Distributed Reconstruction of Nonlinear Networks: An ADMM Approach

Wei Pan, Aivar Sootla, and Guy-Bart Stan

Centre for Synthetic Biology and Innovation and the Department of Bioengineering, Imperial College London, United Kingdom; e-mail: {w.pan11, a.sootla, g.stan}@ imperial.ac.uk

Abstract: In this paper, we present a distributed algorithm for the reconstruction of large-scale nonlinear networks. In particular, we focus on the identification from time-series data of the nonlinear functional forms and associated parameters of large-scale nonlinear networks. In previous work, a nonlinear network reconstruction problem was formulated as a nonconvex optimisation problem based on the combination of a marginal likelihood maximisation procedure with sparsity inducing priors. In this paper, we derive an iterative reweighted lasso algorithm to solve the initial nonconvex optimisation problem based on the concave-convex procedure (CCCP). Moreover, by exploiting the structure of the objective function of this algorithm, a distributed algorithm is designed. To this end, we apply the alternating direction method of multipliers (ADMM) to decompose the original problem into several subproblems. To illustrate the effectiveness of the proposed methods, we use our approach to identify a network of interconnected Kuramoto oscillators with different network sizes (500∼100,000 nodes).

1. INTRODUCTION

The importance of reconstructing nonlinear systems and associated difficulties are widely recognised. Reconstruction methods focus on specific system classes such as those described by Wiener and Volterra series or nonlinear autoregressive with exogenous inputs (NARX) models to name just a few examples. However, nonlinear systems can be described by other functional forms. One of the most important and challenging problems in nonlinear network reconstruction is nonlinear structure identification (see Ljung [1999] and references therein).

In this paper, nonlinear systems are represented in a general state-space form. In our framework, we use some a priori knowledge about the type of system we want to reconstruct, i.e., we consider a set of candidate dictionary functions appropriate for the specific type of systems from which the data have been collected (e.g., biological, chemical, mechanical, or electrical system). We assume the measured time-series data and a set of candidate dictionary functions are given. Our main objective is to identify the most parsimonious state-space representation that explains the collected time-series data at best. Generally, we cast the reconstruction problem into a sparse signal recovery problem (Candès and Tao [2005]). In Pan et al. [2013], the nonlinear network reconstruction problem was cast as a nonconvex optimisation problem which was shown to be solvable using a centralised reweighted lasso algorithm.

Social networks, communication networks and biological networks are typically very large (e.g. 100,000 nodes) and the data set collected from them is therefore quite "big". Typically, centralised reconstruction algorithms cannot handle such problems due to their associated very large memory and computational requirements. Here, we apply the alternating direction method of multipliers (ADMM) to split the centralised problem into several subproblems with each subproblem solving a weighed lasso problem independently. This approach has the advantage that memory and computational requirements can both be reduced in comparison to generic centralised solvers.

ADMM is a powerful algorithm for solving structured convex optimisation problems. The ADMM method was introduced for optimisation in the 1970's and is closely related to many other optimisation algorithms including Bregman iterative algorithms, Douglas-Rachford splitting, and proximal point methods (see Boyd et al. [2011] and references therein).

The paper is organised as follows. In Section 2, we formulate the nonlinear network reconstruction problem. In Section 3, we re-interpret this reconstruction problem from a Bayesian point of view. In Section 4, we derive an iterative reweighted ℓ_1 lasso algorithm to solve the associated nonconvex optimisation problem based on the concave-convex procedure. In Section 5, we review ADMM, apply it to our optimisation problem, and derive a distributed algorithm. In Section 6, we apply our method to the reconstruction of networks of interconnected Kuramoto oscillators. Finally, in Section 7, we conclude and discuss several future problems. Some details of the algorithm are omitted due to space limitations, the full version of the paper is available on arXiv (Pan et al. [2014]).

2. PROBLEM FORMULATION

2.1 Nonlinear Dynamical Systems

We consider dynamical systems described by multi-input multi-output (MIMO) nonlinear discrete-time equations with additive noise:

$$\mathbf{x}(t_{k+1}) = \mathbf{F}(\mathbf{x}(t_k), \mathbf{u}(t_k)) + \boldsymbol{\xi}(t_k), \tag{1}$$

where $\mathbf{x} = [x_1, \ldots, x_{n_\mathbf{x}}]^\mathrm{T} \in \mathbb{R}^{n_\mathbf{x}}$ denotes the state vector; $\mathbf{u} = [u_1, \ldots, u_{n_\mathbf{u}}]^\mathrm{T} \in \mathbb{R}^{n_\mathbf{u}}$ denotes the input vector; $\mathbf{F}(\cdot) \triangleq [\mathbf{F}_1(\cdot), \ldots, \mathbf{F}_{n_\mathbf{x}}(\cdot)]^T : \mathbb{R}^{n_\mathbf{x}+n_\mathbf{u}} \to \mathbb{R}^{n_\mathbf{x}}$, and $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_{n_\mathbf{x}}]^\mathrm{T} \in \mathbb{R}^{n_\mathbf{x}}$ is assumed to be a zero-mean Gaussian white noise vector with constant positive covariance matrix $\boldsymbol{\Xi}$, i.e., $\boldsymbol{\xi}(t_k) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Xi})$. Since this description covers most of the discrete-time nonlinear dynamical systems with infinite number of possible functional forms for $\mathbf{F}(\cdot)$, we additionally confine the scope and assume system (1) satisfies the following assumptions:

Assumption 1. System (1) is fully measurable, i.e., all the state variables x_i can be measured and there are no hidden variables.

Assumption 2. The function terms $\mathbf{F}(\mathbf{x}(t_k), \mathbf{u}(t_k))$ in (1) are smooth and can be represented as a linear combinations of several dictionary functions, (see Sec. 5.4 in Ljung [1999]).

