Aalto University
School of Science

Tuomas Rintamäki

A parallel implementation of the ADMM algorithm for power network control

 We acknowledge the computational resources provided by the Aalto Science-IT project.

Mat-2.4108 Independent Research Project in Applied Mathematics.

Espoo, March 31, 2015

Supervisor: Professor Ahti Salo
Instructor: Professor Ahti Salo

Aalto University

School of Science

Degree Programme in Engineering Physics and Mathematics ABSTRACT

Author: Tuomas Rintamäki

Title: A parallel implementation of the ADMM algorithm for power network control

Date: March 31, 2015 **Pages:** 3+18

Major subject: Systems and Operations Research

Supervisor: Professor Ahti Salo

Instructor: Professor Ahti Salo

In this study, a distributed optimization algorithm called alternating direction method of multipliers (ADMM) is applied with receding horizon control (RHC) to a energy management problem for a power network. The solve times are in the order of milliseconds and they are observed to be constant with respect to the problem size when the number of processors varies accordingly. Consequently, the RHC controller could be implemented for operating our simple network at a kilohertz sampling rate. The model is implemented in MPI and is fully parallellized.

Keywords: Power network modelling, distributed optimization, ADMM, receding horizon control, MPI

Language: English

Contents

1 Introduction

Convex optimization problems with huge datasets can sometimes be solved efficiently using distributed optimization techniques. For example, Boyd et al. (2010) solves a least squares problem with a 30 GB dataset in 6 minutes using an algorithm called alternating direction method of multipliers (ADMM). In this study, ADMM is presented and applied to a network energy management problem using a parallel implementation. The algorithm has been applied to optimal power flow problems in a smart grid (see Kraning et al., 2013 and Chakrabarti et al., 2014, for example), image processing (Chen et al., 2015), wireless communication (Leinonen et al., 2013), as well as data mining (Lubell-Doughtie and Sondag, 2013), and found to exhibit linear convergence rate. In many applications, ADMM outperforms state-of-the-art serial algorithms (see Ramani and Fessler, 2012 for a comparison in a biomedical application).

However, the standard ADMM has been found to exhibit scalability issues in some problem types, which has motivated the development of scalable modifications (see Erseghe, 2014, for example). Parallel implementation also requires communication between threads and variants are developed to minimize the need for communication for security and data loss reasons (Mota et al., 2012). Since high precision solutions are found to take many ADMM iterations, ADMM is best suited for applications, where modest accuracy is enough (Boyd et al., 2010).

Historically, distributed and parallel algorithms have already been discussed in Bertsekas and Tsitsiklis (1989). However, they were not accessible until recently as standard computers now have multicore processors and cloud computing services such as Amazon EC2 are becoming cheaper. Moreover, general purpose computing on graphics processing units (GPGPU) allows running algorithms on thousands of threads on a single machine. Moreover, tools such as CVXGEN (Mattingley and Boyd, 2015) and Message Passing Interface (MPI) allow rapid development of parallel algorithms.

The outline of this study is as follows. In Section 2, we present algorithms that ADMM is based on. Then, in Section 3 the standard ADMM problem is presented and a few examples

provided. Section 4 introduces our application of ADMM to an energy management problem and in Section 5 we optimize power flows and generation under load imbalances using receding horizon control, which utilises the scalability of ADMM. Finally, Section 6 concludes by presenting ideas for developing the model further.

2 Preliminaries

Following Boyd et al. (2010), we briefly review dual ascent, dual decomposition and the method of multipliers that are precursors to alternating direction method of multipliers.

2.1 Dual ascent

Consider a convex optimization problem

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{2.1}$$

with $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $f : \mathbb{R}^n \to \mathbb{R}$ convex. The Lagrangian for Eq. (2.1) is

$$L(x,y) = f(x) + y^\top (Ax - b), \tag{2.2}$$

where $y \in \mathbb{R}^m$ is the vector of dual variables. The associated dual function is

$$g(y) = \inf_x L(x,y). \tag{2.3}$$

The dual problem is

$$\text{maximize} \, g(y). \tag{2.4}$$

Assuming that f has an unique optimizer and that strong duality holds, we can recover the optimal primal point x^* from the optimal dual point y^* as

$$x^* = \underset{x}{\text{argmin}} \, L(x,y^*). \tag{2.5}$$

Dual ascent is the gradient method for the dual problem that consist of the following steps (Boyd et al., 2010)

$$x^{k+1} = \underset{x}{\operatorname{argmin}} L(x, y^k) \tag{2.6}$$

$$y^{k+1} = y^k + \alpha^k \nabla g(y^k), \tag{2.7}$$

where $\nabla g(y^k) = Ax^{k+1} - b$ and α^k is the step size for iteration k. The drawback of dual ascent is that it may converge slowly and the x-update step can become unbounded from below.

