Distributed Optimization
Augmented Lagrangian ADMM

Swapnil Saha, Brian B Cheng
Department of Electrical and Computer Engineering, Rutgers University

1 Abstract

In the field of convex optimization, the Alternating Direction Method of Multipliers (ADMM) is an old but versatile optimization method. The ADMM combines the decomposability of dual ascent with the convergence properties of the method of Lagrangian multipliers to produce an algorithm whose computations can be naturally partitioned off to independent nodes for distributed computation. This study explores the theoretical foundations of ADMM and then explores a distributed ADMM method using a global consensus strategy. We implement the ADMM in Python and apply it to Lasso Regression on a square dense matrix and report the convergence results. We first experiment with the basic non-distributed ADMM, then compare its results to our distributed ADMM implementation. All the simuation codes are uploaded in GitHub.

2 Introduction

Distributed optimization is increasingly popular in the fields of engineering and computer science for decision-making and data processing due to high computational demands [9, 11, 13]. The versatility of distributed optimization is a key aspect of its appeal. It involves the collaboration of dispersed smart devices or microprocessors to achieve a specific system-level goal, regardless of the field. In engineering systems, it's a tool for empowerment, allowing subsystems to make decisions locally while working together to achieve optimal system performance. In computer science, it's a solution, distributing a heavy training task across multiple microprocessors and

coordinating them to achieve a unified training objective. The scenarios may vary, but the philosophy of distributed optimization remains constant, breaking down a comprehensive mathematical optimization problem into smaller, more manageable sub-problems. This approach empowers multiple computing agents to solve these sub-problems in a coordinated manner, leading to an optimal or near-optimal solution to the original mathematical optimization problem.

The Alternating Direction Method of Multipliers (ADMM) has become a popular tool for distributed optimization due to its modular structure (decomposability), superior convergence, easy implementation, and high flexibility. ADMM has been applied in various fields such as statistical learning [3, 5], multi-agent reinforcement learning [12], imaging processing[1, 2], multi-robot coordination [6], wireless communication control[10, 7], and more.

The remaining of the report is organized as follows: section 4 describes background knowledge, section 5 talks about basic ADMM, section 6 formulates the problems that we are interested in solving with ADMM, section 7 shows the simulation result and section 8 describes the future work. Most of the work preseted in this project is inspired by [3].

3 Notation

We use bold lower-case letters ($\mathbf{v}$), bold upper-case letters ($\mathbf{V}$) and unbolded letters (M) to denote vectors, matrices and scalars, respectively. Superscript k is used to indicate the iteration instant, and subscript i is used to indicate local node number. For element-wise matrix multiplication, we use the notation $\odot$. We denote the $\mathcal{L}_2$ norm (Euclidean norm) with $\|.\|_2$, the $\mathcal{L}_{1,1}$ norm with $\|.\|_{1,1}$, and the Frobenius norm with $\|.\|_F$. $*$ denotes the optimum convergent point. $\mathbb{R}$, and $\mathbb{R}_+$ denote the set of real numbers and set of positive real numbers, respectively.

4 Background

4.1 Dual Problem

Consider the following constrained optimization problem

$$\begin{aligned} \text{minimize} \quad & f(x) \\ \text{subject to} \quad & Ax = b, \end{aligned} \tag{1}$$

with variable $x \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m \times n}$ and $f : \mathbb{R}^n \to \mathbb{R}$ is convex. The Lagrangian of problem (1) is as follows:

$$L(x, y) = f(x) + y^\top (Ax - b) \tag{2}$$

The dual function is

$$g(y) = \inf_x L(x, y) \tag{3}$$

where y is the dual variable. The dual problem is

$$\text{maximize} \quad g(y) \tag{4}$$

with variable $y \in R^m$. Assuming that strong duality holds, the optimal values of the primal and dual problems are the same. We can recover a primal optimal point x^* from a dual optimal point y^* as

$$x^* = \operatorname*{argmin}_x L(x, y^*) \tag{5}$$

4.1.1 Dual Ascent

As Equ. (3) is point wise infimum, the dual function $g(y)$ is concave function [4]. Thus gradient ascent method, which is known as also dual ascent method in this context, can be applied to find the optimum dual variable y^*. If g is differentiable, the gradient $\nabla g(y)$ can be evaluated as follows. We first find $x^+ = \operatorname{argmin}_x L(x, y)$; then we have

$$\nabla g(y) = Ax^+ - b,$$

which is the residual for the equality constraint. The dual ascent method consists of iterating the updates

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L(x, y^k), \\ y^{k+1} &:= y^k + \alpha^k (Ax^{k+1} - b), \end{aligned} \tag{6}$$

where $\alpha^k > 0$ is a step size, and the superscript k denotes the iteration number. The first step, Equ.(6), is an x-minimization step, and the second is a dual variable update.

