Newton's Method for Constrained Norm Minimization

Mahmoud El Chamie[1] Giovanni Neglia[1]

***Abstract*— Due to increasing computer processing power, Newton's method is receiving again increasing interest for solving optimization problems. In this paper, we provide a methodology for solving a general norm optimization problem under some linear constraints using the Newton's method. This problem arises in many machine learning and graph optimization applications. We consider as a case study optimal weight selection for average consensus protocols for which we show how Newton's method significantly outperforms gradient methods both in terms of convergence speed and in term of robustness to the step size selection.**

I. Introduction

Solutions of actual optimization problems rarely can be expressed in a closed-form. More often they can be obtained through iterative methods, that can be very effective in some cases (e.g. when the objective function is convex). Among the iterative approaches, gradient methods converge under quite general hypotheses, but they suffer from very slow convergence rates as they are coordinate dependent (scaling variables in the problem affect the convergence speed). The Newton's method converges locally quadratically fast and is coordinate independent, and in presence of constraints, they can be addressed by considering KKT conditions [1] with the drawback that Newton's method requires the knowledge of the Hessian of the function that may be computationally too expensive to calculate. However, with the continuous increase of computation power and the existence of efficient algorithms for solving linear equations, Newton's method is again the object of an increasing interest (e.g. [2], [3], [4]).

In this paper, we deal with an optimization problem that appears in many application scenarios. Up to our knowledge, exact line search Newton method is not yet developed for constrained schatten p-norm problems and are usually solved by first order gradient methods. The optimization problem we are interested in is the following:

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & ||X||_{\sigma p} \\ \text{subject to} \quad & \phi(X) = \mathbf{y}, \\ & X \in \mathbb{R}^{n_1,n_2} \ , \ \mathbf{y} \in \mathbb{R}^c, \end{aligned} \tag{1}$$

where $||X||_{\sigma p}$ is the Schatten p-norm of the matrix X which is the L-p norm of its singular values, i.e. $||X||_{\sigma p} = (\sum_i \sigma_i^p)^{1/p}$, and $\phi(X)$ is a linear function of the elements of X.

The Schatten p-norm is orthogonally invariant and is often considered in machine learning for the regularization problem in applications such as multi-task learning [5], collaborative filtering [6] and multi-class classification [7]. The authors in [8] refer to problem (1) as the *minimal norm interpolation* problem. However, the problem is not just limited to machine learning, and it can also include graph optimization problems where X is the square weighted adjacency matrix. In particular, in what follows we will consider as a case study the calculation of weights that guarantee fast convergence of average consensus protocols [9].

The main obstacle to apply Netwon method is the difficulty to calculate the Hessian and for this reason slower gradient methods are preferred. However, in this paper, we show that in problem (1) by exploiting the special structure of the objective function, constraints linearity, and by carefully rewriting the Schatten norm problem by stacking the columns of the matrix to form a long vector, we can easily calculate explicitly both the gradient and the Hessian. While we still need to invert the Hessian numerically, this matrix has lower dimension than the typical KKT matrix used in Newton's methods for solving such constrained problems. We then consider a specific case study in this class of optimization problems, i.e. determining optimal weights for consensus protocols, for which we specify an approximate easy-to-implement line search algorithm. Interestingly, using the proposed method, we give a closed form solution for the special case of $p = 2$. Simulations are carried to show the advantage of this method over used gradients techniques.

We adopt in this paper the notation that bold small letters are used for vectors (ex: $\mathbf{x}$ is a vector and x_l is its l-th component), and capital letters for matrices (ex: X is a matrix and x_{ij} or $X_{i,j}$ is the element of row i and column j in that matrix). Let $Tr(.)$ be the trace of a matrix, vect(.) denotes the operation that stacks the columns of an n_1 by n_2 matrix in one vector of dimensions $n_1 n_2 \times 1$, and diag(.) denotes the operation of changing a vector into a diagonal matrix by placing its elements on the diagonal. The paper is organized as follows. In Sec. II we provide some background on Newton's methods. Sec. III presents the general methodology we have developed, while Sec. IV shows a specific case study.

II. Preliminaries

A. The gradient and Hessian of a scalar function

In this section we provide the notation used across this paper for the gradient and the Hessian of scalar functions. The functions of our interest can have as input a vector or a matrix.

1) Scalar function of a vector: Given a function $h : \mathbb{R}^m \rightarrow \mathbb{R}$, the gradient of the function $h(\mathbf{x})$ with respect

[1]INRIA Sophia Antipolis-Méditerranée, 2004 route des Lucioles - BP 93. 06902 Sophia Antipolis Cedex, France. Emails: { mahmoud.el_chamie, giovanni.neglia}@inria.fr

to the vector $\mathbf{x} \in \mathbb{R}^m$ denoted by $\nabla_{\mathbf{x}} h \in \mathbb{R}^m$ is given by:

$$(\nabla_{\mathbf{x}} h)_l = \frac{\partial h}{\partial x_l} \quad \text{for } l = 1, \ldots, m \ .$$

The Hessian of the function $h(\mathbf{x})$ is denoted by the matrix $\nabla^2_{\mathbf{x}} h \in \mathbb{R}^{m,m}$ whose elements are given by the following equation:

$$\left(\nabla^2_{\mathbf{x}} h\right)_{l,k} = \frac{\partial^2 h}{\partial x_l \partial x_k} \quad \text{for } l, k = 1, \ldots, m \ .$$

2) Scalar function of a matrix: Let us consider the function $h : \mathbb{R}^{n_1,n_2} \to \mathbb{R}$. We define its gradient and Hessian considering it as a function of the vector vect(X), obtained stacking the columns of the matrix X^{n_1,n_2} in one vector of dimensions $n_1 n_2 \times 1$. With some abuse of notation we will consider that $h(X) = h(\text{vect}(X))$. We can then define the gradient as follows:

$$\begin{aligned}(\nabla_X h)_{(j-1)n_1+i} &= \left(\nabla_{\text{vect}(X)} h\right)_{(j-1)n_1+i} \\ &= \frac{\partial h}{\partial x_{ij}} \quad \text{for } i = 1, \ldots, n_1 \text{ and } j = 1, \ldots, n_2 \ .\end{aligned}$$

For simplicity we also denote $\nabla_X h_{((j-1)n_1+i)}$ as $\nabla_X h_{(ij)}$.