2.2 Dual decomposition

If the function f is separable as $f = \sum_{i=1}^{n} f_i(x_i)$, where x_i are subvectors of $x = (x_1, \ldots, x_n)$, and the matrix A can be partitioned into $[A_1 \ldots A_n]$ so that $Ax = \sum_{i=1}^{n} A_i x_i$, then the Lagrangian can be written as

$$L(x, y) = \sum_{i=1}^{n} (f_i(x_i) + y^\top A_i x_i - (1/n) y^\top b). \tag{2.8}$$

Consequently, the x-minimization step in dual ascent in Eq. (2.6) can be solved independently and in parallel for each $i = 1 \ldots n$

$$x_i^{k+1} = \underset{x_i}{\operatorname{argmin}} L_i(x_i, y^k) \tag{2.9}$$

$$y^{k+1} = y^k + \alpha^k (Ax^{k+1} - b), \tag{2.10}$$

where $L_i = f_i(x_i) + y^\top A_i x_i - (1/n) y^\top b$.

2.3 Method of multipliers

The convergence properties of dual ascent can be improved by introducing the augmented Lagrangian function (Boyd et al., 2010)

$$L_\rho(x, y) = f(x) + y^\top (Ax - b) + (\rho/2) \|Ax - b\|_2^2, \tag{2.11}$$

where $\rho > 0$ is a penalty parameter. Thanks to the quadratic penalty, assumptions such as finiteness and strict convexity of f are not required. When dual ascent is applied to the augmented problem with ρ as the step size, we obtain the method of multipliers

$$x^{k+1} = \operatorname*{argmin}_x L(x, y^k) \tag{2.12}$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} - b). \tag{2.13}$$

The choice of ρ as the step size is motivated by the first-order optimality condition (Boyd et al., 2010)

$$\nabla_x L_\rho(x^{k+1}, y^k) = \nabla_x f(x^{k+1}) + A^\top(y^k + \rho(Ax^{k+1} - b)) \tag{2.14}$$

$$= \nabla_x f(x^{k+1}) + A^\top y^{k+1} = 0. \tag{2.15}$$

3 Alternating direction method of multipliers

Consider the optimization of two variables with a separable objective function

$$\begin{aligned} \underset{x,z}{\text{minimize}} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c, \end{aligned} \tag{3.1}$$

with f and g convex, $x \in \mathbb{R}^n$, $z \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$. The augmented Lagrangian of this problem with penalty parameter ρ is

$$L_\rho(x, z, y) = f(x) + g(z) + y^\top(Ax + Bz - c) + (\rho/2)\,\|Ax + Bz - c\|_2^2. \tag{3.2}$$

The method of multipliers for this problem is

$$(x^{k+1}, z^{k+1}) = \operatorname*{argmin}_{x,z} L(x, z, y^k) \tag{3.3}$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \tag{3.4}$$

Because of the quadratic penalty term $(\rho/2)\,\|Ax + Bz - c\|_2^2$ and joint minimization of x and z, the separability of the objective function cannot be exploited (Boyd et al., 2010). ADMM

addresses this issue by introducing the iterations

$$x^{k+1} = \underset{x}{\operatorname{argmin}} L(x, z^k, y^k) \tag{3.5}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} L(x^{k+1}, z, y^k) \tag{3.6}$$

$$y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \tag{3.7}$$

The algorithm can be written in a slightly different form by defining the residual $r = Ax + Bz - c$ and a scaled dual variable $u = (1/\rho)y$. Using these variables, the augmented Lagrangian becomes

$$L_\rho(x,z,y) = f(x) + g(z) + y^\top r + (\rho/2)\|r\|_2^2 \tag{3.8}$$

$$= f(x) + g(z) + (\rho/2)\|r + (1/\rho)y\|_2^2 - (1/2\rho)\|y\|_2^2 \tag{3.9}$$

$$= f(x) + g(z) + (\rho/2)\|r + u\|_2^2 - (\rho/2)\|u\|_2^2 \tag{3.10}$$

$$= L_\rho(x,z,u). \tag{3.11}$$

By ignoring the last term of $L_\rho(x,z,u)$ as a constant, the iterations become

$$x^{k+1} = \underset{x}{\operatorname{argmin}} \left(f(x) + (\rho/2)\left\| Ax + Bz^k - c + u^k \right\|_2^2 \right) \tag{3.12}$$

$$z^{k+1} = \underset{z}{\operatorname{argmin}} \left(g(z) + (\rho/2)\left\| Ax^{k+1} + Bz - c + u^k \right\|_2^2 \right) \tag{3.13}$$

