iteration or decreasing the trust region size, as in the classical trust region approach, is applied locally, by each node, while the consensus step ensures that the full objective function is considered. The general distributed framework is then further developed for the first order model functions in each iteration. This particular method in fact results in distributed gradient method with node specific and time varying step sizes. The method is numerically tested and its performance is compared with the standard distributed gradient method [11].

The rest of the paper is organized as follows. Section II introduces the concept of Trust-Region in centralized optimization. The proposed algorithm is introduced in Section III, while Section IV presents numerical results. Section V concludes the paper, discussing the model and ideas for future work.

II. Trust Region Methods

In this section, we review the concept of trust region in centralized settings.

A. General Framework

Trust region encompasses a wide range of optimization methods. The general idea is to generate a model that is a good approximation of the objective function in a specific region, and then find the step that minimizes the model function in

that specific region, determined by the trust-region size. Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be the objective function, and let x^k and Δ^k be the current approximate solution and trust region size at iteration k, respectively. Then, at the $(k+1)^{th}$ iteration, in a typical trust region method one builds a quadratic model function

$$m^k(p) = f(x^k) + \nabla^T f(x^k)p + \frac{1}{2}p^T B^k p. \tag{2}$$

Here, $B^k \in \mathbb{R}^{d\times d}$ is a symmetric matrix which approximates the Hessian $\nabla^2 f(x^k)$. If $B^k = \nabla^2 f(x^k)$ we say that the methods is Newton trust region. The next step is to minimize the model within the trust region, i.e. the step p^k is generated as

$$p^k = \underset{p\in\mathbb{R}^d}{\arg\min}\, m^k(p) \text{ s.t. } \|p\| \le \Delta^k. \tag{3}$$

The step p^k can be generated as the exact minimizer of (3) or an inexact minimizer. Under suitable conditions, like positive definiteness of B^k, one can guarantee that the problem (3) is solvable and thus the reduction in the model is obtained, i.e., $m^k(0) - m^k(p^k) > 0$. Keeping in mind that the aim is to minimize the objective function f, we define the predicted reduction as the reduction in the model value,

$$pred = m^k(0) - m^k(p^k),$$

and the actual reduction in the objective function

$$ared = f(x^k) - f(x^k + p^k).$$

Based on their quotient,

$$\rho^k = \frac{f(x^k) - f(x^k + p^k)}{m^k(0) - m^k(p^k)} \tag{4}$$

the iteration is either successful and $x^{k+1} = x^k + p^k$, or unsuccessful, $x^{k+1} = x^k$. In the later case one has to reduce the trust region size Δ^k and repeat the process. The size of the trust region is usually adapted in a bit more complex way than the simple statement successful/unsuccessful as precisely stated in the Algorithm 1 below. But roughly speaking the reasoning behind the update is the following. If the actual reduction is too small (or even non-existing) then the model function does not approximate the objective function well and therefore we need to shrink the region and repeat the process. If there is a good agreement between the actual reduction in the objective function and the model reduction then the trust region size is kept the same. Finally, if the reduction in the model and objective function are close enough then we conclude that the approximation is very good and increase the trust region size hoping to obtain a larger step in the next iteration.

Algorithm 1 Trust-Region method

Input: $0 < \Delta_{\min} < \Delta_{\max}$, $\Delta^0 \in (\Delta_{\min}, \Delta_{\max})$, $\eta \in \left[0, \frac{1}{4}\right)$, $x^0 \in \mathbb{R}^d$

for $k = 0, 1, \ldots$ **do**
  Obtain p^k by solving (3)
  Evaluate ρ^k from (4)
  if $\rho^k < \frac{1}{4}$ **then**
    $\Delta^{k+1} = \frac{1}{4}\Delta^k$
  else
    if $\rho^k > \frac{3}{4}$ and $\|p^k\| = \Delta^k$ **then**
      $\Delta^{k+1} = \min\left\{2\Delta^k, \Delta_{max}\right\}$
    else
      $\Delta^{k+1} = \Delta^k$
    end if
  end if
  if $\rho^k > \eta$ **then**
    $x^{k+1} = x^k + p^k$
  else
    $x^{k+1} = x^k$
  end if
end for