Similarly the Hessian of the function $h(X)$ is given by the matrix $\nabla^2_X h \in \mathbb{R}^{n_1 n_2, n_1 n_2}$ whose elements are given by:

$$\begin{aligned}\left(\nabla^2_X h\right)_{(j-1)n_1+i,(t-1)n_1+s} &= \left(\nabla^2_{\text{vect}(X)} h\right)_{(j-1)n_1+i,(t-1)n_1+s} \\ &= \frac{\partial^2 h}{\partial x_{ij} \partial x_{st}} \quad \text{for } i, s = 1, \ldots, n_1 \text{ and } j, t = 1, \ldots, n_2\end{aligned}$$

and we denote $\nabla^2_X h_{(j-1)n_1+i,(t-1)n_1+s}$ also as $\nabla^2_X h_{(ij)(st)}$. Note that when $n_2 = 1$, we reobtain the above definitions for a scalar function of a vector.

B. Newton's method

Newton's method is an iterative technique that finds the roots of a function. For an unconstrained convex minimization problem, the roots of the gradient of the function to minimize are the minimizers of the function itself. The Newton's method is very popular due to its fast speed of convergence. Consider the following unconstrained minimization problem:

$$\text{minimize } f(\mathbf{w}), \tag{2}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is strongly convex and twice continuously differentiable. We suppose that the problem has a solution f^* and the solution is obtained at $\mathbf{w}^*$, i.e. $f^* = f(\mathbf{w}^*)$. Since f is a convex and differentiable function, a point $\mathbf{w}^*$ is optimal if and only if the gradient of the function vanishes:

$$\nabla_{\mathbf{w}} f(\mathbf{w}^*) = \mathbf{0}. \tag{3}$$

Therefore, solving the m equations of m variables in (3) is equivalent to solving the optimization problem (2). The Newton's method (also called damped Newton's method) is outlined below (see [1]):

Newton's Method Algorithm

Given

A starting point $\mathbf{w} \in \mathbf{dom} f$, a tolerance $\epsilon > 0$.

Repeat

1) Compute Newton's step and decrement:

$$\Delta \mathbf{w} := -\left(\nabla^2_{\mathbf{w}} f(\mathbf{w})\right)^{-1} \nabla_{\mathbf{w}} f(\mathbf{w}),$$

$$\delta^2 := \nabla_{\mathbf{w}} f(\mathbf{w})^T \left(\nabla^2_{\mathbf{w}} f(\mathbf{w})\right)^{-1} \nabla_{\mathbf{w}} f(\mathbf{w}).$$

2) Stopping criterion: if $\delta^2/2 \leq \epsilon$ exit.
3) Line search: use exact or backtracking line search to find t.
4) Update:

$$\mathbf{w} := \mathbf{w} + t\Delta \mathbf{w}.$$

III. The constrained norm minimization

In this paper, we deal with the following optimization problem that appears in a quite large number of applications:

$$\begin{aligned}\underset{X}{\text{minimize}} \quad & ||X||_{\sigma p} \\ \text{subject to} \quad & \phi(X) = \mathbf{y}, \\ & X \in \mathbb{R}^{n_1,n_2} \ , \ \mathbf{y} \in \mathbb{R}^c,\end{aligned} \tag{4}$$

where $||X||_{\sigma p} = \left(\sum_i \sigma_i^p\right)^{1/p}$ is the Schatten p-norm of the matrix X, and $\phi(X)$ is a linear function of the elements of X and then it can be written also as:

$$\phi(X) = A \text{ vect}(X),$$

where $A \in \mathbb{R}^{c,n_1 n_2}$ and c is the number of constraints. We suppose that the problem admits always a solution X^*.

Since we are interested in applying Newton's method to solve equation (4), the objective function should be twice differentiable. Not all the norms satisfy this property, we limit then our study to the case where p is an even integer because in this case we show that the problem (4) is equivalent to a smooth optimization problem. Let $p = 2q$, raising the objective function to the power p will not change the solution set, so we can equivalently consider the objective function:

$$h(X) = ||X||^p_{\sigma p} = Tr\left(\left(XX^T\right)^q\right).$$

Since we only have linear constraints (A vect$(X) = \mathbf{y}$), by taking only the linearly independent equations, and using Gaussian elimination to have a full row rank matrix, we can rewrite the constraints as follows:

$$\left[\ I_r \quad B\ \right] P \text{ vect}(X) = \hat{\mathbf{y}},$$

where I_r is the r-identity matrix, r is the rank of the matrix A (the number of linearly independent equations), $B \in \mathbb{R}^{r,n_1 n_2 - r}$, P is an $n_1 n_2 \times n_1 n_2$ permutation matrix of the variables, and $\hat{\mathbf{y}} \in \mathbb{R}^r$ is a vector. We arrive at the conclusion that the original problem (4) is equivalent to:

$$\begin{aligned}\underset{X}{\text{minimize}} \quad & h(X) = Tr\left(\left(XX^T\right)^q\right) \\ \text{subject to} \quad & \left[\ I_r \quad B\ \right] P \text{ vect}(X) = \hat{\mathbf{y}}.\end{aligned} \tag{5}$$