2.2 Construction of Dictionary Functions

Depending on the field for which the dynamical model needs to be built, only a few typical nonlinearities specific to this field need to be considered. For example, the class of models that arise from genetic regulatory networks (GRN) typically involves nonlinearities that capture fundamental biochemical kinetic laws, which are confined to either polynomial or rational functions. In what follows we gather the set of *all* candidate/possible dictionary functions that we want to consider for reconstruction. Consider state variable x_i, $i = 1, \ldots, n_\mathbf{x}$. Under Assumption 2, the function terms for state i can be written as: $\mathbf{F}_i(\mathbf{x}(t_k), \mathbf{u}(t_k)) = \sum_{s=1}^{N_i} w_{is}\mathbf{f}_{is}(\mathbf{x}(t_k), \mathbf{u}(t_k)) = \mathbf{w}_i^\mathrm{T}\mathbf{f}_i(\mathbf{x}(t_k), \mathbf{u}(t_k))$, where $\mathbf{w} \in \mathbb{R}^{N_i}$ and $\mathbf{f}_i : \mathbb{R}^{n_\mathbf{x}+n_\mathbf{u}} \to \mathbb{R}^{N_i}$ are dictionary functions that are assumed to govern the dynamics. $\mathbf{f}_i(\mathbf{x}(t_k), \mathbf{u}(t_k))$ can be monomial, polynomial, constant or any other functional form such as rational, exponential, trigonometric etc.

Under the assumptions above, the dynamics of state i can be described by:

$$x_i(t_{k+1}) = \mathbf{f}_i^T(\mathbf{x}(t_k), \mathbf{u}(t_k))\mathbf{w}_i + \xi_i(t_k),\ i = 1, \ldots, n_\mathbf{x}, \tag{2}$$

where $\xi_i(t_k)$ is assumed to be i.i.d. Gaussian distributed: $\xi_i(t_k) \sim \mathcal{N}(0, \lambda_i)$. If $M+1$ data samples satisfying (2) can be obtained from the system of interest, the system in (2) can be written as

$$\mathbf{y}_i = \mathbf{A}_i\mathbf{w}_i + \boldsymbol{\xi}_i,\ i = 1, \ldots, n_\mathbf{x}. \tag{3}$$

with

$$\begin{aligned}
&\mathbf{y}_i \triangleq [x_i(t_1), \ldots, x_i(t_M)]^\mathrm{T} \in \mathbb{R}^M,\\
&\mathbf{A}_i \triangleq \begin{bmatrix} f_{i1}(\mathbf{x}(t_0), \mathbf{u}(t_0)) & \ldots & f_{iN_i}(\mathbf{x}(t_0), \mathbf{u}(t_0)) \\ \vdots & \vdots & \vdots \\ f_{i1}(\mathbf{x}(t_{M-1}), \mathbf{u}(t_{M-1})) & \ldots & f_{iN_i}(\mathbf{x}(t_{M-1}), \mathbf{u}(t_{M-1})) \end{bmatrix}\\
&\quad \in \mathbb{R}^{M \times N_i},\\
&\mathbf{w}_i \triangleq \begin{bmatrix} w_{i1}, \ldots, w_{iN_i} \end{bmatrix}^T \in \mathbb{R}^{N_i},\\
&\boldsymbol{\xi}_i \triangleq [\xi_i(t_0), \ldots, \xi_i(t_{M-1})]^\mathrm{T} \in \mathbb{R}^M.
\end{aligned} \tag{4}$$

Since the $n_\mathbf{x}$ linear regression problems in (3) are independent, for simplicity of notation, we omit the subscript i in (3) and write

$$\mathbf{y} = \mathbf{A}\mathbf{w} + \boldsymbol{\xi}. \tag{5}$$

The problem is thus to find $\mathbf{w}$ given the measured noisy data stored in $\mathbf{y}$.

2.3 Discussion on relaxations of solutions

Considering the network reconstruction problem in practice, several problems arise with respect to (5). Firstly, a low number of time-series measurements will render the linear regression in (5) under-determined. Secondly, the number of columns of the dictionary matrix might be very large, due to the potential introduction of non-relevant and/or non-independent dictionary functions in $\mathbf{A}$. However, finding the sparsest solution is NP-hard. Classically, a lasso algorithm is typically used as a relaxation to this NP-hard optimisation problem (Tibshirani [1996]). Lasso usually works well when the dictionary matrix has certain properties such as the *restricted isometry property* (RIP), (Candès and Tao [2005]) or the incoherence property, (Donoho and Elad [2003]). These properties basically state that two or more of the columns of dictionary matrix cannot be co-linear or close to co-linear. Unfortunately, such properties are hardly guaranteed in typical network reconstruction problems. In order to relax the RIP or incoherence requirements, we employ a probabilistic viewpoint, that is we treat the problem as a Bayesian inference one as prescribed in Tipping [2001].