$$u^{k+1} = u^k + Ax^{k+1} + Bz - c. \tag{3.14}$$

Boyd et al. (2010) shows the convergence of the algorithm in case of two convex functions, Hong and Luo (2013) for any number of functions, and Ghadimi et al. (2015) analyzes optimal penalty parameter selection for quadratic problems, for example. The necessary and sufficient optimality conditions for the problem (3.1) are primal feasibility

$$Ax^* + Bz^* - c = 0, \tag{3.15}$$

and dual feasibility

$$\nabla f(x^*) + A^\top y^* = 0 \tag{3.16}$$

$$\nabla g(z^*) + B^\top y^* = 0. \tag{3.17}$$

Primal feasibility motivates the stopping criteria $\left\|r^k\right\|_2^2 \leq \varepsilon_{pri}$, where ε_{pri} is a small positive constant. Because x^{k+1} minimizes $L_\rho(x, z^k, y^k)$, we have

$$0 = \nabla f(x^{k+1}) + A^\top \left(y^k + \rho(Ax^{k+1} + Bz^k - c)\right) \tag{3.18}$$

$$= \nabla f(x^{k+1}) + A^\top \left(y^k + \rho r^{k+1} + \rho(Bz^k - Bz^{k+1})\right) \tag{3.19}$$

$$= \nabla f(x^{k+1}) + A^\top y^{k+1} - \rho A^\top B(z^{k+1} - z^k). \tag{3.20}$$

Thus, the quantity $s^k = \rho A^\top B(z^{k+1} - z^k)$ can be interpreted as dual residual, which gives the second stopping criteria $\left\|s^k\right\|_2^2 \leq \varepsilon_{dual}$.

3.1 Proximal operators

A useful reformulation of the ADMM algorithm can be stated by using proximal operators. Following Parikh and Boyd (2013), the proximal operator $\mathbf{prox}_{\rho f} : \mathbb{R}^n \to \mathbb{R}^n$ of a scaled convex function $\rho f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ evaluated at point $v \in \mathbb{R}^n$ with $\rho > 0$ is defined by

$$\mathbf{prox}_{\rho f}(v) = \underset{x}{\operatorname{argmin}} \left(f(x) + \frac{1}{2\rho} \|x - v\|_2^2 \right). \tag{3.21}$$

The function minimized on the right hand side is strictly convex, and, thus, it has a unique minimizer. The parameter ρ represents a trade-off between minimizing f and being near to v. For example, when f is the indicator function

$$I_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{if } x \notin C, \end{cases}$$

where C is a closed nonempty convex set, the proximal operator of f reduces to

$$\mathbf{prox}_f(v) = \underset{x \in C}{\operatorname{argmin}} \|x - v\|_2, \tag{3.22}$$

which is the Euclidean projection of v onto a set C denoted by $\Pi_C(v)$.

Consider the problem (see problem (3.1))

$$\begin{aligned} &\text{minimize} && f(x) + g(z) \\ &\text{subject to} && x - z = 0, \end{aligned} \tag{3.23}$$

which is called the consensus form (Parikh and Boyd, 2013). Using the proximal operator, the ADMM iterations in Eqs. (3.12)-(3.14) with the scaled dual variable u become

$$x^{k+1} = \mathbf{prox}_{\rho f}(z^k - u^k) \tag{3.24}$$

$$z^{k+1} = \mathbf{prox}_{\rho g}(x^{k+1} + u^k) \tag{3.25}$$

$$u^{k+1} = u^k + x^{k+1} - z^{k+1}. \tag{3.26}$$

Note that in Eq. (3.25) the signs of the arguments were changed thanks to the quadratic term. Following Parikh and Boyd (2013), an exchange problem of the form

$$\begin{aligned} &\text{minimize} && \sum_{i=1}^{N} f_i(x_i) \\ &\text{subject to} && \sum_{i=1}^{N} x_i = 0, \end{aligned} \tag{3.27}$$

with $x_i \in \mathbb{R}^n$, $i = 1,\ldots,N$ can be rewritten in the above consensus form

$$\text{minimize} \quad \sum_{i=1}^{N} f_i(x_i) + I_C(x_1,\ldots,x_N), \tag{3.28}$$

where $C = \{(x_1,\ldots,x_N) \in \mathbb{R}^{nN} \,|\, x_1 + \cdots + x_N = 0\}$. The z-update in Eq. (3.25), i.e., the projection of $x^{k+1} + u^k$ onto C can be computed analytically and is given by de-meaning

$$(\Pi_C(x_1^{k+1} + u^k, ..., x_N^{k+1} + u^k))_i = x_i^{k+1} + u^k - \bar{x}^{k+1} - \bar{u}^k = x_i^{k+1} - \bar{x}^{k+1}. \tag{3.29}$$

Thus, the iterations in Eqs. (3.24)-(3.26) for this problem reduce to

$$x_i^{k+1} = \mathbf{prox}_{\rho f_i}(x_i^k - \bar{x}^k - u^k) \tag{3.30}$$

$$u^{k+1} = u^k + \bar{x}^{k+1}, \tag{3.31}$$

where the x-update can be carried out independently and in parallel for each $i = 1 \ldots N$.