B. The Cauchy Point

The Cauchy Point is particularly important as it defines the minimal decrease in the objective function needed for convergence. It is based on the first order model, i.e. the steepest descent direction. Let us consider the linear model and its solution p^k,

$$p^k = \underset{p\in\mathbb{R}^d}{\arg\min}\, f(x^k) + \nabla^T f(x^k)p \ s.t.\ \|p^k\| \le \Delta^k. \tag{5}$$

It is easy to verify that the closed-form expression for p^k is given by

$$p^k = -\frac{\Delta^k}{\|\nabla f(x^k)\|}\nabla f(x^k). \tag{6}$$

Now, the Cauchy point is obtained as the minimizer of the model function (2) along the search direction (6)

$$\tau^k = \underset{\tau\ge 0}{\arg\min}\, m^k(\tau p^k) \ s.t.\ \|\tau^k p^k\| \le \Delta^k\,. \tag{7}$$

So, the Cauchy point is defined as:

$$p_c^k = \tau^k p^k, \tag{8}$$

with

$$\tau^k = \begin{cases} 1, & \nabla^T f(x^k) B^k \nabla f(x^k) \le 0 \\ \min\{\frac{\|\nabla f(x^k)\|^3}{(\nabla^T f(x^k) B^k \nabla f(x^k)\Delta^k)}, 1\}, & \nabla^T f(x^k) B^k \nabla f(x^k) > 0 \end{cases} \tag{9}$$

Under a set of standard assumptions, see [5] a trust region method generates a convergent sequence if the sufficient decrease condition

$$m_k(0) - m_k(p^k) \ge \theta(m_k(0) - m_k(p_c^k))$$

holds.

If we assume that the model function is linear, i.e. $B^k = 0$ in each iteration then the trust region method actually reduces

to the gradient method with the variable step sizes (6). The convergence conditions in this case are standard, including the smooth gradient and boundedness of the objective function from below.

III. Distributed Trust-Region

In this section we introduce the trust region concept in distributed settings. In Subsection III-A the general formulation of distributed optimization problems is introduced, while in Subsection III-B the proposed algorithm is described.

A. Network and Optimization Models

The goal in distributed optimization is to cooperatively solve (1). Each agent keeps its local copy of the solution, $x_i \in \mathbb{R}^d$, and the aim is for agents to achieve consensus, i.e. to achieve $x_1 = x_2 = \ldots = x_n$.

Additionally, each agent can exchange information with other agents in the network. The network is modeled as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where vertices represent agents, while edges represent communication links between them. We make the following standard assumptions about the network model:

Assumption A1. The underlying graph $\mathcal{G}$ is an undirected and connected graph.

Assumption A2. The matrix $W \in \mathbb{R}^{n\times n}$ is symmetric and doubly stochastic, with elements w_{ij} such that $w_{ij} > 0$ if $\{i,j\} \in \mathcal{E}$, $w_{ij} = 0$ if $\{i,j\} \notin \mathcal{E}$ and $w_{ii} = 1 - \sum_{j\in\mathcal{N}_i} w_{ij}$. Here, $\mathcal{N}_i$ denotes the set of all vertices adjacent to vertice i.

Assumption A1 ensures that communication can go both ways, i.e. if $\{i,j\} \in \mathcal{E}$, then agent i can communicate with agent j and vice-versa. It also ensures that a consensus between all agents is possible, since the network is connected.

B. Proposed Method

The method we propose is a fully distributed trust region based method. In this section we use subscripts to refer to agents, while superscripts refer to the iteration counter.

Assume that the method is initiated with the upper and the lower region size bounds, denoted by Δ_{max} and Δ_{min}, respectively[1]. Let us consider the agent i at the iteration k. The agent has the local approximate solution x_i^k and the trust region size parameter Δ_i^k. The general concept is then the following. First, a local (quadratic) model function is defined,

$$m_i^k(p) = f_i(x_i^k) + \nabla^T f_i(x_i^k)p + \frac{1}{2}p^T B_i^k p. \tag{10}$$

The minimizer of the model (exact or inexact) is then computed

$$p_i^k = \arg\min_{p\in\mathbb{R}^d} m_i^k(p) \; s.t. \; \|p\| \leq \Delta_i^k. \tag{11}$$

After that, the node compares the predicted and actual reduction computing

$$\rho_i^k = \frac{f_i(x_i^k) - f_i(z_i^k + p_i^k)}{m_i^k(0) - m_i^k(p_i^k)}, \tag{12}$$