4.1.2 Dual decomposition

The most important advantage that the dual ascent method enjoys is the dual decomposition. That makes the algorithm so useful in decentralized system, the ADMM also used similar kind of advantage.
If the objective function f is separable (with respect to a partition or splitting of the variable into subvectors), meaning that

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

where $x = (x_1, \ldots, x_N)$ and the variables $x_i \in \mathbb{R}^{n_i}$ are subvectors of x. Partitioning the matrix A comfortably as

$$A = [A_1 \ldots A_N],$$

so $Ax = \sum_{i=1}^{N} A_i x_i$, the Lagrangian can be written as

$$L(x, y) = \sum_{i=1}^{N} L_i(x_i, y) = \sum_{i=1}^{N} \left(f_i(x_i) + y^\top A_i x_i - \frac{1}{N} y^\top b \right),$$

which is also separable in x. This means that the x-minimization step (6) splits into N separate problems that can be solved in parallel. Explicitly, the algorithm is

$$\begin{aligned} x_i^{k+1} &:= \operatorname*{argmin}_{x_i} L_i(x_i, y^k) \\ y^{k+1} &:= y^k + \alpha^k (A x^{k+1} - b). \end{aligned} \tag{7}$$

The x-minimization in (7) step is carried out independently, in parallel, for each $i = 1, \ldots, N$. In this case, dual ascent method is also considered as dual decomposition.

4.2 Method of Multipliers

One better algorithm compared to the dual decomposition is the method of multipliers. Here, "better" indicates that one needs less required assumptions to apply the multipliers method than the dual decomposition, such as f can be nondifferentiable, taking on value $+\infty$.This is done by introducing augmented Lagrangian methods. But this "better" comes with a price; the method of multipliers loses the flexibility of the decomposition.

Augmented Lagrangian methods: The augmented Lagrangian for (1) is

$$L_\rho(x, y) = f(x) + y^\top (Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2, \tag{8}$$

where $\rho > 0$ is called the penalty parameter. The associate dual function is

$$g_\rho(y) = \inf_x L_\rho(x, y). \tag{9}$$

Now g_ρ can be shown to be differentiable under rather mild conditions compared to the original problem. For instance, if $f(x)$ is convex, then $L_\rho(x, y)$ is strongly convex, and thus method of multipliers converges under far more general conditions than dual ascent.
Applying dual ascent to the modified problem yields the algorithm

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, y_k), \\ y^{k+1} &:= y^k + \rho(Ax_{k+1} - b), \end{aligned} \tag{10}$$

what are the conditions on the original problem?

which is known as the method of multipliers for solving our original constrained problem (1). The downside of the method of multipliers is that $L_\rho(x, y)$ is not decomposable. Here is where the ADMM algorithm comes to play. ADMM enjoys the robustness of the method of multipliers, which can also support decomposition.

5 Alternating Direction Method of Multipliers

The (Alternating Direction Method of Multipliers) ADMM enjoys both the decomposability of the dual ascent and the optimum convergence properties of the multiplier method. Thus, it can support both the multiplier method's good robustness and the decomposition of the dual ascent. The algorithm solves problems in the following format.

$$\begin{aligned} &\text{minimize} && f(x) + g(z) \\ &\text{subject to} && Ax + Bz = c \end{aligned} \tag{11}$$

where $x \in \mathbf{R}^n$ and $z \in \mathbf{R}^m$ with $A \in \mathbf{R}^{p\times n}$ and $B \in \mathbf{R}^{p\times m}$. The specific assumptions of the Equ.(11) will be discussed in the section.5.2

In the light of the method of multipliers, the augmented Lagrangian of problem (11) is as follows:

$$L_p(x, y, z) = f(x) + g(z) + y^T(Ax + bz - c) + (p/2)\|Ax + Bz - c\|_2^2 \quad (12)$$

where p is the penalty parameter. Due to this penalty parameter p, the augmented Lagrangian (12) converges under more relaxed conditions. The ADMM consists of the following iterations.

$$x^{k+1} := \underset{x}{\operatorname{argmin}}\ L_\rho(x, z^k, y^k) \quad (13)$$

$$z^{k+1} := \underset{z}{\operatorname{argmin}}\ L_\rho(x^{k+1}, z, y^k) \quad (14)$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) \quad (15)$$

The Equ.(13) and Equ.(34) are primal variable minimization, and (15) is the dual variable minimization. As seen from the above equations, the primal variables, x and z, update their values in an alternating or sequential fashion instead of updating jointly. Separating the minimization over x and z into two different setups makes the ADMM decomposable when f and g are separable.

5.1 Optimality Conditions

The necessary and sufficient conditions optimality conditions are

- Primal Feasibility
- Dual Feasibility

Primal feasibility implies that

$$Ax^* + Bz^* - c = 0 \quad (16)$$

Dual feasibility implies that

$$\begin{aligned} \nabla_x L(x^*, y^*, z^*) &= 0 \\ \nabla_z L(x^*, y^*, z^*) &= 0 \end{aligned} \quad (17)$$

From Equ.(17), it implies that

$$\nabla f(x^*) + A^T y^* = 0 \quad (18)$$

$$\nabla g(z^*) + B^T y^* = 0 \tag{19}$$

Again as z^{k+1} minimizes $L_p(x^{k+1}, z, y^k)$, it implies that $\nabla_z L_p(x^{k+1}, z, y^k) = 0$

$$\begin{aligned} L_p(x^{k+1}, z, y^k) &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T r^{k+1} \\ &= \nabla g(z^{k+1}) + B^T y^{k+1} = 0 \end{aligned} \tag{20}$$

Equ.(20) suggests that Equ.(19) always holds for any value of k.