3. BAYESIAN VIEWPOINT

Bayesian modelling treats all unknowns as stochastic variables with certain probability distributions (Bishop [2006]). For $\mathbf{y} = \mathbf{A}\mathbf{w} + \boldsymbol{\xi}$, it is assumed that the stochastic variables in $\boldsymbol{\xi}$ are i.i.d. Gaussian distributed with $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \lambda\mathbf{I})$. In this case, the likelihood of the data given $\mathbf{w}$ is

$$\mathcal{P}(\mathbf{y}|\mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{w}, \lambda\mathbf{I}) \propto \exp\left[-\frac{1}{2\lambda}\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2\right].$$

However, what we really need to estimate is $\mathcal{P}(\mathbf{w}|\mathbf{y})$. In order to do so, we first specify the prior distribution as $\mathcal{P}(\mathbf{w}) = \prod_j \mathcal{P}(w_j)$. To enforce sparsity on $\mathbf{w}$, we consider *super-Gaussian* priors, which yield a lower bound for the priors $\mathcal{P}(w_j)$ (Palmer et al. [2005]). More specifically, if we define $\boldsymbol{\gamma} \triangleq [\gamma_1, \ldots, \gamma_N]^T \in \mathbb{R}_+^N$, we can represent the prior in the following relaxed (variational) form:

$$\mathcal{P}(\mathbf{w}) = \prod_{j=1}^{n} \mathcal{P}(w_j),\ \mathcal{P}(w_j) = \max_{\gamma_j > 0} \mathcal{N}(w_j|0, \gamma_j)\varphi(\gamma_j),$$

where $\varphi(\gamma_j)$ is a nonnegative function which is treated as a hyperprior with γ_j being its associated hyperparameters. Throughout, we call $\varphi(\gamma_j)$ the "*potential function*". This Gaussian relaxation is possible if and only if $\log \mathcal{P}(\sqrt{w_j})$ is concave on $(0, \infty)$.

Note that, if the parameters γ are known, we can estimate $\mathcal{P}(\mathbf{w}|\mathbf{y}, \gamma)$ instead of computing $\mathcal{P}(\mathbf{w}|\mathbf{y})$. Therefore, the problem should be recast in terms of finding the most appropriate hyperparameters of the priors $\hat{\boldsymbol{\gamma}} = [\hat{\gamma}_1, \ldots, \hat{\gamma}_N]$.

A good way of selecting $\hat{\boldsymbol{\gamma}}$ is to choose it as the minimiser of the sum of the misaligned probability mass, e.g.,

$$\begin{aligned}\hat{\boldsymbol{\gamma}} &= \underset{\boldsymbol{\gamma}\geq\mathbf{0}}{\operatorname{argmin}} \int \mathcal{P}(\mathbf{y}|\mathbf{w})\,|\mathcal{P}(\mathbf{w}) - \mathcal{P}(\mathbf{w};\boldsymbol{\gamma})|\,d\mathbf{w} \\ &= \arg\max_{\boldsymbol{\gamma}\geq\mathbf{0}} \int \mathcal{P}(\mathbf{y}|\mathbf{w}) \prod_{j=1}^{n} \mathcal{N}(w_j|0,\gamma_j)\varphi(\gamma_j)d\mathbf{w}.\end{aligned} \tag{6}$$

The procedure in (6) is referred to as evidence/marginal likelihood maximisation or type-II maximum likelihood, (Tipping [2001], Wipf et al. [2011]). It means that the marginal likelihood can be maximised by selecting the most probable hyperparameters able to explain the observed data. To get an estimate to $\boldsymbol{\gamma}$ and $\mathbf{w}$, we can formulate the following optimisation problem

$$\min_{\boldsymbol{\gamma}\geq\mathbf{0},\mathbf{w}} \|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda \mathbf{w}^T\boldsymbol{\Gamma}^{-1}\mathbf{w} + \log|\lambda\mathbf{I} + \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathrm{T}}|. \tag{7}$$

More details and derivations can be found in, e.g. Wipf et al. [2011], Pan et al. [2013]. It should be noted that such formulation allows to incorporate convex constraints on $\mathbf{w}$, which are typically very useful in network reconstruction problems. To ease notation and avoid interrupting the logical flow of this paper, we will derive the algorithm without considering convex constraints in what follows.

4. CONCAVE-CONVEX PROCEDURE

The cost function in (7) is convex in $\mathbf{w}$ but nonconvex in $\boldsymbol{\Gamma}$. In this section, we show how this nonconvex optimisation problem can be solved by a concave-convex procedure (CCCP). CCCP is another interpretation of the iterative reweighted lasso algorithm (Wipf and Nagarajan [2010]). Let

$$\begin{aligned}u(\mathbf{w},\boldsymbol{\gamma}) &\triangleq \|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda\sum_j \frac{w_j^2}{\gamma_j}, \\ v(\boldsymbol{\gamma}) &\triangleq -\log|\lambda\mathbf{I} + \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathrm{T}}|.\end{aligned} \tag{8}$$

Note that $u(\mathbf{w},\boldsymbol{\gamma})$ is jointly convex in $\mathbf{w}$ and $\boldsymbol{\gamma}$, and $v(\boldsymbol{\gamma})$ is convex in $\boldsymbol{\gamma}$. The minimisation of the cost function (7) can thus be formulated as a concave-convex procedure

$$\min_{\boldsymbol{\gamma}\geq\mathbf{0},\mathbf{w}} \left(u(\mathbf{w},\boldsymbol{\gamma}) - v(\boldsymbol{\gamma})\right) \tag{9}$$

Since $v(\boldsymbol{\gamma})$ is differentiable over $\boldsymbol{\gamma}$, the problem in (9) can be transformed into the following iterative convex optimisation problem

$$\left[\mathbf{w}^{k+1},\boldsymbol{\gamma}^{k+1}\right] = \underset{\boldsymbol{\gamma}\geq\mathbf{0},\mathbf{w}}{\operatorname{argmin}} \left(u(\mathbf{w},\boldsymbol{\gamma}) - \nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k)^T\boldsymbol{\gamma}\right). \tag{10}$$