[1]It is possible to define trust region method without lower and upper bound on the trust region size.

with

$$z_i^k = \sum_{j\in\mathcal{N}_i\cup\{i\}} w_{ij} x_j^k. \tag{13}$$

Notice that here we have different definition of the actual reduction than in the classical trust region approach. Namely the actual reduction is computed at the (subject to acceptance criterion) next point which is the consensus point plus correction $z_i^k + p_i^k$. Computation of the consensus point z_i^k is the only step in the algorithm where inter-agent communication takes place. The decision on the next (local) trust region size is based on the same criterion as in the previous Section. If $\rho_i^k < 1/4$ then $\Delta_i^{k+1} = \max\{1/4\Delta_i^k, \Delta_{min}\}$ otherwise if $\rho_i^k > 3/4$, the trust region size is enlarged, $\Delta_i^{k+1} = \min\{2\Delta_i^k, \Delta_{\max}\}$, and if $\rho_i^k \in [1/4, 3/4]$ then the trust region size is kept unchanged, $\Delta_i^{k+1} = \Delta_i^k$. The reasoning behind this trust region size update is the same, if the quotient of the actual and predicted reduction is too small then the model function is not a good approximation of the objective function in the considered region and we need to shrink the region. On the contrary, if the agreement between actual and predicted reduction is good (close to 1) then we might consider enlarging the region, to obtain larger steps in the next iteration. The crucial difference with respect to the centralized trust region method is the update of the next iteration. Although the criterion for successful iteration is defined as before with the coefficient ρ_i^k defined in (12), i.e., if $\rho_i^k > \eta_i$, for $\eta_i \in (0, 1/4)$, the update is performed as

$$x_i^{k+1} = z_i^k + p_i^k.$$

Otherwise the iteration is unsuccessful and $x_i^{k+1} = z_i^k$. This way, we not only shrink the region, but move the local solution toward a consensus solution, with the aim of obtaining a better model function (10). Thus, the proposed method is fully distributed, as each agent computes all the necessary information locally, with the exception of z_i^k, which is obtained by exchanging information only with neighbouring agents, as explained above.

In this paper we are particularly interested in the linear model trust region method, i.e. in the gradient based method. Thus we will consider the model functions

$$m_i^k(p) = f_i(x_i^k) + \nabla^T f_i(x_i^k)p + \frac{1}{2}p^T Ip,$$

where $I \in \mathbb{R}^{d\times d}$ represents the identity matrix. Thus each node solves the problem

$$\min \; f_i(x_i^k) + \nabla^T f_i(x_i^k)p, \;\; \text{s.t. } \|p\| \leq \Delta_i^k \tag{14}$$

for p_i^k defined as

$$p_i^k = -\frac{\Delta_i^k}{\|\nabla f_i(x_i^k)\|}\nabla f_i(x_i^k)\,. \tag{15}$$

After that the Cauchy coefficient is computed as in (7) and we get

$$\tau_i^k = \min\Big\{\frac{\|\nabla f(x_i^k)\|}{\Delta_i^k}, 1\Big\}, \tag{16}$$

and

$$p_{i,C}^k = \tau_i^k p_i^k. \tag{17}$$

Equation (17) shows that the the step-size is adaptively adjusted, based on the performance in the previous iteration. Furthermore, the step size is locally computed, i.e., node specific. The algorithm below states formally the first order model trust region method, listing one iteration at a node i.

Algorithm 2 DTR method at Agent i

Input: $\Delta_{min}, \Delta_{max} > 0$, $\Delta_i^0 \in (\Delta_{min}, \Delta_{max})$, $\eta_i \in \left[0, \frac{1}{4}\right)$, $x_i^0 \in \mathbb{R}^d$

for $k = 0, 1, \ldots$ **do**
  Compute $p_{i,C}^k$ as in (17)
  Evaluate ρ_i^k from (12)
  if $\rho_i^k < \frac{1}{4}$ **then**
    $\Delta_i^{k+1} = \max\left\{\frac{1}{4}\Delta_i^k, \Delta_{min}\right\}$
  else
    if $\rho_i^k > \frac{3}{4}$ and $\|p_i^k\| = \Delta_i^k$ **then**
      $\Delta_i^{k+1} = \min\left\{2\Delta_i^k, \Delta_{max}\right\}$
    else
      $\Delta_i^{k+1} = \Delta_i^k$
    end if
  end if
  if $\rho_i^k > \eta_i$ **then**
    compute z_i^k by (13) and take $x_i^{k+1} = z_i^k + s_i^k$
  else
    compute z_i^k by (13) and take $x_i^{k+1} = z_i^k$
  end if
end for

IV. Numerical Results

In this section, we present the results obtained from numerical experiments. In Subsection IV-A distributed models used as benchmarks are introduced. In Subsection IV-B we define the problem used to evaluate the performances of the distributed models. Finally, in Subsection IV-C we present the results.