Also. as x^{k+1} minimizes $L_p(x, z^k, y^k)$, it also implies that $\nabla_x L_p(x, z^k, y^k) = 0$

$$\begin{aligned} L_p(x, z^k, y^k) &= \nabla f(x^{k+1}) + A^T y^k + \rho A^T (Ax^{k+1} + Bz^k - c) \\ &= \nabla f(x^{k+1}) + A^T (y^k + \rho r^{k+1} + \rho B(z^k - z^{k+1})) \\ &= \nabla f(x^{k+1}) + A^T y^{k+1} + \rho A^T B(z^k - z^{k+1}) = 0, \end{aligned} \tag{21}$$

which can be written as

$$\rho A^T B(z^{k+1} - z^k) = \nabla f(x^{k+1}) + A^T y^{k+1} \tag{22}$$

The quantity

$$s^{k+1} = \rho A^T B(z^{k+1} - z^k) \tag{23}$$

can be considered as the residual for the dual feasibility condition.

In the same format of dual residual, we can also define primal residual r^{k+1}

$$r^{k+1} = Ax^{k+1} + Bz^{k+1} - c \tag{24}$$

As ADMM proceeds, the two residuals Equ.(23) and Equ.(24) converge to zero.

5.2 Convergence Analysis

Chapter 3 - Swapnil Section 3.2.1 and Appendix A

The convergence analysis is based on the two assumptions.

Assumption 1: The (extended-real-valued) functions $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ are closed, and convex.

Assumption 1 implies that both the x-update ((13)) and z-update((34)) are solvable, that means there exist optimal value x^* and z^* which minimizes the augmented Lagrangian.

Assumption 2: The unaugmented Lagrangian L_0 has a saddle point. That means there exist (x^*, y^*, z^*), does need to be unique, for which $L_0(x^*, z^*, y) <= L_0(x^*, z^*, y^*) <= L_0(x, z, y^*)$

Assumption 2 is strong max-min property, which states that y^* maximizes $L_0(x^*, z^*)$ and (x^*, z^*) minimizes $L_0(x, z, y^*)$.

Under assumptions 1 and 2, each ADMM iteration satisfies the following :

- **Residual convergence**: $r_k \to 0$ as $k \to \infty$, i.e., the iterates approach to the primal feasibility.
- **Objective convergence**: $f(x_k) + g(z_k) \to p^*$ as $k \to \infty$, i.e., the objective function of the iterates approaches the optimal value.
- **Dual variable convergence**: $y_k \to y^*$ as $k \to \infty$, i.e., the iterates approach to the dual feasibility, where y^* is a dual optimal point.

The proof of residual convergence is based on the following inequality [3]:

$$V^{k+1} \leq V^k - \rho\|r^{k+1}\|_2^2 - \rho\|B(z^{k+1} - z^k)\|_2^2.$$

This implies that V^k decreases in each iteration by an amount that depends on the norm of the residual and on the change in z over one iteration. As $V_k \leq V_0$, it follows that y_k and Bz_k are bounded. Iterating the inequality

above gives that

$$\rho \sum_{k=0}^{\infty} \left(\|r^{k+1}\|_2^2 + \|B(z^{k+1} - z^k)\|_2^2 \right) \leq V^0, \tag{25}$$

which implies that $r_k \to 0$ and $B(z_{k+1} - z_k) \to 0$ as $k \to \infty$.

To proof the objctive convergence based on the following two inequalities [3]

$$p^{k+1} - p^* \leq -(y^{k+1})^\top r^{k+1} - \rho(B(z^{k+1} - z^k))^\top (-r^{k+1} + B(z^{k+1} - z^*)) \tag{26}$$

$$p^* - p^{k+1} \leq (y^*)^\top r_{k+1} \tag{27}$$

The right-hand side of (26) goes to zero as $k \to \infty$, because $B(z_{k+1} - z^*)$ is bounded and both r_{k+1} and $B(z_{k+1} - z_k)$ go to zero. The right-hand side in (27) goes to zero as $k \to \infty$, as r_k goes to zero. Thus, we have $\lim_{k\to\infty} p_k = p^*$, i.e., objective convergence.

5.3 Stopping Criteria

The residuals of the optimal conditions can be expressed with respect to the current iteration point, i.e., $f(x^k) + g(z^k) - p^*$. As shown in the convergence proof in section.5.2, the following relationship holds

$$f(x^k) + g(z^k) - p^* \leq -(y^k)^\top r^k + (x^k - x^*)^\top s^k. \tag{28}$$

It implies that when the residuals r^k and s^k are small, the objective function converges to the optimal value p^*. As x^* can not be known exactly, one can not use this inequality as the stopping criterion. But if we define that $\|x^k - x^*\|_2 \leq d$, and thus we have

$$f(x^k) + g(z^k) - p^* \leq -(y^k)^\top r^k + d\|s^k\|_2 \leq \|y^k\|_2 \|r^k\|_2 + d\|s^k\|_2. \tag{29}$$

The Equ.29 can be used as an approximate bound on the objective suboptimality (which depends on value d). This implies that a reasonable termination criterion is that the primal and dual residuals has to be small, i.e.,

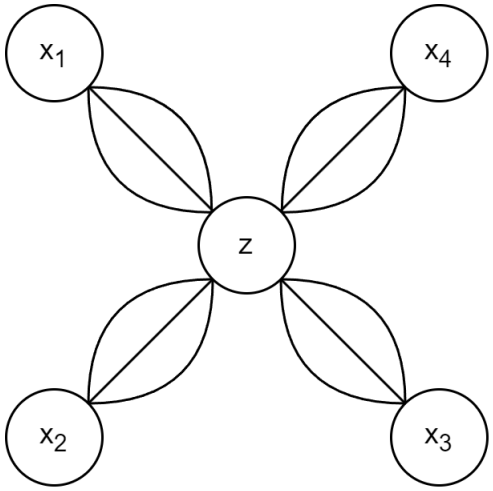

Figure 1: Global Consensus Optimization. There are total $N = 4$ subsystems. Each has $n = 3$ dimension local varibale x_1, x_2, x_3, x_4. Each subsystem is connected to a global node with the global variable $z \in R^3$