Using basic principles in convex analysis, we then obtain the following analytic form for the negative gradient of $v(\boldsymbol{\gamma})$ at $\boldsymbol{\gamma}$:

$$\begin{aligned}\boldsymbol{\alpha}^k &= -\nabla_{\boldsymbol{\gamma}} v(\boldsymbol{\gamma}^k)^T \\ &= -\nabla_{\boldsymbol{\gamma}}\left(-\log|\lambda\mathbf{I} + \mathbf{A}\boldsymbol{\Gamma}\mathbf{A}^{\mathrm{T}}|\right)|_{\boldsymbol{\gamma}=\boldsymbol{\gamma}^k} \\ &= \operatorname{diag}\left[\mathbf{A}^{\mathrm{T}}\left(\lambda\mathbf{I} + \mathbf{A}\boldsymbol{\Gamma}^{\mathbf{k}}\mathbf{A}^{\mathrm{T}}\right)^{-1}\mathbf{A}\right].\end{aligned} \tag{11}$$

The iterative procedure (10) can then be formulated as

$$\begin{aligned}&\left[\mathbf{w}^{k+1},\boldsymbol{\gamma}^{k+1}\right] = \\ &\underset{\boldsymbol{\gamma}\geq\mathbf{0},\mathbf{w}}{\operatorname{argmin}} \left(\|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda\sum_j\left(\frac{w_j^2}{\gamma_j} + \alpha_j^k\gamma_j\right)\right).\end{aligned} \tag{12}$$

The objective function in (12) is jointly convex in $\mathbf{w}$ and $\boldsymbol{\gamma}$ and can be globally minimised by solving over $\boldsymbol{\gamma}$ and then $\mathbf{w}$. If $\mathbf{w}$ is fixed, it gives

$$\boldsymbol{\gamma}^{k+1} = \underset{\boldsymbol{\gamma}\geq\mathbf{0}}{\operatorname{argmin}} \left(\|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda\sum_j\left(\frac{w_j^2}{\gamma_j} + \alpha_j^k\gamma_j\right)\right). \tag{13}$$

We notice that in (13), $\boldsymbol{\gamma}^{k+1}$ has the closed form solution $\gamma_j^{k+1} = |w_j|/\sqrt{\alpha_j^k}$. If $\gamma_j^{k+1} = |w_j|/\sqrt{\alpha_j^k}$ is substituted into (13), we get

$$\begin{aligned}\mathbf{w}^{k+1} &= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda\sum_j\left(\frac{w_j^2}{\gamma_j^{k+1}} + \alpha_j^k\gamma_j^{k+1}\right)\right) \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \left(\|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + 2\lambda\sum_{j=1}^{N}\sqrt{\alpha_j^k}|w_j|\right).\end{aligned} \tag{14}$$

We can then set $\gamma_j^{k+1} = |w_j^{k+1}|/\sqrt{\alpha_j^k}$, $\forall j$ and update $\boldsymbol{\alpha}^{k+1}$ by (11). However, some of the estimated weights will be several orders of magnitude lower than the average "energy", e.g., $w_j^2 \ll \|\mathbf{w}\|_2^2$. Thus a threshold can be defined *a priori* to prune "small" weights at each iteration.

We can now explain how the update of the parameters can be obtained based on the above: Set the iteration count k to zero and $\theta_j^0 = 1$, $\forall j$. At this stage, the optimisation is a typical lasso algorithm. Then at the k^{th} iteration, let $\theta_j^{(k)} = \sqrt{\alpha_j^k}$, $\forall j$. The above described procedure is summarised in Algorithm 1.

In the next section, we will reformulate the centralised optimisation in Algorithm 1 into a distributed optimisation problem using ADMM.

Algorithm 1 Reweighted lasso on $\mathbf{w}$

1: Initialise $\theta_j^0 = 1$, $\forall j$
2: **for** $k = 0,\ldots,k_{\max}$ **do**
3: Solve the weighted lasso problem

$$\mathbf{w}^{k+1} = \underset{\mathbf{w}}{\operatorname{argmin}}\left(\frac{1}{2}\|\mathbf{A}\mathbf{w}-\mathbf{y}\|_2^2 + \lambda\|\boldsymbol{\Theta}^k\mathbf{w}\|_1\right); \tag{15}$$

4: Set $\boldsymbol{\Theta}^{(k)} \triangleq \operatorname{diag}\left[\boldsymbol{\theta}^{(k)}\right]$, $\mathbf{W}^{(k)} \triangleq \operatorname{diag}\left[|\mathbf{w}^{(k)}|\right]$ and $\theta_j^{k+1} = \left[(\mathbf{A}_j)^{\mathrm{T}}\left(\lambda\mathbf{I} + \mathbf{A}(\boldsymbol{\Theta}^k)^{-1}\mathbf{W}^{k+1}(\mathbf{A})^T\right)^{-1}\mathbf{A}_j\right]^{-\frac{1}{2}}$
5: **if** a stopping criterion is satisfied **then**
6: Break;
7: **end if**
8: **end for**

5. ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

ADMM is a numerical algorithm for solving optimisation problems such as

$$\begin{aligned}\min_{\mathbf{w}} \quad & f(\mathbf{w}) + g(\mathbf{z}), \\ \text{subject to} \quad & P\mathbf{w} + Q\mathbf{z} = \mathbf{c},\end{aligned} \tag{16}$$

with variable $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, where $P \in \mathbb{R}^{p\times n}$, $Q \in \mathbb{R}^{p\times m}$, and $\mathbf{c} \in \mathbb{R}^p$ (Boyd et al. [2011]). We will assume that $f(\cdot)$ and $g(\cdot)$ are convex. As for the method of multipliers, we form the augmented Lagrangian

$$\begin{aligned}L_\rho =& f(\mathbf{w}) + g(\mathbf{z}) + \mathbf{v}^T(P\mathbf{w} + Q\mathbf{z} - c) \\ &+ \rho/2\|P\mathbf{w} + Q\mathbf{z} - c\|_2^2.\end{aligned} \tag{17}$$