Before applying Newton's method to (5), we can further reduce the problem to an unconstrained minimization problem. By considering the equality constraints, we can form a mapping from $X \in \mathbb{R}^{n_1,n_2}$ to the vector $\mathbf{x} \in \mathbb{R}^{n_1 n_2 - r}$ as follows:

$$\mathbf{x} = \left[\begin{array}{cc} 0^{n_1 n_2 - r, r} & I_{n_1 n_2 - r} \end{array} \right] P \text{ vect}(X), \tag{6}$$

and X can be obtained from $\mathbf{x}$ and $\hat{\mathbf{y}}$ as

$$X = \text{vect}^{-1} \left(P^{-1} \left[\begin{array}{c} \hat{\mathbf{y}} - B\mathbf{x} \\ \mathbf{x} \end{array} \right] \right), \tag{7}$$

where $\text{vect}^{-1} : \mathbb{R}^{n_1 n_2} \rightarrow \mathbb{R}^{n_1,n_2}$ is the inverse function of vect(), i.e. $\text{vect}^{-1}(\text{vect}(X)) = X$. The unconstrained minimization problem is then:

$$\underset{\mathbf{x}}{\text{minimize}}\, f(\mathbf{x}), \tag{8}$$

where $f(\mathbf{x}) = Tr\left(\left(XX^T\right)^q\right)$ and X is given as in (7).

All three problems (4), (5), and (8) are convex and are equivalent to each other. We apply Newton's method to (8) to find the optimal vector $\mathbf{x}^*$ and then deduce the solution of the original problem X^*. The main difficulty in most Newton's methods is the calculation of the gradient and the Hessian. In many applications, the Hessian is not known and for this reason gradient methods are applied rather than the faster Newton's methods. However, in this paper, we show that by exploring the special structure of the function $h(X)$, we can calculate explicitly both $\nabla_{\mathbf{x}} f$ and $\nabla^2_{\mathbf{x}} f$. To this purpose, we first calculate the gradient and Hessian of $h(X)$ by the following theorem and then use the linearity of the constraints.

Theorem 1. *Let $h(X) = Tr\left(\left(XX^T\right)^q\right)$ where $X \in \mathbb{R}^{n_1,n_2}$, then the gradient of h is given by,*

$$\nabla_X h_{(ij)} = 2q \left(\left(XX^T\right)^{q-1} X \right)_{i,j} \text{ for } \begin{array}{l} i = 1, \ldots, n_1 \\ j = 1, \ldots, n_2, \end{array} \tag{9}$$

and the Hessian,

$$\begin{aligned} \nabla^2_X h_{(ij)(st)} = 2q \sum_{k=0}^{q-2} \left(\left(XX^T\right)^k X \right)_{i,t} \left(\left(XX^T\right)^{q-2-k} X \right)_{s,j} \\ + 2q \sum_{k=0}^{q-1} \left(\left(XX^T\right)^k \right)_{i,s} \left(\left(X^T X\right)^{q-1-k} \right)_{t,j}. \end{aligned} \tag{10}$$

Proof. See Appendix. □

We can now apply the chain rule to calculate the gradient and Hessian of $f(\mathbf{x})$, taking into account the mapping from $\mathbf{x}$ to X in (7).

For the gradient $\nabla_{\mathbf{x}} f$, it holds for $l = 1, \ldots, n_1 n_2 - r$:

$$(\nabla_{\mathbf{x}} f)_l = \frac{\partial f}{\partial x_l} = \sum_{i,j} \nabla_X h_{(ij)} \frac{\partial x_{ij}}{\partial x_l}, \tag{11}$$

where all the partial derivatives $\frac{\partial x_{ij}}{\partial x_l}$ are constant values because (7) is a linear transformation[1]. Applying the chain rule for the Hessian and considering directly that all the second order derivatives like $\frac{\partial^2 x_{ij}}{\partial x_l \partial x_k}$ are null (again because the mapping (7) is a linear transformation), we obtain that for $l, k = 1, \ldots, n_1 n_2 - r$:

$$\begin{aligned} \left(\nabla^2_{\mathbf{x}} f\right)_{l,k} &= \frac{\partial^2 f}{\partial x_l \partial x_k} \\ &= \sum_{i,j,s,t} \nabla^2_X h_{(ij)(st)} \frac{\partial x_{ij}}{\partial x_l} \frac{\partial x_{st}}{\partial x_k}. \end{aligned} \tag{12}$$

Since $f(\mathbf{x})$ is a convex function, then the calculated matrix $\nabla^2_{\mathbf{x}} f$ is semi-definite positive. We can add to the diagonals a small positive value γ to guarantee the existence of the inverse without affecting the convergence. The calculated Hessian is a square matrix having dimensions d by d where $d = n_1 n_2 - r$ may be large for some applications, and at every iteration of the Newton's method, we need to calculate the inverse of the Hessian. Efficient algorithms for inverting large matrices are largely discussed in the literature (see [10] for example) and are beyond the scope of this paper. Nevertheless, the given matrix has lower dimension than the typical KKT matrix[2] used in Newton's method [1]:

$$\left[\begin{array}{cc} \nabla^2_X h & A^T \\ A & 0 \end{array} \right], \tag{13}$$

where A is considered here to be a full row rank matrix, so the KKT matrix is a square matrix of dimensions d_{KKT} by d_{KKT} where $d_{KKT} = n_1 n_2 + r$. Once we know the gradient $\nabla_{\mathbf{x}} f$ and the Hessian $\nabla^2_{\mathbf{x}} f$, we just apply the Newton's method given in section II-B to find the solution $\mathbf{x}^*$ and then obtain the solution of the original problem X^*. In the next section, as a case study, we will apply the optimization technique we developed here to a graph optimization problem.