A. Benchmark

We benchmarked the performances of the proposed algorithm with distributed gradient descent (DGD) [11].

DGD is a well-known first-order method that uses the following update rule:

$$x_i^{k+1} = z_i^k - \alpha \nabla f_i(x_i^k), \tag{18}$$

where z_i^k is defined as in (13), while α is a fixed step-size and $x_i^0 \in \mathbb{R}^d$ is usually chosen to be the zero vector.

B. Problem and Parameters

We evaluated the performances of the methods on a binary classification problem, using logistic regression. The objective function is defined as:

$$f(x) = \sum_{i=1}^{N} \left[\frac{\lambda}{2N}\|x\|^2 + \sum_{j=1}^{q_i} \log\left(1+\exp\left(-v_{ij}u_{ij}^T x\right)\right)\right], \tag{19}$$

where λ is the regularization parameter, q_i represents the number of samples available to agent i, $v_{ij} \in \{-1, 1\}$ represents the class label of the j^th sample at agent i, while $u_{ij} \in \mathbb{R}^d$ is the j^{th} sample of agent i.

We use $N = 30$ agents, and each has $q_i = 50$ samples. The dimension of the problem is $d = 10$, and the regularization parameter is set to $\lambda = 10^{-4}$. We define the weight matrix using the Metropolis weighting scheme:

$$w_{ij} = \begin{cases} 0, & j \notin \mathcal{N}_i \\ \frac{1}{2(1+\max\{d_i, d_j\})}, & j \in \mathcal{N}_i \\ 1 - \sum_{k\in\mathcal{N}_i} w_{ik}, & j = i \end{cases} \tag{20}$$

where d_i denotes the degree of vertice i.

We calculate the approximate Lipschitz constant for the gradient of f as

$$L = \frac{\lambda}{N} + C \max_{i,j} \|u_{ij}\|^2, \tag{21}$$

where $C = \max_i q_i$. In our example, $C = 50$. We use L to calculate the step-size for DGD as $\alpha = 1/(100L)$. For DTR, we set the region size boundaries at $\Delta_{min} = 10^{-2}$ and $\Delta_{max} = 10^5$. Each agent had the same initial region size $\Delta_i^0 \in [\Delta_{min}, \Delta_{max}]$.

C. Results

We generated two synthetic datasets to test the performances of the methods in the following way: each component of the sample vector u_{ij} was generated from the normal distribution. If the j^{th} sample of agent i belongs to the class labeled 1, its components are drawn from the normal distribution with mean $\mu = 3$ and standard deviation $\sigma = 1$. Otherwise, the components of the sample vector are drawn from the normal distribution with mean $\mu = -3$ and standard deviation $\sigma = 1$. The dataset generated in this way is linearly separable. We also generated a nonseparable dataset, with parameters $\mu = 2$ and $\sigma = 3$ if the j^{th} sample of agent i belongs to the class labeled 1 and $\mu = -2$ and $\sigma = 3$ if the j^{th} sample of agent i belongs to the class labeled -1.

We present 3 plots: the optimality gap, the behaviour of parameter τ_i^k, defined in (16), and the behaviour of Δ_i^k, the region size parameter. The optimality gap is defined as:

$$x^k - x^* = \frac{1}{N}\sum_{i=1}^{N} \frac{\|x_i^k - x^*\|}{\|x^*\|} \tag{22}$$

and x^* represents the optimal point, calculated using the centralized gradient descent algorithm. For τ and Δ, we plot the minimum, maximum and mean values of all agents' values at each iteration. Additionally, we randomly choose an agent and follow the behaviour of its parameters. Those values are denoted by 'test' in the graphs.

We can see from Fig. 2 and Fig. 3 that the region size stabilizes at the 7^{th} iteration, approximately the same iteration as τ starts to decrease. This represents the moment that the region size becomes an inactive constraint, and the algorithm begins to take the full gradient step, as defined in (15).