Defining the residual $r = P\mathbf{w} + Q\mathbf{z} - c$ and $\mathbf{u} = (1/\rho)\mathbf{v}$ as the scaled dual variable, we can express the ADMM problem as

$$\begin{aligned}\mathbf{w}^{k+1} &:= \operatorname*{argmin}_{\mathbf{w}} \left(f(\mathbf{w}) + \frac{\rho}{2}\|P\mathbf{w} + Q\mathbf{z}^k - \mathbf{c} + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{z}^{k+1} &:= \operatorname*{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho}{2}\|P\mathbf{w}^{k+1} + Q\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{u}^{k+1} &:= \mathbf{u}^k + P\mathbf{w}^{k+1} + Q\mathbf{z}^{k+1} - \mathbf{c}. \end{aligned} \tag{18}$$

More details on stopping criteria, such as the absolute tolerance ϵ_{abs} and the relative tolerance ϵ_{rel} can be found in (Boyd et al. [2011]).

In our setting, the number of candidate functions will be very large. Therefore we partition across the candidate functions. Each subsystem can deal with its split of candidate functions independently then update the shared variables. The following are direct consequences of the so-called sharing problem in (Boyd et al. [2011]).

We partition the parameter vector $\mathbf{w}$ as $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_n)$, with $\mathbf{w}_i \in \mathbb{R}^{N_i}$, where $\sum_{i=1}^n N_i = N$. We similarly partition the dictionary matrix $\mathbf{A}$ as $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_n]$, with $\mathbf{A}_i \in \mathbb{R}^{M \times N_i}$. Thus $\mathbf{A}\mathbf{w} = \sum_{i=1}^n \mathbf{A}_i\mathbf{w}_i$, i.e., $\mathbf{A}_i\mathbf{w}_i$ can be thought of as a 'partial' prediction of $\mathbf{y}$ using only the candidate functions referenced in $\mathbf{w}_i$. Then the reweighted lasso problem (15) $\min_{\mathbf{w}} \left(\frac{1}{2}\|\mathbf{A}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{\Theta}\mathbf{w}\|_1\right)$ becomes $\min_{\mathbf{w}} \left(\frac{1}{2}\|\sum_{i=1}^N \mathbf{A}_i\mathbf{w}_i - \mathbf{y}\|_2^2 + \lambda \sum_{i=1}^N \|\mathbf{\Theta}_i\mathbf{w}_i\|_1\right)$.

Following the approach used for the sharing problem, we express the problem as

$$\begin{aligned} \min \quad & \left(\frac{1}{2}\|\sum_{i=1}^N \mathbf{z}_i - \mathbf{y}\|_2^2 + \lambda \sum_{i=1}^N \|\mathbf{\Theta}_i\mathbf{w}_i\|_1 \right), \\ \text{subject to} \quad & \mathbf{A}_i\mathbf{w}_i - \mathbf{z}_i = 0,\ i = 1, \ldots, N, \end{aligned}$$

with new variables $\mathbf{z} \in \mathbb{R}^m$. The derivation and simplification of ADMM also follows that for the sharing problem. The scaled form of ADMM is

$$\begin{aligned}\mathbf{w}_i^{k+1} &:= \operatorname*{argmin}_{\mathbf{w}_i} \left(\frac{\rho}{2}\|\mathbf{A}_i\mathbf{w}_i - \mathbf{z}^k + \mathbf{u}^k\|_2^2 + \lambda\|\mathbf{\Theta}_i\mathbf{w}_i\|_1 \right) \\ \mathbf{z}^{k+1} &:= \operatorname*{argmin}_{\mathbf{z}} \left(\frac{1}{2}\|\sum_{i=1}^N \mathbf{z}_i - \mathbf{y}\|_2^2 + \sum_{i=1}^N \frac{\rho}{2}\|\mathbf{A}\mathbf{w}^{k+1} - \mathbf{z}^k + \mathbf{u}^k\|_2^2 \right) \\ \mathbf{u}^{k+1} &:= \mathbf{u}^k + \mathbf{A}\mathbf{w}^{k+1} - \mathbf{z}^{k+1}. \end{aligned} \tag{19}$$

As in the discussion for the sharing problem, we carry out the $\mathbf{z}$-update by first solving for the average

$$\begin{aligned}\overline{\mathbf{z}}^{k+1} &:= \operatorname*{argmin}_{\overline{\mathbf{z}}} \left(\|N\overline{\mathbf{z}} - \mathbf{y}\|_2^2 + \sum_{i=1}^N \frac{N\rho}{2}\|\overline{\mathbf{z}} - \overline{\mathbf{A}\mathbf{w}}^{k+1} - \overline{\mathbf{u}}^k\|_2^2 \right) \\ \mathbf{z}^{k+1} &:= \overline{\mathbf{z}}^{k+1} + \mathbf{A}_i\mathbf{w}_i^{k+1} + \mathbf{u}^k - \overline{\mathbf{A}\mathbf{w}}^{k+1} - \overline{\mathbf{u}}^k, \end{aligned} \tag{20}$$

where $\overline{\mathbf{A}\mathbf{w}}^{k+1} = (1/N)\sum_{i=1}^N \mathbf{A}_i\mathbf{w}_i^{k+1}$. Substituting the last expression into the update for $\mathbf{u}_i$, we find that

$$\mathbf{u}_i^{k+1} = \overline{\mathbf{A}\mathbf{w}}^{k+1} + \overline{\mathbf{u}}^k - \overline{\mathbf{z}}^{k+1},$$

which shows that, as in the sharing problem, all the dual variables are equal. Using a single dual variable $\mathbf{u}^k \in \mathbb{R}^m$, eliminating $\mathbf{z}_i$, and defining

$$\mathbf{b} = \mathbf{A}_i\mathbf{w}_i^k + \overline{\mathbf{z}}^k - \overline{\mathbf{A}\mathbf{w}}^k - \mathbf{u}^k,$$

we arrive at Algorithm 2.