IV. A case study: Weighted Graph Optimization

In average consensus protocols, nodes in a network, each having an initial estimate (e.g. node i has the estimate $y_i(0) \in \mathbb{R}$), perform an iterative procedure where they update their estimate value by the weighted average of the estimates in their neighborhood according to the following equation:

$$y_i(k+1) = w_{ii} y_i(k) + \sum_{j \in N_i} w_{ij} y_j(k).$$

Under some general conditions on the network topology and the weights, the protocol guarantees that every estimate in the network converges asymptotically to the average of all initial estimates. The speed of convergence of average consensus protocols depends on the weights selected by nodes for their neighbors [11]. Minimizing the trace of the weighted adjacency matrix leads to weights that guarantee fast speed of convergence (see [9]). In what follows, we show that this problem is a specific case of our general problem (4) and then apply the methodology presented above to solve it using the Newton's method.

[1] Because of space constraints and for the sake of conciseness we do not write explicitly the value of these partial derivatives in the general case, but only for the specific case study we consider in the next section.

[2] Note that the sparsity of the matrix to invert is preserved by the proposed method, i.e. if the KKT matrix is sparse due to the sparsity of A and $\nabla^2_X h$, then $\nabla^2_{\mathbf{x}} f$ is also sparse.

A. Problem formulation

We consider a directed graph $G = (V, E)$ where the vertices (also called nodes) $V = \{1, \dots, n\}$ are ordered and E is the set of edges (also called links). The graph G satisfies the following symmetry condition: if there is a link between two nodes ($(ij) \in E$) then there is also the reverse link ($(ji) \in E$). We also consider the nodes to have self links, i.e. $(ii) \in E$ for every node i. Then the number of links can be written as $2m + n$ with m being a positive integer. The graph is weighted, i.e. a weight w_{ij} is associated to each link $(ij) \in E$. By considering $w_{ij} = 0$ if $(ij) \notin E$, we can group the values in a weight matrix $W \in \mathbb{R}^{n \times n}$ (i.e. $(W)_{i,j} = w_{ij}$ for $i, j = 1, \dots, n$). A graph optimization problem is to find the weights that can minimize a function $h(W)$ that depend on these weights subject to some constraints. In particular for average consensus protocols, it is meaningful [9] to consider the following problem:

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & Tr(W^p) \\ \text{subject to} \quad & W = W^T, \\ & W\mathbf{1} = \mathbf{1}, \\ & W \in \mathcal{C}_G, \end{aligned} \tag{14}$$

where $p = 2q$ is an even positive integer and $\mathcal{C}_G$ is the condition imposed by the underlying graph connectivity, i.e. $w_{ij} = 0$ if $(ij) \notin E$. We denote by $W_{(p)}$ the solution of this optimization problem. The authors in [9] show that problem (14) well approximates (the larger p, the better the approximation) the well known fastest distributed linear averaging problem [11], that guarantees the fastest asymptotic convergence rate by maximizing the spectral gap of the weight matrix. Due to the constraint that the matrix is symmetric, we can write the objective function as $h(W) = Tr\left(\left(WW^T\right)^q\right)$. Moreover, we can see that all constraints are linear equalities. Therefore, the technique derived in the previous section applies here.

B. The unconstrained minimization

We showed that the general problem (4) is equivalent to an unconstrained minimization problem (8). This is obviously true also for the more specific minimization problem (14) we are considering. It can be easily checked that in this case the number of independent constraints is equal to $r = n^2 - m$ and then the variables' vector for the unconstrained minimization has size m. We denote this vector $\mathbf{w}$. There are multiple ways to choose the m independent variables. Here we consider a variable for each pair (i, j) and (j, i) where $j \neq i$. We express that the l-th component of the weight vector $\mathbf{w}$ corresponds to the links (i, j) and (j, i) by writing $l \sim (ij)$ or $l \sim (ji)$. This choice of the independent variables corresponds to consider the *undirected* graph $G' = (V, E')$ obtained from G by removing self loops and merging links (i, j) and (j, i) and then determine a weight for each of the residual m links. Due to space constraints, we do not write the expression of B, P and $\hat{\mathbf{y}}$ that allow us to map the weight matrix W to the vector $\mathbf{w}$ so defined, but it can be easily checked that all the weights can be determined from $\mathbf{w}$ as follows: $w_{ij} = w_{ji} = w_l$ for $l \sim (ij)$ and $w_{ii} = 1 - \sum_{j \in N_i} w_{ij}$ where N_i is the set of neighbors of node i. This can be expressed in a matrix form as follows: $W = I_n - Q\text{diag}(\mathbf{w})Q^T$, where I_n is the $n \times n$ identity matrix and Q is the *incidence matrix* of graph G' (the incidence matrix of a graph having n nodes and m links is defined as the $n \times m$ matrix where for every link $l \sim (ij)$, the l-th column of Q is all zeros except for $Q_{il} = +1$ and $Q_{jl} = -1$). The equivalent unconstrained problem is then:

$$\underset{\mathbf{w}}{\text{minimize}} \; f(\mathbf{w}) = Tr((I_n - Q\text{diag}(\mathbf{w})Q^T)^p). \tag{15}$$

C. Gradient and Hessian

To apply Newton's method to minimize the function f, we have to calculate first the gradient $\nabla_{\mathbf{w}} f$ and the Hessian matrix $\nabla^2_{\mathbf{w}} f$. The function f is a composition function between $h(W) = Tr(W^p)$ and the matrix function $W = I - Q\text{diag}(\mathbf{w})Q^T$:

$$f(\mathbf{w}) = Tr(W^p)|_{W = I_n - Q\text{diag}(\mathbf{w})Q^T}.$$

From Eq. (11), we have

$$(\nabla_{\mathbf{w}} f)_l = \sum_{i,j \in V} \nabla_W h_{(ij)} \frac{\partial w_{ij}}{\partial w_l},$$

where $\nabla_W h_{(ij)} = p(W^{p-1})_{ij}$ (it follows from (9) and the fact that $W = W^T$). Due to the conditions mentioned earlier ($w_{ij} = w_{ji} = w_l$ for all $l \sim (ij)$ and $w_{ij} = 0$ if $(ij) \notin E$ and $w_{ii} = 1 - \sum_{j \in N_i} w_{ij}$), if $l \sim (ab)$ we have

$$\frac{\partial w_{ij}}{\partial w_l} = \begin{cases} +1 & \text{if } i = a \text{ and } j = b \\ +1 & \text{if } i = b \text{ and } j = a \\ -1 & \text{if } i = a \text{ and } j = a \\ -1 & \text{if } i = b \text{ and } j = b \\ 0 & \text{else.} \end{cases} \tag{16}$$