4 Exchange problem in an electrical power system

Kraning et al. (2013) develops an energy management model for a large-scale electrical power network using the proximal formulation of ADMM. The model minimizes the cost functions

of devices such as generators subject to fixed and curtailable loads, transmission constraints and many other details, while maintaining the power balance in subsets of the whole network called nets. The problem is solved in a distributed manner by alternating between the parallel optimization of single device objective functions and computing average power imbalances in the nets which the devices belong to. They implement the parallel optimization of the device objective functions in OpenMP with 64 threads, whereas the net imbalances are computed serially because the overhead from spawning the threads overcomes that of computing the averages. They note that, in theory, the solution time can be expected to be constant if the number of threads matches the number of devices.

We present a very simple network which is partitioned to nets which contain a number of unconstrained generators and fixed loads. The nets are connected through unconstrained direct current (DC) tranmission lines. An object such as a generator is represented by a power schedule p_d which is a scalar for single terminal devices (generators and loads) and a vector for multiterminal devices (transmission lines). The elements of p_d, denoted by p_t, are positive if the terminal is a source of power and negative if the terminal is a sink. The problem is implemented in MPI, which supports the distributed memory framework, where every process has its own private memory and the sharing of private information is done through message passing. The problem is as follows

$$\begin{aligned} &\text{minimize} && f(p) \\ &\text{subject to} && \bar{p} = 0, \end{aligned} \tag{4.1}$$

where p is the vector of all power schedules in the network, and $\bar{p} = \frac{1}{T}\sum_{t=1}^{T} p_t$ the average power schedule over all terminals t. For terminals, we define $\bar{p}_t = \frac{1}{|n|}\sum_{t' \in n} p_{t'}$, where $t \in n$ and $|n|$ is the number of terminals in net n. Following Kraning et al. (2013), the notation is overloaded for devices and nets by defining $\bar{p}_d = \{\bar{p}_t \,|\, t \in d\}$ and $\bar{p}_n = \{\bar{p}_d \,|\, d \in n\}$, respectively. Effectively, $\bar{p}_n$ contains $|n|$ copies of $\bar{p}_t$ for net n. The objective function is separable as each device has its own cost function. Moreover, the whole network is in balance only if all of its nets are in balance.

Consequently, the problem can reformulated as

$$\begin{aligned} &\text{minimize} \quad \sum_{d=1}^{D} f_d(p_d) + \sum_{n=1}^{N} g_n(z_n) \\ &\text{subject to} \quad p = z, \end{aligned} \tag{4.2}$$

where p_d is the power schedule of a device d, f_d its cost function, z_n the vector of power schedules p_d in the net n, i.e., $z_n = \{p_d \,|\, d \in n\}$, and $g_n(z_n)$ is the indicator function on the set $\{z_n \,|\, \bar{z}_n = 0\}$. Similarly, we define z_d to be the part of p pertaining to device d. The augmented Lagrangian of the problem is

$$L_\rho(x,z,u) = \sum_{d=1}^{D} f_d(p_d) + \sum_{n=1}^{N} g_n(z_n) + (\rho/2)\,\|p - z + u\|_2^2 \tag{4.3}$$

The quadratic penalty term can be split across devices or nets

$$\|p - z + u\|_2^2 = \sum_{d=1}^{D} \|p_d - z_d + u_d\|_2^2 = \sum_{n=1}^{N} \|p_n - z_n + u_n\|_2^2 \tag{4.4}$$

Consequently, we can write the ADMM iterations as follows

$$p_d^{k+1} = \mathbf{prox}_{\rho f_d}(p_d^k - \bar{p}_d^k - u_d^k) \tag{4.5}$$

$$u_n^{k+1} = u_n^k + \bar{p}_n^{k+1}, \tag{4.6}$$

where u_d and u_n include the elements of the scaled dual vector u pertaining to the device d and net n, respectively. The z-update, which equals to the one in Eq. (3.29), has been plugged in to the iterations.

In practice, the power schedules of the fixed loads remain unchanged so they need not be updated but only initialised once. For a generator i, we set a fixed generation cost c_i, which allows the proximal operator in Eq. (4.5) to be computed analytically. On the other hand, transmission line j with terminal power schedules $p_{t,j1}$ and $p_{t,j2}$ has no cost function, and, thus, the step (4.5) reduces to a projection onto a hyperplane $p_{t,j1} + p_{t,j2} = 0$, which implies zero losses on the line. We limit ourselves to unconstrained generators and transmission lines as they allow us to solve all equations analytically without a need for additional software libraries.