Algorithm 2 ADMM for splitting across candidate functions

1: **for** $k = 0, \ldots, k_{\max}$ **do**
2: Solve
$$\mathbf{w}_i^{k+1} = \operatorname*{argmin}_{\mathbf{w}_i} \left(\frac{\rho}{2}\|\mathbf{A}_i\mathbf{w}_i - \mathbf{b}\|_2^2 + \lambda\|\mathbf{\Theta}_i\mathbf{w}_i\|_1 \right);$$
3: Compute $\overline{\mathbf{z}}^{k+1} = \frac{1}{N+\rho}\left(\mathbf{y} + \rho\overline{\mathbf{A}\mathbf{w}}^{k+1} + \rho\mathbf{u}^k\right)$;
4: Update $\mathbf{u}^{k+1} = \mathbf{u}^k + \overline{\mathbf{A}\mathbf{w}}^{k+1} - \overline{\mathbf{z}}^{k+1}$;
5: **if** a stopping criterion is satisfied **then**
6: Break;
7: **end if**
8: **end for**

Each $\mathbf{w}_i$-update is a lasso problem with n_i variables, which can be solved using any weighted lasso method. In the $\mathbf{w}_i$-update, we have $\mathbf{w}_i^{k+1} = \mathbf{0}$ (meaning that none of the features in the i-th block are used) if and only if $\|\mathbf{A}_i^T\mathbf{b}\|_2 \le \frac{\lambda}{\rho}$.

The weighted lasso problem can be solved using ADMM as well, as we now explain briefly:

$$\begin{aligned} \min \quad & \frac{1}{2}\|\mathbf{A}_i\mathbf{w}_i - \mathbf{b}\|_2^2 + \frac{\lambda}{\rho}\|\mathbf{\Theta}_i\mathbf{w}_i\|_1, \\ \text{subject to} \quad & \mathbf{\Theta}_i\mathbf{w}_i - \hat{\mathbf{z}}_i = 0,\ i = 1, \ldots, N. \end{aligned}$$

Defining $\hat{\lambda} = \lambda/\rho$ yields the following ADMM algorithm (Algorithm 3):

Algorithm 3 ADMM for weighted lasso

1: **for** $k = 0, \ldots, k_{\max}$ **do**
2: Update $\mathbf{w}_i^{k+1} = (\mathbf{A}_i^T\mathbf{A}_i + \hat{\rho}\mathbf{\Theta}_i^T\mathbf{\Theta}_i)^{-1}(\mathbf{A}_i^T\mathbf{b} + \hat{\rho}\mathbf{\Theta}_i^T(\hat{\mathbf{z}}_i - \hat{\mathbf{u}}_i))$;
3: Update $\hat{\mathbf{z}}_i^{k+1} = S_{\hat{\lambda}/\hat{\rho}}(\mathbf{\Theta}_i\mathbf{w}_i^{k+1} + \hat{\mathbf{u}}_i^k)$;
4: Update $\hat{\mathbf{u}}_i^{k+1} = \hat{\mathbf{u}}_i^k + \mathbf{\Theta}_i\mathbf{w}_i^{k+1} - \hat{\mathbf{z}}_i^{k+1}$;
5: **if** a stopping criterion is satisfied or when k reaches a predefined iteration number $k_{\max}$ **then**
6: Break;
7: **end if**
8: **end for**

where $\hat{\rho}$ is the penalty parameter and the soft thresholding operator $S_{\hat{\lambda}/\hat{\rho}}$ is defined as

$$S_{\hat{\lambda}/\hat{\rho}}(x) = \max(0, x - \hat{\lambda}/\hat{\rho}) - \max(0, -x - \hat{\lambda}/\hat{\rho}).$$

We are now ready to summarise the procedure for nonlinear network reconstruction in Algorithm 4.

6. AN EXAMPLE OF RECONSTRUCTION OF A NETWORK OF KURAMOTO OSCILLATORS

A classical example in physics, engineering and biology is the Kuramoto oscillator network (Strogatz [2000]). We consider a network where the Kuramoto oscillators are nonidentical (each has its own natural oscillation frequency ω_i) and the coupling strengths between nodes are not the same. The corresponding discrete-time dynamics can be described by $\phi_i(t_{k+1}) = \phi_i(t_k) + (t_{k+1} - t_k)\left[\omega_i + \sum_{j=1}^n w_{ij}g_{ij}(\phi_j(t_k) - \phi_i(t_k)) + \xi_i(t_k)\right]$, where $i = 1, \ldots, n$, $\phi_i \in [0, 2\pi)$ is the phase of oscillator i, ω_i is

Algorithm 4 ADMM on $\mathbf{w}$

1: Initialisation
   (1) Collect the time-series data, specify the candidate functions and construct the dictionary matrix;
   (2) Partition the dictionary matrix $\mathbf{A}$ as $\mathbf{A} = [\mathbf{A}_1, \ldots, \mathbf{A}_P]$, with $\mathbf{A}_i \in \mathbb{R}^{M \times P_i}$;
   (3) Initialise the weight $\mathbf{\Theta}^0$ as $\mathbf{\Theta}^0 = [\mathbf{\Theta}_1^0, \ldots, \mathbf{\Theta}_P^0]$, $\mathbf{\Theta}_i^0 = [\theta_{i1}^0, \ldots, \theta_{iP_i}^0]$, with $\theta_{ij}^0 = 0$.
2: **for** $k = 0, \ldots, k_{\max}$ **do**
3: Apply Algorithm 2 and 3 to to get an estimate on $\mathbf{w}^{k+1}$ according to (15);
4: Set $\mathbf{\Theta}^{(k)} \triangleq \operatorname{diag}\left[\boldsymbol{\theta}^k\right]$, $\mathbf{W}^k \triangleq \operatorname{diag}\left[|\mathbf{w}^k|\right]$;
5: Update θ_j^{k+1} for the next iteration using:

$$\theta_j^{(k+1)} = \left[\mathbf{A}_j^{\mathrm{T}}\left(\lambda \mathbf{I} + \mathbf{A}(\mathbf{\Theta}^k)^{-1}\mathbf{W}^{k+1}\mathbf{A}^T\right)^{-1}\mathbf{A}_j)\right]^{-\frac{1}{2}};$$