We can then calculate the gradient $\nabla_{\mathbf{w}} f \in \mathbb{R}^m$. In particular for $l \sim (ab)$ we have,

$$\begin{aligned} (\nabla_{\mathbf{w}} f)_l &= \nabla_W h_{(ab)} + \nabla_W h_{(ba)} - \nabla_W h_{(aa)} - \nabla_W h_{(bb)} = \\ &= p(W^{p-1})_{b,a} + p(W^{p-1})_{a,b} \\ &\quad - p(W^{p-1})_{a,a} - p(W^{p-1})_{b,b}. \end{aligned} \tag{17}$$

For the calculation of the Hessian, let $l \sim (ab)$, $k \sim (cd)$ be given links. Only 16 of the $n_1^2 n_2^2$ terms in Eq. (12)—those corresponding to $i, j \in \{a, b\}$ and $s, t \in \{c, d\}$—are different from zero because of (16), and they are equal to 1 or to -1. Moreover we can simplify the expression of $\nabla^2_X h_{(ij)(st)}$ in (10) by considering that $X = W = W^T$. Finally after grouping the terms, we obtain the more compact form:

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = p \sum_{z=0}^{p-2} A(z) B(z) \tag{18}$$

where

$$A(z) = \left[(W^z)_{a,c} + (W^z)_{b,d} - (W^z)_{a,d} - (W^z)_{b,c}\right],$$

$$B(z) = \left[(W^{K-z})_{a,c} + (W^{K-z})_{b,d} - (W^{K-z})_{a,d} - (W^{K-z})_{b,c}\right],$$

and $K = p - 2$.

D. Newton's direction $\Delta\mathbf{w}$

Let $\mathbf{g} \in \mathbb{R}^m$ and $H \in \mathbb{R}^{m\times m}$ such that $\mathbf{g} = \nabla_{\mathbf{w}} f(\mathbf{w})$ as in equation (17) and $H = \nabla^2_{\mathbf{w}} f(\mathbf{w})$ as in equation (18). Then the direction $\Delta\mathbf{w}$ to update the solution in Newton's method can be obtained solving the linear system $H\Delta\mathbf{w} = \mathbf{g}$.

E. Line search

As in the Newton's method described above, at each iteration the algorithm calculates a search direction ($\Delta\mathbf{w}$) and then decides how far to move along that direction (choosing a stepsize t) which results in the update equation of line 4 of the algorithm ($\mathbf{w} := \mathbf{w} + t\Delta\mathbf{w}$). The procedure of selecting the stepsize for a given direction is called line search. Most line search algorithms require $\Delta\mathbf{w}$ to be a *descent direction* (i.e. $\Delta\mathbf{w}^T\mathbf{g} < 0$ which guarantees that the function decreases along the chosen direction). Newton's direction in our problem is a descent direction because it satisfies this property:

$$\Delta\mathbf{w}^T\mathbf{g} = -\mathbf{g}^T H^{-1}\mathbf{g} < 0,$$

since H is a positive definite matrix (due to the convexity of the problem).

The Newton's method uses exact line search if at each iteration the step size is selected in order to guarantee the maximum amount of decrease of the function f in the descent direction, i.e. t is selected as the global minimizer of the univariate function $\phi(t)$:

$$\phi(t) = f(\mathbf{w} + t\Delta\mathbf{w}),\ t > 0.$$

Usually exact line search is very difficult to implement, but we benefit from the convexity of our problem to derive a procedure which gives a high precision estimate of the optimal choice of the stepsize. Notice that $\phi(t)$ can be written as follows:

$$\begin{aligned}\phi(t) = f(\mathbf{w} + t\Delta\mathbf{w}) &= Tr((I_n - Q\text{diag}(\mathbf{w} + t\Delta\mathbf{w})Q^T)^p) = \\ &= Tr((I_n - Q\text{diag}(\mathbf{w})Q^T - tQ\text{diag}(\Delta\mathbf{w})Q^T)^p) = \\ &= Tr((W + tU)^p) = h(W + tU)\end{aligned}$$

where $U = Q\text{diag}(\Delta\mathbf{w})Q^T$ and is also symmetric. Since (14) is a smooth convex optimization problem, $h()$ is smooth and convex also when it is restricted to any line that intersects its domain. Then $\phi(t) = h(W + tU)$ is convex in t and we can apply a basic Newton method to find the optimal t:

Let $t_1 = 1$ and $t_0 = 0$, select a tolerance $\eta > 0$,
while $|t_n - t_{n-1}| > \eta$
$t_n \leftarrow t_{n-1} - \frac{\phi'(t_{n-1})}{\phi''(t_{n-1})}$;
end while

At the end of this procedure, we select $t = t_n$ to be used as the stepsize of the iteration. Applying the chain rule to the composition of the function $h(Y) = Tr(Y^p)$ and $Y(t) = W + tU$ (similarly to what we have done for f in (12)), we can find the first and second derivative:

$$\phi'(t) = \sum_{i,j}\frac{\partial h}{\partial y_{ij}}u_{ij} = p\sum_i (Y^{p-1}U)_{i,i} = pTr(Y^{p-1}U),$$

$$\phi''(t) = \frac{\mathrm{d}\phi'(t)}{\mathrm{d}t} = p \times Tr\left(\sum_{q=0}^{p-2} Y^{p-2-q}UY^qU\right),$$

where $Y = W + tU$ and for the second expression we have used a result in [12].