.

$$\begin{aligned} \|r_k\|_2 &\leq \varepsilon_{\text{pri}} \\ \|s_k\|_2 &\leq \varepsilon_{\text{dual}} \end{aligned} \tag{30}$$

where $\varepsilon_{\text{pri}} > 0$ and $\varepsilon_{\text{dual}} > 0$ are feasibility tolerances for the primal and dual feasibility conditions of Equ.(16) and Equ.(17), respectively.

6 Problem Formulation

In this project, we focused on ADMM-based methods that can solve distributed optimization problems. More specifically, we focus on consensus-based optimization problems. In the next section, a special consensus optimization problem, global optimization, is presented with how to solve the problem with the ADMM. Later, we present the formulation of the general consensus optimization.

6.1 Global Consensus Optimization

One special form of consensus problem, which is also called global consensus, is as follows

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} \quad & x_i - z = 0, i = 1...N \end{aligned} \tag{31}$$

where $x_i \in \mathbf{R}^n$ is local variables and $z \in \mathbf{R}^n$ is common global variable. The intuition behind problem (31) is as follows: each objective function, f_i, is distributed to N nodes, and each node updates its own variable x_i. The role of the global variable, z, is to learn throughout the nodes collaboratively. As the constraint function in problem (31) suggests, the feasible sets of the problem are all the variables that agree to be equal to each other. Fig. 1 shows one example of global consensus optimization setup. It forms a special bi-parties graph- which is called a star graph where each node is connected to to a central node, but no nodes are connected. In Fig. 1, there are toalt $N = 4$ subsystems, each having its own local variable x_i, $i = 1, 2, 3, 4$. The dimension of both local and global variable is $n = 3$.
The augmeneted Lagrangian of problem (31) is as follows

$$L_p(x1, ...x_N, z, y) = \sum_{i=1}^{N} (f_i(x_i) + y_i^T(x_i - z) + (\rho/2) \|x_i - z\|_2^2) \tag{32}$$

The resulting ADMM algorithm is the following

$$x_i^{k+1} := \arg\min_{x_i} \left(f_i(x_i) + y_i^k T(x_i - z_k) + \frac{\rho}{2} \|x_i - z_k\|_2^2 \right) \tag{33}$$

$$z_{k+1} := \frac{1}{N} \sum_{i=1}^{N} \left(x_i^{k+1} + \frac{1}{\rho} y_i^k \right) \tag{34}$$

$$y_i^{k+1} := y_k^i + \rho \left(x_i^{k+1} - z^{k+1} \right) \tag{35}$$

Equ. (33) and Equ. (35) can be done parallel in the each node i. Equ.(34) behaves like a central collector or the fusion centre.
The ADMM can be simplified more, revealing some important algorithm characters. For instance, the z update in Equ. (34) can be rewritten as

$$z^{k+1} = \bar{x}^{k+1} + (\frac{1}{\rho})\bar{y}^k \tag{36}$$

where $\bar{x}^k = \frac{1}{N}\sum_{i=1}^{N} x_i^k$ is the average of x with respect to all local nodes. Simalrly, $\bar{y}^k = \frac{1}{N}\sum_{i=1}^{N} y_i^k$ is the avaragre dual variable with respect to the all nodes.
Similarly for the y update

$$\bar{y}^{k+1} = \bar{y}^k + \rho(\bar{x}^{k+1} - z_{k+1}) \tag{37}$$

Subsisting the Equ.(36) on Euq.(37) yields that $\bar{y}^{k+1} = 0$. So the dual variable has zero value after the first iteration. From Equ.(34), $\bar{z}^k = \bar{y}^k$. Using the fact of the paragraph mentioned above, the ADDM iteration can be rewritten as follows,

$$\begin{aligned} x_i^{k+1} &:= \text{argmin}_{\ x_i} \left(f_i(x_i) + y_i^{kT}(x_i - \bar{x}^k) + \frac{\rho}{2}\|x_i - \bar{x}^k\|^2 \right) \\ y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - \bar{x}^{k+1}) \end{aligned} \tag{38}$$

The above-mentioned format is ideal for understanding the intuition of the algorithm. The dual variables are separately updated to drive the variables into consensus, and quadratic regularization helps pull the variables toward their average value while still attempting to minimize each local object function f_i. We can interpret consensus ADMM as a method for solving problems in which the objective and constraints are distributed across multiple processors. Each processor only has to handle its own objective and constraint term, plus a quadratic term, which is updated for each iteration. The quadratic terms are updated in such a way that the variables converge to a common value, which is the solution to the full problem.

The main ideas behind the iterations of the ADMM algorithm is listed below

- The central node gather all the x_i^k and average it to $\bar{x}^k$.
- The average $\bar{x}^k$ is distributed to each local node.
- With the average value $\bar{x}^k$, each loacl node updates theirs local variable x^{k+1}. This can be done parallelly, which means that all the nodes can updates their value simultaneously.
- With new updated value x_i^{k+1} and $\bar{x}^{k+1}$, the dual variable y_i^{k+1} is upadted.