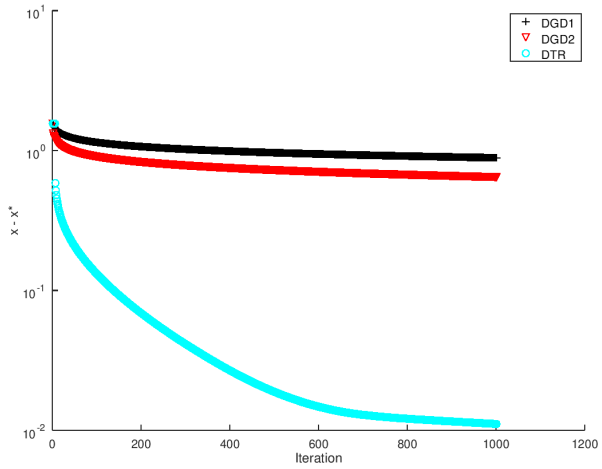

Fig. 1. Optimality gap versus the number of iterations for separable data

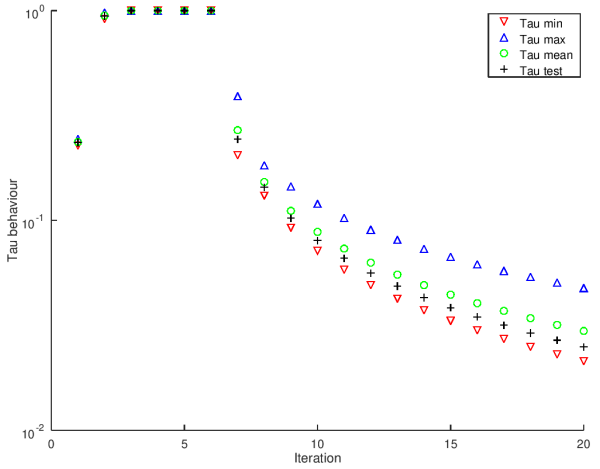

Fig. 3. Behaviour of τ versus the number of iterations for separable data

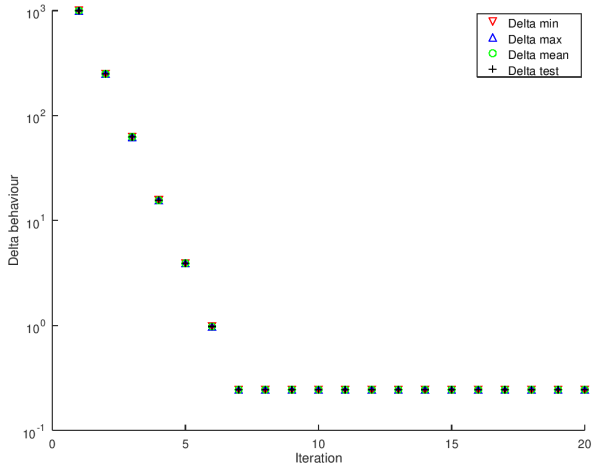

Fig. 2. Region size behaviour versus the number of iterations for separable data

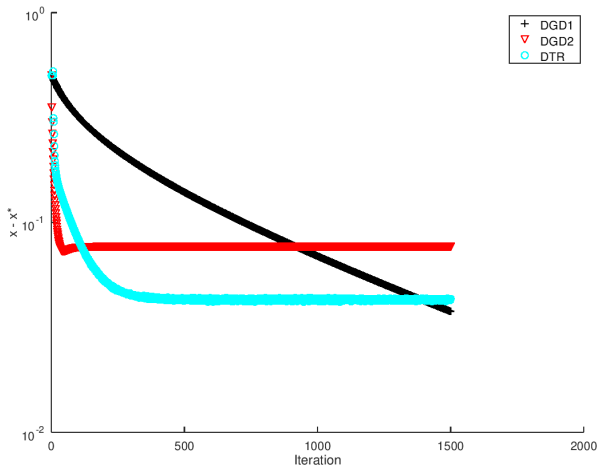

Fig. 4. Optimality gap versus the number of iterations for separable data

V. Conclusion

The method proposed in this paper, DTR, is a distributed version of the Trust-Region optimization method. The key novelty developed here is the step-size selection which is defined in a Trust-Region manner, thus allowing for adaptable, locally computable step-sizes. The numerical experiments show favorable results compared to DGD. Some of the ideas for future work include using a different Hessian approximation in (10) in order to incorporate a stronger second-order information in the search direction, and convergence analysis, to name a few.