6: **if** a stopping criterion is satisfied **then**
7: Break;
8: **end if**
9: **end for**

its natural frequency, and the coupling function g_{ij} is usually taken as $\sin(\cdot)$ for all i, j. w_{ij} represent the coupling strength between oscillators i and j, and thus $[w_{ij}]_{n\times n}$ defines the topology of the oscillator network. Here, assuming we don't know the exact form of g_{ij}, we reconstruct from time-series data of the individual phases ϕ_i a dynamical network consisting of n Kuramoto oscillators, i.e., we identify the coupling functions $g_{ij}(\cdot)$ as well as the model parameters, i.e., ω_i and w_{ij}, $i, j = 1, \ldots, n$.

To define the dictionary matrix $\mathbf{A}$, we assume that all the dictionary functions are functions of a pair of state variables only and consider 2 candidate coupling functions g_{ij}: $\sin(x_j - x_i)$ and $\cos(x_j - x_i)$. Based on this, the dictionary matrix terms are all be of the form $[\sin(x_j(t_k) - x_i(t_k)), \cos(x_j(t_k) - x_i(t_k))] \in \mathbb{R}^2$. To also take into account the natural frequencies, we add to the last column of the dictionary matrix a unit vector. Following the procedure in (4), this leads to a dictionary matrix $\mathbf{A}_i \in \mathbb{R}^{M\times(2n+1)}$. The output is then defined as $\mathbf{y}_i \triangleq \left[\frac{\phi_i(t_1)-\phi_i(t_0)}{t_1-t_0}, \ldots, \frac{\phi_i(t_M)-\phi_i(t_{M-1})}{t_M-t_{M-1}}\right]^{\mathrm{T}} \in \mathbb{R}^M$.

To generate the time-series data, we simulated a Kuramoto oscillator network for which 10% of the non-diagonal entries of the weight matrix $[w_{ij}]_{n\times n}$ are nonzero (assuming g_{ii} and w_{ii} are zeros), and the non-zero w_{ij} values are drawn from a standard uniform distribution on the interval $[-10, 10]$. The natural frequencies ω_i are drawn from a normal distribution with mean 0 and variance 10. In order to create simulated data, we simulated the discrete-time model and took 'measurements data points' every $t_{k+1} - t_k = 0.1$ between $t = 0$ and $t = 100$ (in arbitrary units) from random initial conditions which are drawn from a standard uniform distribution on the open interval $(0, 2\pi)$. Thus a total of 1001 measurements for each oscillator phase ϕ_i were collected (including the initial value). Once again, it should be noted that the the number of rows of the dictionary matrix is less than that of columns.

We compared the centralised algorithm using CVX (Grant et al. [2008]), the centralised algorithm using ADMM, and the distributed algorithm using ADMM. We fixed the number of measurements M to be 1001 and varied the network size n between 500 and 100,000. For the distributed algorithm, we split the problem into 1000 subproblems where each one had the same dimension. The algorithm was implemented in MATLAB R2012b. The calculations were performed on a HP workstation with two 8 core Intel® Xeon(R) CPU E5-2650 2.00GHz and 64 GB RAM.

Since the reconstruction problem in (3) for each node is independent, we therefore consider the performance of a single node for illustration. We first investigate the performance for different signal-to-noise ratios of the data generated for this example. For this, we define the signal-to-noise ratio (SNR) by $\mathrm{SNR(dB)} \triangleq 20\log(\|\mathbf{A}\mathbf{w}_{\mathrm{true}}\|_2/\|\boldsymbol{\xi}\|_2)$. We considered SNR ranging from 5 dB to 25 dB for each generated weight. To characterise the accuracy of a reconstruction, we use the normalised mean square error (NMSE) as a performance index, defined by $\|\hat{\mathbf{w}}-\mathbf{w}\|/\|\mathbf{w}\|$, where $\hat{\mathbf{w}}$ is the estimate of the true weight $\mathbf{w}$. For each SNR, we generated 50 independent experiments (with different initial conditions and parameters) and calculated the average NMSE for each SNR over these 10 experiments. The results are shown in Fig. 1.

For all the experiments, we set the penalty parameter $\rho = 1$ in the augmented Lagrangian and the scalar regularisation parameter $\lambda = 0.05\|\mathbf{A}^T\mathbf{y}\|_\infty$. We also considered termination tolerances $\epsilon_{abs} = 10^{-4}$ and $\epsilon_{rel} = 10^{-2}$ and set the ADMM iteration number to be 200 and the reweighted iteration number to be 10 throughout all algorithms.

We compared the computation time for different network sizes for each method. These methods were tested with the different SNRs considered in Figure 1 . In the implementation of the distributed algorithm, we used the Matlab command `parfor` to parallelise the $\mathbf{w}_k$-update in Algorithm 2. We used the Matlab command `matlabpool('size')` to start a worker pool. The `size` varied from 2 to 10. For each method, we found that the computation time over each SNR varied slightly (at least within the same magnitude). We calculated the average computation time from a total 250 (=5×50) independent experiments for each method. The computation times in seconds are shown in Table 1. For larger problem, with network size larger than 50,000, CVX-based reweighted lasso runs into memory difficulties. Thus there are no results reported in the table for these network sizes. For the distributed reweighted lasso algorithm, the computation time decreases when the problem is split between an increasing number of processors. It should be noted that the computation time for distributed reweighted lasso is small partially because Matlab performs some matrix computations in parallel. On the other hand, CVX exploits only one core.