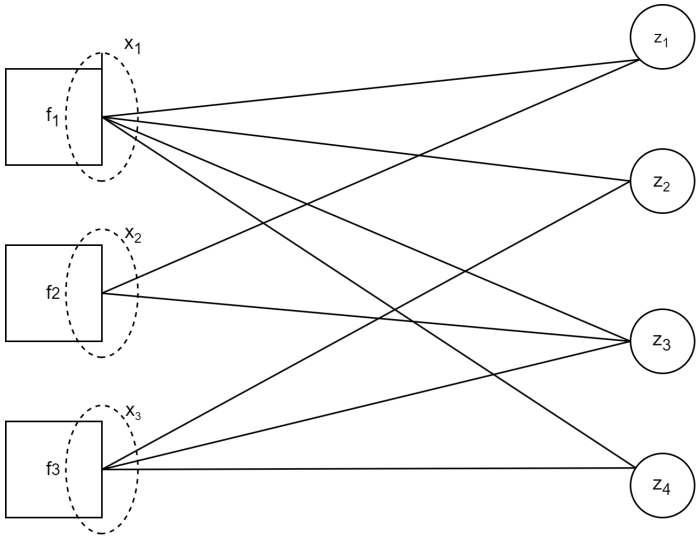

Figure 2: General Consensus Optimization

6.1.1 Optimality Conditions

As discussed in section.5.1, the dual residual and primal residual are used to check how far the objective function is from the optimal value. Using the Equ. (24) and (23), the squared norm residuals of the ADMM can be written as follows:

$$\begin{aligned} \left\|r^k\right\|_2^2 &= \sum_{i=1}^{N} \left\|x_i^k - \bar{x}\right\|_2^2 \\ \left\|s^k\right\|_2^2 &= N\rho^2 \sum_{i=1}^{N} \left\|\bar{x}^k - \bar{x}^{k-1}\right\|_2^2 \end{aligned} \tag{39}$$

It is interesting to analyze the Equ.(39) intuitively. The first of the equation (primal residual) is the standard deviation of the local variable, which reflects the lack of consensus among the local servers.

6.2 General Form of Consensus Optimization

In this section, the consensus optimization is expressed in generalized form. Compared to the global consensus optimization problem, here the local varibale has control for selecting the compmnent from the global vriable.

Here we have local variables $x_i \in R^n$, $i = 1, ..., N$, with the objective $f_1(x_1) + \ + f_N(x_N)$ separable in the x_i. Each local variable consists of a selection of the components of the global variable $z \in R^n$; that is, each component of each local variable corresponds to some global variable component zg. The mapping from local variable indices into global variable index can be written as $g = G(i, j)$, which means that local variable component $(xi)j$ corresponds to global variable component zg. This can be expressed as flowwing format

$$(x_i)_j = z_{G(i,j)} \tag{40}$$

If $G(i, j) = j$ for all i, then each local variable is eqaul to the global variable, and thus consensus reduces to global variable consensus, $xi = z$. Fig.2 shows one example of the general consensus optimization. This forms a bipartite graph where each edge represts the constraint between the local variable constraint and global variable. Here there are $N = 3$ local subsystems, and the global variable dimension $n = 4$. The lacal variable dissensions are as follows: $n_1 = 4$,$n_2 = 2$ and $n_3 = 3$.

7 Implementation and Simulation

7.1 ADMM Applied to LASSO Regression

The ADMM algorithm can be naturally applied to the Least Absolute Shrinkage and Selection Operator (LASSO), whose objective function is as follows.

$$f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1 \tag{41}$$

The LASSO is a convenient choice as a demonstration of the ADMM because the objective function has two terms, an L2 norm, and an L1 norm, which can be naturally separated from each other and performed independently. To prepare the LASSO function for ADMM application, we can recontextualize the LASSO as the sum of separable functions, f(x) and g(z), where we map the least squares term (L2 norm) to f(x) and the penalty term (L1 norm) to g(z).

$$f(x) + g(z) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|z\|_1 \tag{42}$$

Recall that the ADMM general form minimizes the separated objective function f(x) + g(z) subject to some affine constraints. Note that the matrix A in the constraints function is distinct from the input data matrix A in the LASSO function.

$$\begin{aligned} \text{minimize} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned} \tag{43}$$

The LASSO regression problem can now be expressed in ADMM general format as follows.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} \quad & x - z = 0 \end{aligned} \tag{44}$$

A constraint of x = z is imposed to ensure that while the algorithm iterates and approaches convergence, the distance between x and z approaches zero. This makes sense because in the original objective function before separation, they are the same variable, and so they should be roughly equal after separation and convergence. Compared to the general form of the constraints, matrix A is equal to identity, matrix B is equal to minus identity, and c is equal to zero.