V. The algorithm

We summarize the Newton's method used for the trace minimization problem (14):

Step 0: Choose a weight matrix $W^{(0)}$ that satisfies the conditions given in (14) (e.g. I_n is a feasible starting weight matrix). Choose a precision ϵ and set $k \leftarrow 0$.

Step 1: Calculate $\nabla_{\mathbf{w}} f^{(k)}$ from equation (17) (call this gradient $\mathbf{g}$).

Step 2: Calculate $\nabla^2_{\mathbf{w}} f^{(k)}$ from equation (18) (since f is a convex function, we have $\nabla^2_{\mathbf{w}} f^{(k)}$ is a semi-definite positive matrix, let $H = \nabla^2_{\mathbf{w}} f^{(k)} + \gamma I_m$ where γ can be chosen to be the machine precision to guarantee that H is positive definite and thus can have an inverse H^{-1}).

Step 3: Calculate Newton's direction $\Delta\mathbf{w}^{(k)} = H^{-1}\mathbf{g}$. **Stop** if $||\Delta\mathbf{w}^{(k)}|| \leq \epsilon$.

Step 4: Use Newton procedure to find the exact stepsize $t^{(k)}$

Step 5: Update the weight matrix by the following equation:

$$W^{(k+1)} = W^{(k)} + t^{(k)}Q\text{diag}(\Delta\mathbf{w}^{(k)})Q^T.$$

Step 6: Increment iteration $k \leftarrow k + 1$. Go to **Step 1**.

VI. Closed form solution for $p = 2$

Interestingly, for $p = 2$ the Newton's method converges in 1 iteration. In fact for $p = 2$, the problem (14) is the following:

$$\begin{aligned}\underset{W}{\text{minimize}} \quad & h(W) = Tr(W^2) = \sum_{i,j} w_{ij}^2 \\ \text{subject to} \quad & W = W^T, \\ & W\mathbf{1} = \mathbf{1}, \\ & W \in \mathcal{C}_G.\end{aligned} \tag{19}$$

Theorem 2. *Let $W_{(2)}$ be the solution of the optimization problem* (19)*, then we have:*

$$W_{(2)} = I_n - Q\,diag\left((I_m + \frac{1}{2}Q^TQ)^{-1}\mathbf{1}_m\right)Q^T, \tag{20}$$

where Q is the incidence matrix of the graph G.

Proof. See the appendix. □

Moreover, we show in the appendix that on D-regular graphs, the given optimization problem for $p = 2$ gives same results as other famous weight selection algorithms as metropolis weight selection (local degree) or maximum degree weight selection (for a survey on weight selection algorithms see [13] and the references therein).

T_{conv} (number of iterations)	$ER(n=100, Pr=0.07)$			
	$p=2$	$p=4$	$p=6$	$p=10$
Newton	1	5	5.7	6.1
E-GD	72.3	230.5	482.7	1500.5
E-Nesterov	130.2	422.8	811.3	1971.2
BT-GD or BT-Nesterov	> 5000	> 5000	> 5000	> 5000

TABLE I

CONVERGENCE TIME OF NEWTON'S METHOD FOR PROBLEM (14).

VII. SIMULATIONS

We apply the above optimization technique to solve problem (14) on Erdos Renyi random networks $ER(n, Pr)$, where n is the number of nodes and Pr is the probability of existence of a link. We compare the number of iterations for convergence with those of first order methods as Nesterov and Descent Gradient (DG) using either backtracking line search (referred as BT-methods in the figure) or exact line search (referred as E-methods in the figure). The Nesterov first order method as described in [14] usually achieves faster rate of convergence with respect to traditional first order methods. The Gradient Descent method follows the same steps of the Newton's algorithm, but in Step 2, the Hessian H is taken as the identity matrix (for Gradient Descent methods $H_{GD} = I_m$ while for Newton's method $H_N = \nabla^2_{\mathbf{w}} f^{(k)}$). Since at the optimal value w^* the gradient vanishes (i.e. $||\mathbf{g}^{(k)}|| = 0$), we consider the convergence time T_{conv} to be:

$$T_{conv} = \min\{k : ||\mathbf{g}^{(k)}|| < 10^{-10}\}.$$

Table 1 shows the results for the Newton's and other first order methods. The initial condition for the optimization is given by $W^{(0)} = I_n$ which is a feasible starting point. The values are averaged over 100 independent runs for each of the (n, Pr, p) values. The results show that the average convergence time of Newton's is much less than the first order methods in terms of the number of iterations. As we can see, when using exact line search, E-Nesterov was slower than E-DG method, this could be interpreted that the Descent Gradient does not suffer from the zig-zag problem usually caused by poorly conditioned convex problems. Moreover, using backtracking line search for first order methods was not converging in a reasonable number of iterations because the function we are considering is not lipschitz continuous when $p > 2$ and due to the high precision gradient stopping condition. Note that, the number of iterations is not the only factor to take into account, in fact the Newton's method requires at each iteration to invert the Hessian matrix, while GD has lower computational cost. However, the GD is very sensitive to changing the step size, while Newton's method is not. By applying constant or backtracking line search stepsizes to the GD method, the algorithm was not converging in a reasonable number of iterations while even with the simplest Newton's method (taking always a stepsize equal to 1) was converging in less than 15 iterations for the $ER(n = 100, Pr = 0.07)$ graphs.

VIII. CONCLUSION

In this paper, we showed how the Newton's method can be used for solving the constrained Schatten norm minimization. As a case study we show how to apply the methodology to graph optimization problem as consensus protocols.