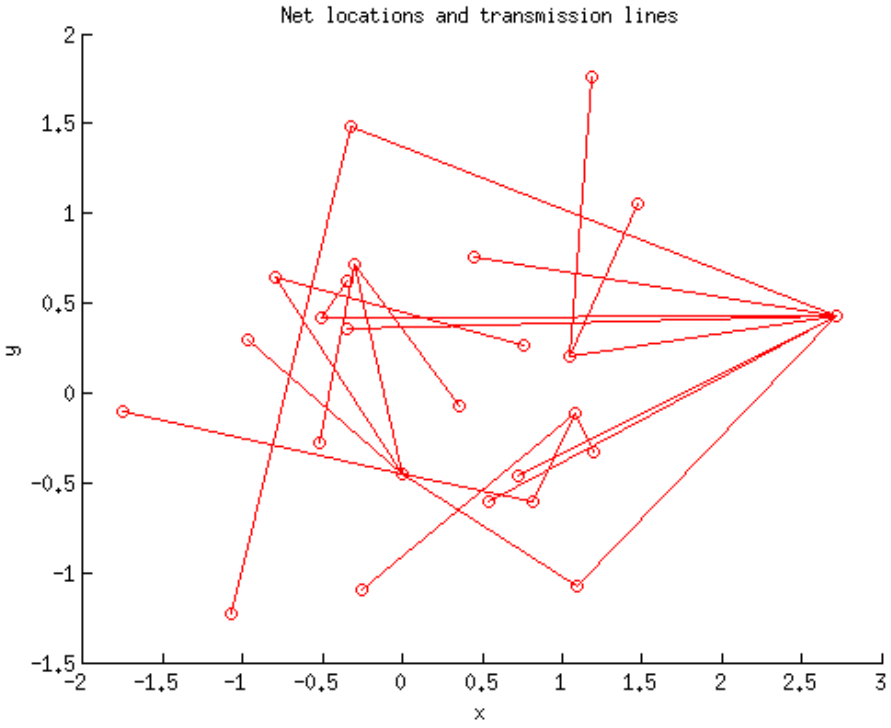

Figure 1: Nets and transmission lines in the sample network

5 Receding horizon control under load imbalances

Receding horizon control (RHC) is a feedback control technique, where an optimization problem is solved at each time step to determine the control input for a fixed time horizon (Mattingley et al., 2011). The optimization takes into account real-time measurements and estimates of future quantities based on them. With RHC, tedious tuning of a controller such as PID is not required.

We solve initial power schedules for $\mathscr{T} = 60$ seconds based on an estimate of the fixed loads in a power network in Figure 1. There is no temporal connection between consecutive seconds but the optimization is repeated $\mathscr{T}$ times with fixed load schedules varying in time. There are 25 nets, which each contains maximum 2 loads, maximum 2 generators and a connection to a random net. There are no duplicate transmission lines.

A sample problem has a total of 156 nets and terminals of which each is assigned an in-

dependent MPI process. The implementation is summarized in Algorithm 1. First, the power schedules for the fixed loads are initialised. Then, the main ADMM loop is entered, where parallel runs alternate between the devices and nets at steps 5.1. and 5.3. After a net has completed updating its average imbalance and the dual variable, the results are communicated to its devices using MPI Bcast (step 5.1.4.). If all nets have imbalance less than a threshold value ε_{pri}, the loop is discontinued and the current power schedules are returned. Otherwise, each device uses the imbalance and dual variable of its net to optimize its power schedule. The updated power schedules of each device in a net are gathered and sent to the net using MPI Reduce (step 5.1.1.). For our sample network, the initial power schedules were determined in 132 iterations and 22 milliseconds. By comparison, on a four core Intel Core i5 CPU clocking at 2.40 GHz the optimization takes around 5 seconds.

We test a simple RHC controller by introducing load imbalances during each second which are modelled with the following simple process for a load l

$$imb_{l,\tau} = -b - w_\tau \qquad \tau = 1 \ldots \mathscr{T}, w_\tau \sim \text{ Unif}[-a,a], \tag{5.1}$$

where a and b are constants. The load terminals are assumed to know the functional form of the imbalances but not the parameters. As the expected value of $imb_{l,\tau}$ is b, each load terminal updates its estimate for future imbalances by averaging previously observed imbalances

$$\hat{imb}_l = \frac{1}{\tau}\sum_{\tau'}^{\tau} imb_{l,\tau'}. \tag{5.2}$$

Such a learning process may happen in real power networks, when load is continously lower than forecasted because of higher temperature, for example, but the observations are noisy.