With the LASSO problem formulated, the steps for primal variable update and dual variable update for the ADMM algorithm itself can be derived as follows. Note that the function S in the z-minimization step is the soft-thresholding operator, which emerges as a result of finding the gradient of an L1 norm term.

$$\begin{aligned} x^{k+1} &:= (A^T A + \rho I)^{-1}(A^T b + \rho z^k - y^k) \\ z^{k+1} &:= S_{\lambda/\rho}(x^{k+1} + y^k/\rho) \\ y^{k+1} &:= y^k + \rho(x^{k+1} - z^{k+1}) \end{aligned} \tag{45}$$

The primal residual is essentially the difference between variables x and z, as imposed by our equality constraint function, x - z = 0, since we expect the difference between x and z to approach zero as the algorithm searches for convergence. The dual residual is the difference between variable y in the current iteration of the algorithm, and itself in the previous iteration. The primal residual r and dual residual s can be expressed as follows.

$$
\begin{aligned}
r^{k+1} &= x^{k+1} - z^{k+1} \\
s^{k+1} &= \rho(z^{k+1} - z^k)
\end{aligned}
\tag{46}
$$

With the ADMM Lasso problem formally expressed and with the primal and dual updates of the algorithm defined, an experiment is written in Python to demonstrate the convergence of the ADMM. Shown in Fig. 3 is a plot of the primal and dual residuals over 4000 iterations. Note that the algorithm does not stop when stopping criterion is met, as we have chosen to allow the algorithm to run for as long as possible to observe long-term behavior. Instead, we record the iteration number and the point x at which stopping criterion is met and mark it with a red dotted line on the iteration axis. We refer to the objective function evaluated at the point at which the stopping criterion is met as "pstop". We refer to the objective function evaluated at the final point x at max iterations as "pstar".

This ADMM Lasso experiment was conducted on a synthetic dataset matrix A which is 2800x2800 in dimension, and whose elements are randomly generated using a normal distribution. Thus, A is a dense matrix. As can be observed in the plot, the algorithm converges quickly in the first 100 iterations, but then slowly and steadily converges up to iteration 1200 at which point the convergence bottoms out, most likely due to data type precision of the variables.

Compared to other algorithms such as Newton's method, the ADMM algorithm is comparably slow, and with certain input matrices, for example when the dataset is wider than it is tall, may offer less convergence precision. However, for many practical optimization tasks where high precision is not necessary, for example minimizing a loss function in a machine learning task, this shortcoming can be acceptable in favor of the decomposability that the ADMM offers.

To utilize the decomposability of ADMM, another experiment is performed on the same synthetic dataset but with a distributed global consensus implementation. To implement this in Python, the Message Passing Interface (MPI) is employed and allows the distribution of the problem amongst multiple CPU nodes in a High-Performance Computer Cluster (HPC) such as the Amarel cluster operated by Rutgers University. An illustration of the partitioning of the problem is shown as follows.

In the distributed global consensus strategy, the root node, also referred to as the central node, splits the dataset matrix A along its examples and

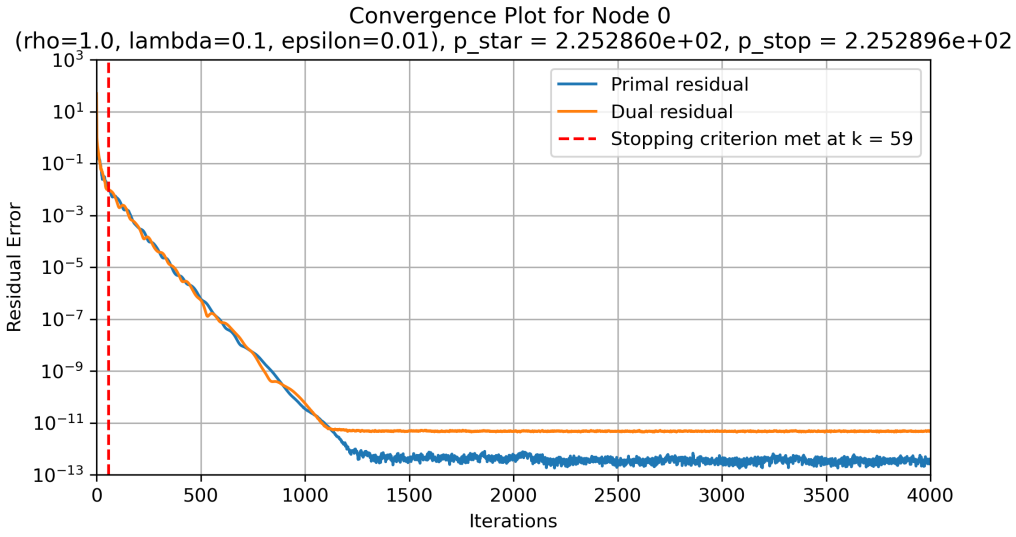

Figure 3: Single Node ADMM Lasso on data matrix A size 2800x2800

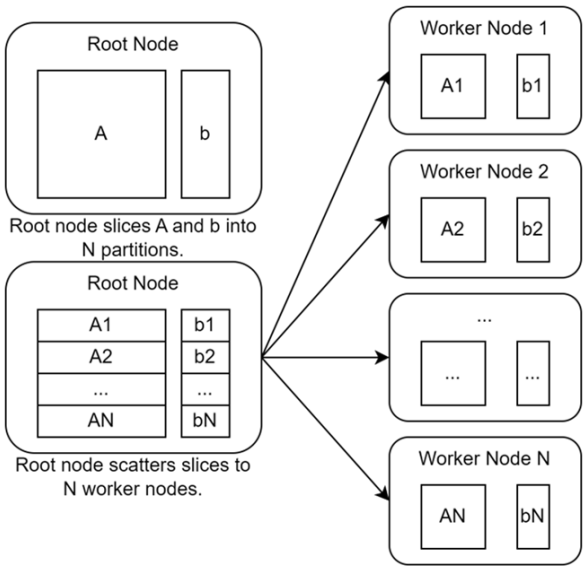

Figure 4: Distributed ADMM Algorithm in MPI Context

scatters the horizontal subblocks to each of the worker nodes. Likewise, the root also divides the response vector b and scatters the subvectors to each worker node. Once this data transfer is complete, each worker has its own local partition of the dataset and no communication of this local partition needs to be communicated with any other node.