APPENDIX
PROOF OF THEOREM 1

Let $h(X) = Tr\left(\left(XX^T\right)^q\right)$ where $X \in \mathbb{R}^{n_1,n_2}$, we first observe that

$$Tr\left(\left(XX^T\right)^q\right) = \sum_{u_1=1}^{n_1}\left(\left(XX^T\right)^q\right)_{u_1,u_1} =$$
$$= \sum_{u_1,u_3,\dots u_{2q-1}=1}^{n_1}\left(XX^T\right)_{u_1,u_3}\left(XX^T\right)_{u_3,u_5}\dots\left(XX^T\right)_{u_{2q-1},u_1}$$
$$= \sum_{u_1,u_3,\dots u_{2q-1}=1}^{n_1}\sum_{u_2,\dots u_{2q}=1}^{n_2} x_{u_1u_2}x_{u_3u_2}\dots x_{u_{2q-1}u_{2q}}x_{u_1u_{2q}}$$

Since for any a, b, we have $\frac{\partial x_{ab}}{\partial x_{ij}} = \delta_{ai}\delta_{bj}$, where δ_{uv} is the Kronecker delta, i.e. $\delta_{uv} = 1$ if $u = v$, $\delta_{uv} = 0$ otherwise. Then the gradient of $h(X)$ is given by:

$$\begin{aligned}
\nabla_X h_{(ij)} &= \frac{\partial Tr\left(\left(XX^T\right)^q\right)}{\partial x_{ij}} = \\
&= \sum_{u_1,u_2,\dots u_{2q}} \delta_{u_1i}\delta_{u_2j}x_{u_3u_2}\dots x_{u_{2q-1}u_{2q}}x_{u_1u_{2q}} + \\
&+ \sum_{u_1,u_2\dots u_{2q}} x_{u_1u_2}\delta_{u_3i}\delta_{u_2j}\dots x_{u_{2q-1}u_{2q}}x_{u_1u_{2q}} + \dots \\
&+ \sum_{u_1,u_2,\dots u_{2q}} x_{u_1u_2}x_{u_3u_2}\dots x_{u_{2q-1}u_{2q}}\delta_{u_1i}\delta_{u_{2q}j} = \\
&= q(X^TX\dots X^T)_{j,i} + q(XX^T\dots X)_{i,j} \\
&= 2q\left(\left(XX^T\right)^{q-1}X\right)_{i,j} \text{ for } \begin{matrix} i=1\dots n_1 \\ j=1\dots n_2 \end{matrix}. \qquad (21)
\end{aligned}$$

We can calculate similarly the Hessian of $h(X)$ for $i, s = 1\dots n_1$ and $j, t = 1\dots n_2$:

$$\begin{aligned}
\nabla^2_X h_{(ij)(st)} &= \frac{\partial^2 Tr\left(\left(XX^T\right)^q\right)}{\partial x_{ij}\partial x_{st}} = \frac{\partial}{\partial x_{st}}\left(2q\left(\left(XX^T\right)^{q-1}X\right)_{i,j}\right) \\
&= 2q\,\frac{\partial}{\partial x_{st}}\sum_{u_1,u_2,u_3,\dots,u_{2q-2}} x_{iu_1}x_{u_2u_1}x_{u_2u_3}\dots x_{u_{2q-2}j} \\
&= 2q\sum_{u_1,u_2,u_3,\dots,u_{2q-2}} \delta_{is}\delta_{u_1t}x_{u_2u_1}x_{u_2u_3}\dots x_{u_{2q-2}j} + \\
&+ 2q\sum_{u_1,u_2,u_3,\dots,u_{2q-2}} x_{iu_1}\delta_{u_2s}\delta_{u_1t}x_{u_2u_3}\dots x_{u_{2q-2}j} + \dots \\
&+ 2q\sum_{u_1,u_2,u_3,\dots,u_{2q-2}} x_{iu_1}x_{u_2u_1}x_{u_2u_3}\dots\delta_{u_{2q-2}s}\delta_{jt} \\
&= 2q\sum_{k=0}^{q-2}\left(\left(XX^T\right)^kX\right)_{i,t}\left(\left(XX^T\right)^{q-2-k}X\right)_{s,j} \\
&+ 2q\sum_{k=0}^{q-1}\left(\left(XX^T\right)^k\right)_{i,s}\left(\left(X^TX\right)^{q-1-k}\right)_{t,j}. \qquad (22)
\end{aligned}$$

APPENDIX
PROOF OF THEOREM 2

The optimization function is quadratic in the variables w_{ij}, so applying Newton's algorithm to minimize the function gives convergence in one iteration independent from the initial starting point $W^{(0)}$. Let $W^{(0)} = I_n$ which is a feasible initial starting point. The gradient $\mathbf{g}$ can be calculated according to equation (17):

$$\begin{aligned} g_l &= 2\left((I_n)_{i,j} + (I_n)_{j,i} - (I_n)_{i,i} - (I_n)_{j,j}\right) \\ &= 2(0+0-1-1) = -4 \quad \forall l = 1, \ldots, m, \end{aligned}$$

in vector form:

$$\mathbf{g} = -4 \times \mathbf{1}_m,$$

where $\mathbf{1}_m$ is a vector of all ones of dimension m.