Every second, the current load imbalance is observed and the estimate of future imbalances updated. The RHC controller optimizes generation for the current second and $T' = 5$ seconds ahead using Algorithm 1. Solving each of these optimization problems took approximately 5 ms for the sample problem, i.e., an optimized controller could be implemented at a kilohertz sampling rate for a large scale network. On the four core system, each RHC run takes 1-2 seconds. The algorithm benefits from a warm start provided by the initial schedules and con-

Algorithm 1: Algorithm for obtaining initial power schedules

1. Initialise a MPI process for each net and terminal
2. Read fixed loads, generation cost parameters and transmission line incidence from input files
3. Initialise p_d for fixed loads
4. For each $\tau \in \mathscr{T}$ repeat
5. Do while iteration $<$ maximum iteration
   - 5.1. For each n do in parallel
     - 5.1.1. MPI Reduce p_d^k from devices to their nets
     - 5.1.2. Compute $\bar{p}_n^{k+1}$
     - 5.1.3. Update u_n^{k+1} using (4.6)
     - 5.1.4. MPI Bcast $\bar{p}_n^{k+1}$ and u_n^{k+1} from nets to their devices
   - 5.2. Exit if $\bar{p}_d < \varepsilon_{pri}$ for all n
   - 5.3. For each d do in parallel
     - 5.3.1. Update p_d^{k+1} using (4.5)

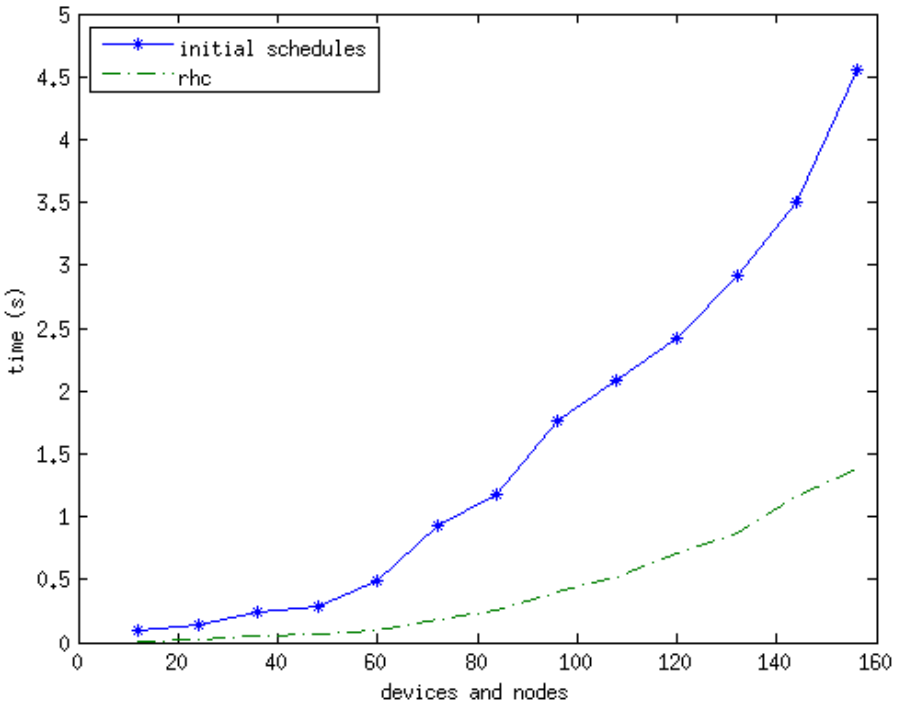

Figure 2: Running times using a quad-core PC

tinuous updates of the load imbalance as the average number of iterations drops approximately to 95.

More detailed solution time results are in Figures 2 and 3 for the quad-core PC and a cluster, respectively. For the PC, solution times seem to grow fast, when the number of required processes increases. This can result from the increasing overhead of spawning the threads, reading input files and scheduling the different processes when there are only four cores available. However, the solution times stay nearly the same when the cluster is used. Small differences can be caused by different CPUs being allocated for different runs and the computational load of the cluster. Moreover, Figure 4 shows how the number of iterations grows for a problem with a total of 48 nodes and devices using the quad-core PC. The iterations could be reduced slightly by finetuning the step size parameter ρ as Figure 5 shows.