At this point, the worker nodes can independently work on their local datasets and update their local variables independently. Shown below is the distributed ADMM global consensus algorithm contextualized by the interactions between the root node and worker nodes.

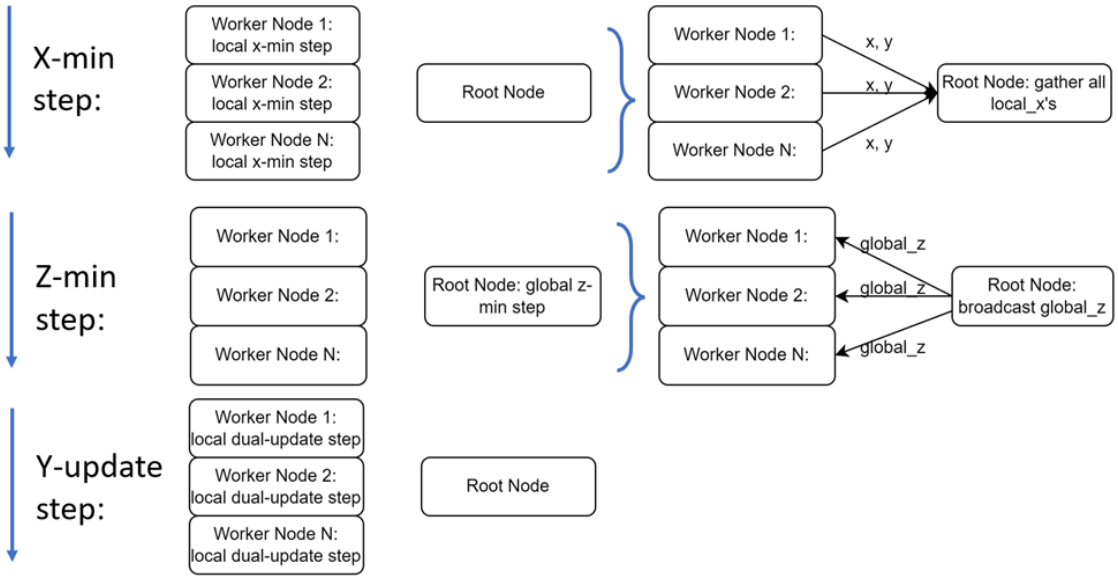

Figure 5: Distributed ADMM primal and dual updates in MPI Context

In the x-minimization step, the worker nodes perform their local x variable update, then altogether, sends the updated x to the root node. The root node gathers up these local x's to perform the global z variable update in the z-minimization step. Then the root node broadcasts the updated global z back to the worker nodes, which then perform their local dual variable update. Then, a stopping criterion is evaluated using the primal and dual residuals, and if the criterion is not met, the algorithm loops again until it does.

When reformulating the ADMM Lasso problem for distributed global consensus, the problem becomes the following, where A_i is the dataset partition

local to each worker node, and the coefficient vector x_i is also local to each worker node.

$$f(x) + g(z) = \frac{1}{2}\|A_i x_i - b_i\|_2^2 + \lambda\|z\|_1 \tag{47}$$

The primal updates x and z and the dual update y are then expressed as follows.

$$\begin{aligned} x_i^{k+1} &:= (A_i^T A_i + \rho I)^{-1}(A_i^T b_i + \rho(z^k - u_i^k)) \\ z^{k+1} &:= S_{\lambda/\rho}(\bar{x}^{k+1} + \bar{u}^k) \\ u_i^{k+1} &:= u_i^k + x_i^{k+1} - z^{k+1}. \end{aligned} \tag{48}$$

With the distributed global consensus version of the ADMM Lasso set up, another experiment is conducted with the same dense square matrix A of dimensions 2800x2800 as in the single node test. In this experiment, the work is split up amongst 10 CPU nodes on the Rutgers Amarel Computer Cluster as described previously.

The dataset is partitioned along examples, meaning that each worker node receives a wide local partition of size 280x2800. Shown below in Fig. 6 are the convergence plots for 2 of the 10 nodes. The convergence plots for the rest of the nodes can be found in the Github repository under the directory "dist-lasso/square-2800x2800/test-2" as well as a text printout containing timing details of the operation.

The convergence plots for the other 8 nodes appear roughly the same, with criterion met at around 2900 iterations and with the pstop value minimizing to around 230. Both the distributed ADMM and non-distributed ADMM method minimizes pstop to roughly the same value, however, the distributed method reaches the stopping point after thousands of iterations, whereas the non-distributed method reaches the stopping point within the first 100 iterations. This result could be a consequence of the "blinding" of each node to the global dataset and thus having less information about the global system to work from, but can also lend itself to a discussion on the tallness or wideness of the dataset and the partitioning strategy of the dataset along examples versus along features and how these properties affect convergence.