References

[1] Dragana Bajovic, Dusan Jakovetic, Natasa Krejic, and Natasa Krklec Jerinkic. Distributed second order methods with variable number of working nodes. *CoRR*, abs/1709.01307, 2017.

[2] Ron Bekkerman, Mikhail Bilenko, and John Langford. *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[3] V. Cevher, S. Becker, and M. Schmidt. Convex optimization for big data: Scalable, randomized, and parallel algorithms for big data analytics. *IEEE Signal Processing Magazine*, 31(5):32–43, Sep. 2014.

[4] Andrew R. Conn, Katya Scheinberg, and Luis N. Vicente. *Introduction to Derivative-Free Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[5] A.R. Conn, N.I.M. Gould, and P.L. Toint. *Trust Region Methods*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2000.

[6] Frank E. Curtis, Katya Scheinberg, and Rui Shi. A Stochastic Trust Region Algorithm Based on Careful Step Normalization. *arXiv e-prints*, page arXiv:1712.10277, December 2017.

[7] Dusan Jakovetic. A unification, generalization, and acceleration of exact distributed first order methods. *CoRR*, abs/1709.01317, 2017.

[8] D. Jakoveti, J. Xavier, and J. M. F. Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, May 2014.

[9] S. Kar, J. M. F. Moura, and K. Ramanan. Distributed parameter estimation in sensor networks: Nonlinear observation models and imperfect

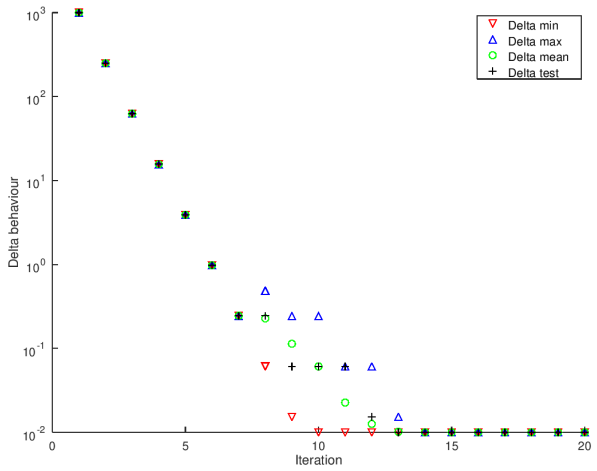

Fig. 5. Region size behaviour versus the number of iterations for separable data

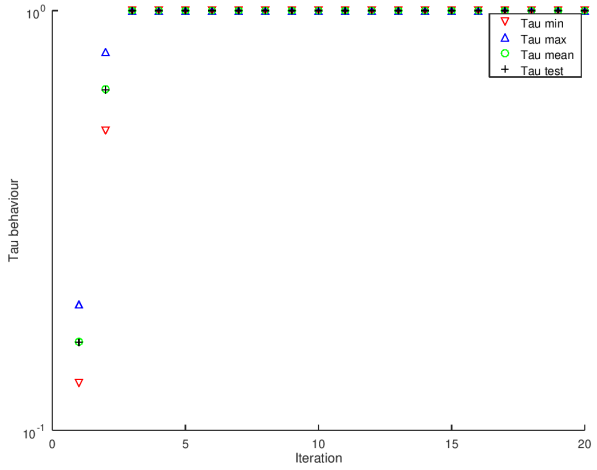

Fig. 6. Behaviour of τ versus the number of iterations for separable data

communication. *IEEE Transactions on Information Theory*, 58(6):3575–3605, June 2012.

[10] Aryan Mokhtari, Qing Ling, and Alejandro Ribeiro. Network newton-part ii: Convergence rate and implementation. 04 2015.

[11] A. Nedic and A. Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, Jan 2009.

[12] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[13] Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 159–166, 2016.

[14] K. I. Tsianos, S. Lawlor, and M. G. Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1543–1550, Oct 2012.

[15] Zifeng Wang, Qing Ling, and Wotao Yin. Decentralized bundle method for nonsmooth consensus optimization. pages 568–572, 11 2017.

[16] K. Yuan, Q. Ling, and W. Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.