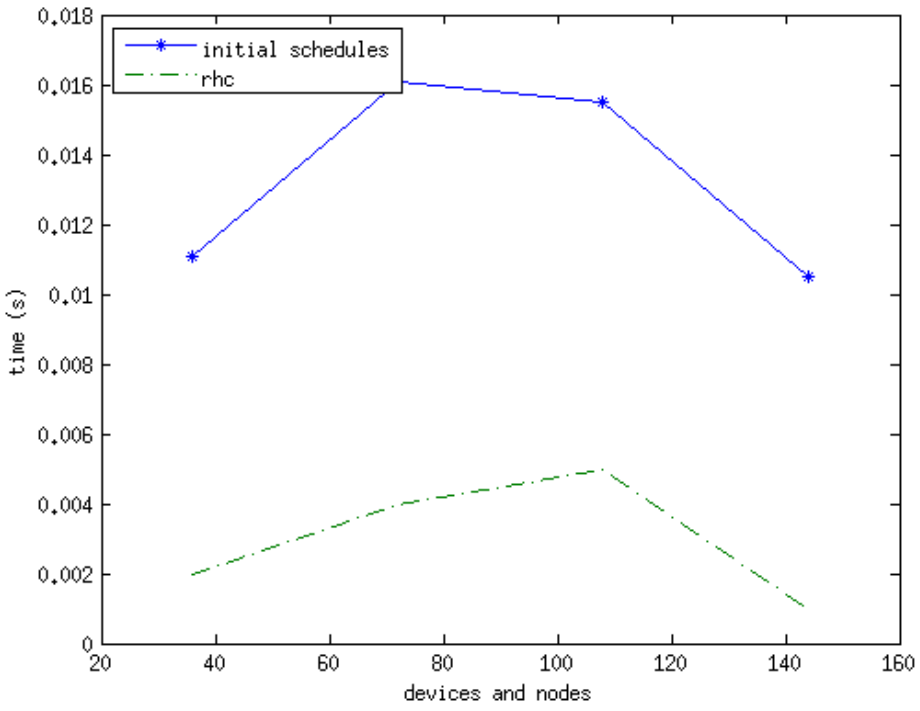

Figure 3: Running times using a cluster

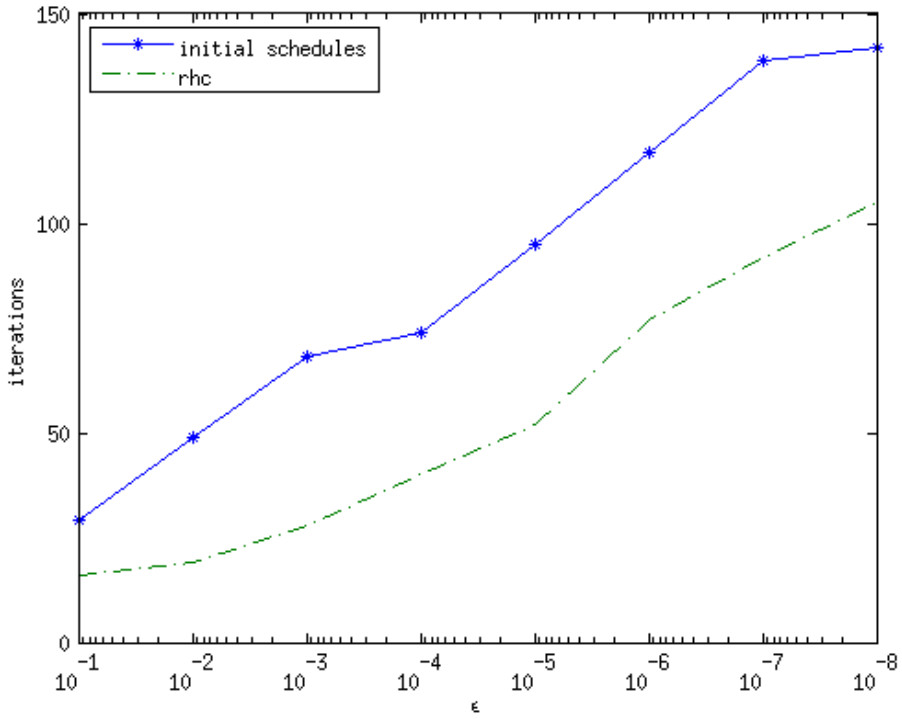

Figure 4: Number of iterations for a problem of 48 processes with different stopping criteria