7. CONCLUSION AND DISCUSSION

In this paper, a new distributed reconstruction method for nonlinear dynamical network is presented. The proposed method only requires time-series data and some prior knowledge about the class of systems for which a dynamical model needs to be built. The network reconstruction problem is cast as a sparse linear regression problem. Using a Bayesian interpretation, this problem is solved using a reweighed ℓ_1 algorithm, which can further

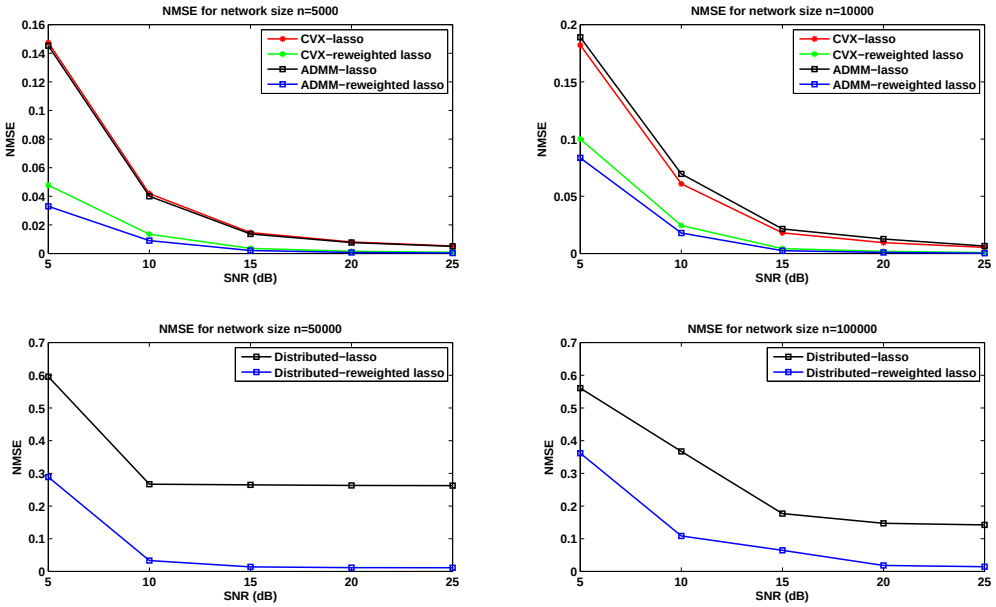

Fig. 1. Normalised Mean Square Error (NMSE) averaged over 50 independent experiments for the signal-to-noise ratios 5dB, 10 dB, 15 dB, 20 dB, and 25 dB, with different network dimensions 5000, 10000, 50000 and 100000.

Table 1.

Methods	500	5000	10000	50000	100000
CVX-reweighted lasso	81.9	428.4	1218.3	N/A	N/A
Distributed reweighted lasso with 2 cores	24.7	84.3	156.5	587.1	1341.5
Distributed reweighted lasso with 4 cores	16.4	64.1	91.5	411.8	868.7
Distributed reweighted lasso with 10 cores	15.5	46.9	62.9	345.2	788.3

Computation times in seconds

reduce the Normalised Mean Square Error in comparison with classic lasso algorithms. Furthermore, our distributed algorithm can deal with networks comprising more than 50,000 nodes, which centralised algorithms typically cannot handle.

8. ACKNOWLEDGEMENT

W. Pan gratefully acknowledges the support of Microsoft Research through the PhD Scholarship Program. A. Sootla and G.-B. Stan acknowledge the support of EPSRC through the project EP/J014214/1 and the EPSRC Science and Innovation Award EP/G036004/1.

REFERENCES

C.M. Bishop. *Pattern recognition and machine learning*, volume 4. springer New York, 2006.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

E.J. Candès and T. Tao. Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12): 4203–4215, 2005.

D.L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization. *Proceedings of the National Academy of Sciences*, 100(5):2197–2202, 2003.

M. Grant, S. Boyd, and Y. Ye. Cvx: Matlab software for disciplined convex programming. *Online accessiable: http://stanford. edu/boyd/cvx*, 2008.

L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1999.

J. Palmer, K. Kreutz-Delgado, B. D. Rao, and D. P. Wipf. Variational em algorithms for non-gaussian latent variable models. In *Advances in neural information processing systems*, pages 1059–1066, 2005.

W. Pan, Y. Yuan, J. Gonçalves, and G.-B. Stan. Bayesian approaches to nonlinear network reconstruction. *submitted*, 2013.

W. Pan, A. Sootla, and G.-B. Stan. Distributed Reconstruction of Nonlinear Networks: An ADMM Approach. *ArXiv e-prints*, 2014. arXiv 1403.7429.

S.H. Strogatz. From kuramoto to crawford: exploring the onset of synchronisation in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1): 1–20, 2000.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

M.E. Tipping. Sparse bayesian learning and the relevance vector machine. *The Journal of Machine Learning Research*, 1:211–244, 2001.

D. Wipf and S. Nagarajan. Iterative reweighted l1 and l2 methods for finding sparse solutions. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):317–329, 2010.

D.P. Wipf, B.D. Rao, and S. Nagarajan. Latent variable bayesian models for promoting sparsity. *Information Theory, IEEE Transactions on*, 57(9):6236–6255, 2011.