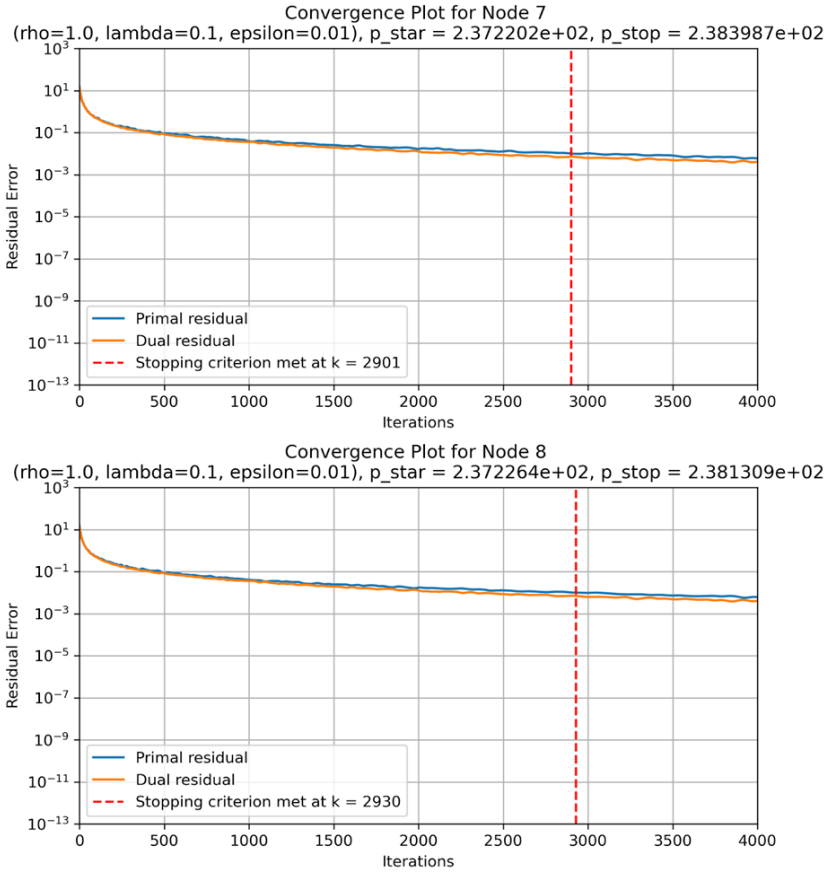

Figure 6: Convergence Plots of Nodes 7 and 8 out of 10

8 Conclusion

In this project, we demonstrated the ADMM algorithm's effectiveness in solving the distributed optimization problem. We showed the generalized ADMM formulation, the optimally condition and convergence analysis, how the values are updated, and the intuition behind such updates. We formulated the ADMM in the global and general consensus optimization problem.

We have also implemented Python realizations of both the single node ADMM method and a distributed global consensus ADMM method and demonstrated their convergence to the same minimum point in the LASSO regression experiments. However, since we had only performed experiments on synthetically generated, square, dense data matrices, future investigations can involve performing LASSO regression on sparse matrices, tall and wide matrices, and very large scale matrices, and observing the effects that these properties have on convergence. Another area of investigation can be comparing the splitting strategy between splitting with examples versus splitting with features, as the partitioning of the dataset will also affect the tallness and wideness of the dataset partitions in the distributed method. Further work could also extend to more complex optimization challenges, fine-tune the performance of the ADMM algorithm under different system architectures, and explore its integration with other machine learning and data processing techniques to leverage the full potential in distributed environment.

References

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. Figueiredo, "An augmented lagrangian approach to the constrained optimization formulation of imaging inverse problems," *IEEE transactions on image processing*, vol. 20, no. 3, pp. 681–695, 2010.

[2] M. S. Almeida and M. Figueiredo, "Deconvolving images with unknown boundaries using the alternating direction method of multipliers," *IEEE Transactions on Image processing*, vol. 22, no. 8, pp. 3074–3086, 2013.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[4] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[5] X. Gong, T. Zhang, C. P. Chen, and Z. Liu, "Research review for broad learning system: Algorithms, theory, and applications," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 8922–8950, 2021.

[6] T. Halsted, O. Shorinwa, J. Yu, and M. Schwager, "A survey of distributed optimization methods for multi-robot systems," *arXiv preprint arXiv:2103.12840*, 2021.

[7] H. Liu, J. Zhang, Q. Wu, H. Xiao, and B. Ai, "Admm based channel estimation for riss aided millimeter wave communications," *IEEE communications letters*, vol. 25, no. 9, pp. 2894–2898, 2021.

[8] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. Jordan, "A general analysis of the convergence of admm," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 343–352. [Online]. Available: https://proceedings.mlr.press/v37/nishihara15.html

[9] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, pp. 1–16, 2016.

[10] C. Shen, T.-H. Chang, K.-Y. Wang, Z. Qiu, and C.-Y. Chi, "Distributed robust multicell coordinated beamforming with imperfect csi: An admm approach," *IEEE Transactions on signal processing*, vol. 60, no. 6, pp. 2988–3003, 2012.

[11] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[12] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, "Multi-agent reinforcement learning via double averaging primal-dual optimization," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[13] W. Zhong, K. Xie, Y. Liu, C. Yang, S. Xie, and Y. Zhang, "Admm empowered distributed computational intelligence for internet of energy," *IEEE Computational Intelligence Magazine*, vol. 14, no. 4, pp. 42–51, 2019.