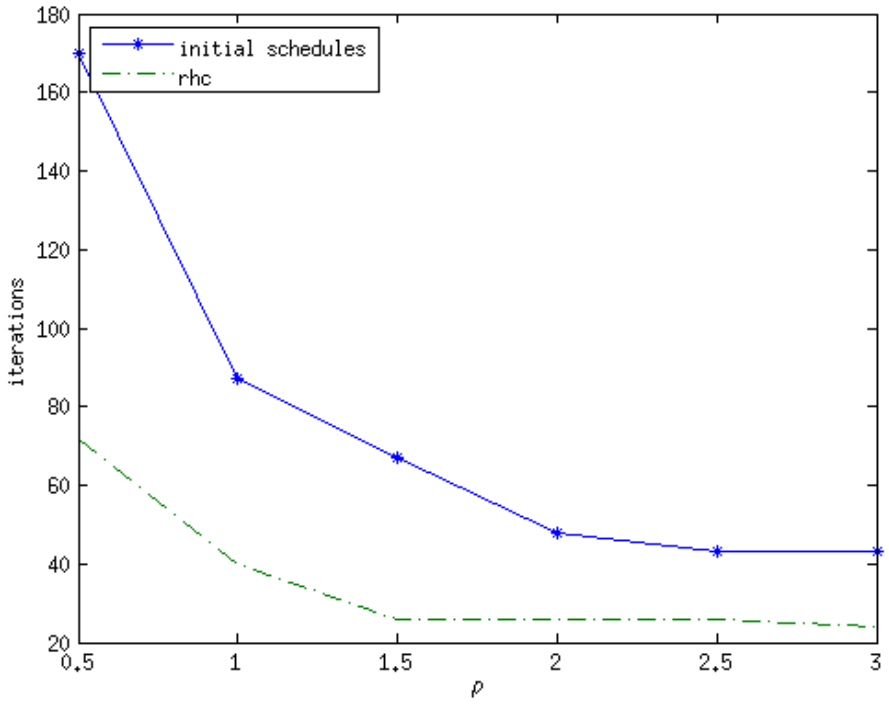

Figure 5: Solution times with different values of ρ for a problem with 48 processes with $\varepsilon_{pri} = 10^{-5}$

6 Conclusion

In this study, a distributed optimization algorithm called alternating direction method of multipliers was presented. The algorithm was applied to a small energy management problem, which was fully parallelized. Nearly constant solve time was observed when the number of processes increased with the number of devices and nets.

The model could be extended by adding more features to the network objects. For example, generators and transmission lines have a constrained operating interval. Moreover, additional objects such as alternating current (AC) transmission lines and flexible loads could be introduced similar to Kraning et al. (2013). However, the simple model presented in this study could be utilized in power market simulations, in situations in which low level of detail suffices.

On the other hand, the implementation could be developed to utilize GPUs through NVIDIA CUDA, for example, as GPUs have a larger number of threads than CPUs. Although GPU threads have less computing power, only a small difference can be expected because each device and net solves a relatively simple optimization problem that would not benefit from access to more computing capacity.

References

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and Distributed Computation: Numerical Methods*. Prentice-Hall, Upper Saddle River, New Jersey.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Chakrabarti, S., Kraning, M., Chu, E., Baldick, R., and Boyd, S. (2014). Security constrained

optimal power flow via proximal message passing. In *Proceedings of the Power Systems Conference (PSC), 2014 Clemson University*, pages 1–8.

Chen, C., Ng, M., and Zhao, X.-L. (2015). Alternating direction method of multipliers for nonlinear image restoration problems. *IEEE Transactions on Image Processing*, 24(1):33–43.

Erseghe, T. (2014). Distributed optimal power flow using ADMM. *IEEE Transactions on Power Systems*, 29(5):2370–2380.

Ghadimi, E., Teixeira, A., Shames, I., and Johansson, M. (2015). Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658.

Hong, M. and Luo, Z.-Q. (2013). On the linear convergence of the alternating direction method of multipliers. `http://arxiv.org/abs/1208.3922v3`. Accessed 26 March 2015.

Kraning, M., Chu, E., Lavaei, J., and Boyd, S. (2013). Dynamic network energy management via proximal message passing. *Foundations and Trends in Optimization*, 1(2):70–122.

Leinonen, M., Codreanu, M., and Juntti, M. (2013). Distributed joint resource and routing optimization in wireless sensor networks via alternating direction method of multipliers. *IEEE Transactions on Wireless Communications*, 12(11):5454–5467.

Lubell-Doughtie, P. and Sondag, J. (2013). Practical distributed classification using the alternating direction method of multipliers algorithm. In *Proceedings of 2013 IEEE International Conference on Big Data*, pages 773–776.

Mattingley, J. and Boyd, S. (2015). CVXGEN: Code generation for convex optimization. `http://cvxgen.com`. Accessed 31 March 2015.

Mattingley, J., Wang, Y., and Boyd, S. (2011). Receding horizon control. *Control Systems, IEEE*, 31(3):52–65.

Mota, J., Xavier, J., Aguiar, P., and Puschel, M. (2012). D-ADMM: A distributed algorithm for compressed sensing and other separable optimization problems. In *Proceedings of 2012*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2869–2872.

Parikh, N. and Boyd, S. (2013). Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):123–231.

Ramani, S. and Fessler, J. (2012). A splitting-based iterative algorithm for accelerated statistical X-Ray CT reconstruction. *IEEE Transactions on Medical Imaging*, 31(3):677–688.