To calculate the Hessian $\nabla^2_W f$, we apply equation (18) for $p = 2$, we get that for any two links $l \sim (ab)$ and $k \sim (cd)$, we have

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = 2 \times \left((I_n)_{a,c} + (I_n)_{b,d} - (I_n)_{a,c} - (I_n)_{b,d}\right)^2,$$

and thus

$$\left(\nabla^2_{\mathbf{w}} f\right)_{l,k} = \begin{cases} 2 \times (2)^2 & \text{if } l = k \\ 2 \times (1)^2 & \text{if } l \text{ and } k \text{ share a common vertex,} \\ 0 & \text{else.} \end{cases} \tag{23}$$

In matrix form, we can write the Hessian as follows:

$$\nabla^2_{\mathbf{w}} f = 2 \times (2I_m + Q^T Q),$$

where Q is the incidence matrix of the graph given earlier (in fact, $Q^T Q - 2I_m$ is the adjacency matrix of what is called the line graph of G). Notice that since $Q^T Q$ is semi-definite positive all the eigenvalues of the Hessian are larger than 2 and then the Hessian is invertible. The Newton's direction is calculated as follows:

$$\Delta \mathbf{w} = H^{-1}\mathbf{g} = -(I_m + \frac{1}{2}Q^T Q)^{-1}\mathbf{1}_m.$$

Thus the optimal solution for the problem for $p = 2$ is:

$$\begin{aligned} W_{(2)} &= W^{(0)} + Q\text{diag}(\Delta \mathbf{w})Q^T \\ &= I_n - Q\text{diag}\left((I_m + \frac{1}{2}Q^T Q)^{-1}\mathbf{1}_m\right)Q^T. \end{aligned}$$

APPENDIX
D-REGULAR GRAPHS

A D-regular graph is a graph where every node has the same number of neighbors which is D. Examples of D regular graphs is the cycle graphs (2-regular), the complete graph ($n-1$-regular), and many others.

On these graphs, the sum of any row in the matrix $Q^T Q$ is equals to $2D$, then $2D$ is an eigenvalue that corresponds to the eigenvector $\mathbf{1}$. Since $Q^T Q$ is a symmetric matrix, it has an eigenvalue decomposition form:

$$Q^T Q = \sum_k \lambda_k \mathbf{v}_k \mathbf{v}_k^T,$$

where $\{\mathbf{v}_k\}$ is an orthonormal set of eigenvectors (without loss of generality, let $\mathbf{v}_1 = \frac{1}{\sqrt{n}}\mathbf{1}$). Moreover, $(I_m + \frac{1}{2}Q^T Q)$ is invertible because it is positive definite and have the same eigenvectors as $Q^T Q$. Considering its inverse as a function of $Q^T Q$, we can write:

$$(I_m + \frac{1}{2}Q^T Q)^{-1} = \sum_k (1 + \frac{\lambda_k}{2})^{-1} \mathbf{v}_k \mathbf{v}_k^T.$$

Since $\mathbf{1}$ is an eigenvector of $Q^T Q$ and therefore of $(I_m + \frac{1}{2}Q^T Q)^{-1}$, so it is perpendicular to all the others ($\mathbf{v}_k^T \mathbf{1} = 0$ for all $k \neq 1$). So,

$$(I_m + \frac{1}{2}Q^T Q)^{-1}\mathbf{1} = (1 + \frac{\lambda_1}{2})^{-1}\mathbf{v}_1(\frac{n}{\sqrt{n}}) = \frac{1}{1+D}\mathbf{1}.$$

As a result the solution of the optimization in G is given by,

$$W_{(2)} = I_n - \frac{1}{1+D}QQ^T,$$

or equivalently the solution in G' is given by $\mathbf{w}$:

$$w_l = \frac{1}{1+D} \ \forall l = 1, \ldots, m.$$

Therefore, the solution of the suggested optimization problem for $p = 2$ gives the same matrix on D-regular graphs as other weight selection algorithms for average consensus.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, March 2004.

[2] E. Wei, A. Ozdaglar, A. Eryilmaz, and A. Jadbabaie, "A distributed newton method for dynamic network utility maximization with delivery contracts," in *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*, March, pp. 1–6.

[3] J. Liu and H. Sherali, "A distributed newton's method for joint multi-hop routing and flow control: Theory and algorithm," in *INFOCOM, 2012 Proceedings IEEE*, March, pp. 2489–2497.

[4] H. Attouch, P. Redont, and B. Svaiter, "Global convergence of a closed-loop regularized newton method for solving monotone inclusions in hilbert spaces," *Journal of Optimization Theory and Applications*, pp. 1–27, 2012.

[5] A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying, "A spectral regularization framework for multi-task structure learning," in *In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, Advances in Neural Information Processing Systems 20*. MIT Press, 2007.

[6] N. Srebro, J. D. M. Rennie, and T. S. Jaakola, "Maximum-margin matrix factorization," in *Advances in Neural Information Processing Systems 17*. MIT Press, 2005, pp. 1329–1336.

[7] Y. Amit, M. Fink, N. Srebro, and S. Ullman, "Uncovering shared structures in multiclass classification," in *Proceedings of the 24th international conference on Machine learning*, ser. ICML '07. New York, NY, USA: ACM, 2007, pp. 17–24.

[8] A. Argyriou, C. A. Micchelli, and M. Pontil, "On spectral learning," *J. Mach. Learn. Res.*, vol. 11, pp. 935–953, Mar. 2010.

[9] M. El Chamie, G. Neglia, and K. Avrachenkov, "Distributed Weight Selection in Consensus Protocols by Schatten Norm Minimization," INRIA, INRIA Research Report, Oct 2012.

[10] E. Isaacson and H. Keller, *Analysis of Numerical Methods*, ser. Dover Books on Mathematics Series. Dover Publ., 1994.

[11] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65 – 78, 2004.

[12] D. Bernstein, *Matrix mathematics: theory, facts, and formulas*. Princeton University Press, 2005.

[13] K. Avrachenkov, M. El Chamie, and G. Neglia, "A local average consensus algorithm for wireless sensor networks," in *IEEE DCOSS 2011 (Barcelona, Spain June 27-29)*, Jun 2011, p. 6.

[14] Y. Nesterov, *Introductory lectures on convex optimization : a basic course*, ser. Applied optimization. Boston, Dordrecht, London: Kluwer Academic Publ., 2004.