Towards Quantized Model Parallelism for Graph-Augmented MLPs Based on Gradient-Free ADMM Framework

Junxiang Wang
Joint works with Hongyi Li (Xidian), Zheng Chai (GMU), Yongchao Wang(Xidian), Yue Cheng (GMU), and Liang Zhao (Emory)

Department of Computer Science and Informatics, Emory University

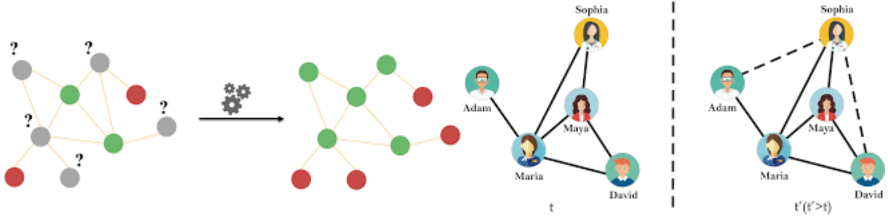
INFORMS Optimization Society Conference(IOS) 2022

Table of Contents

Table of Contents

GNN Introduction

The Graph Neural Network (GNN) models achieve great performance on graph learning tasks such as node classification and link prediction. This is because they handle graph-structured data via aggregating neighbor information.



Node Classification

Link Prediction

Figure: Two applications of GNN models.

GA-MLP

Graph Augmented Multi-Layer Perceptron (GA-MLP) models have recently received fast increasing attention.

- GA-MLP models augment node representations of graphs and feed them into Multi-Layer Perceptron (MLP) models.
- GA-MLP models can be considered as an approximation of Graph Convolutional Network (GCN) models. For example, a two-layer GA-MLP approximates the performance of GCN models on multiple datasets [1].

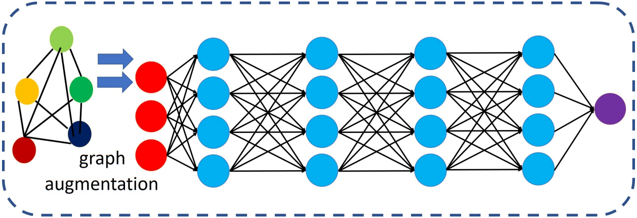


Figure: GA-MLP Models.

ADMM to Train Deep GA-MLP Models in Parallel

Because deeper GA-MLP models have more freedom of expressiveness than shallow ones, the goal of this talk is to devise a scalable training algorithm for training deep GA-MLP models in parallel.
While Stochastic Gradient Descent (SGD) is a state-of-the-art optimizer for deep learning models, we choose ADMM to achieve this goal because of its several advantages:

- The ADMM is a derivative-free algorithm, and can avoid the **gradient vanishing** issue of SGD (i.e. gradient signals vanish when they are transmitted to deep layers).
- The ADMM can address the **backward locking** problem of SGD, which means that the calculation of the gradient in one layer is dependent on its previous layers.

ADMM Introduction

The ADMM is a flexible framework to handle large-scale optimization problems, which splits a complex objective into multiple subproblems, each of which is easy to solve.

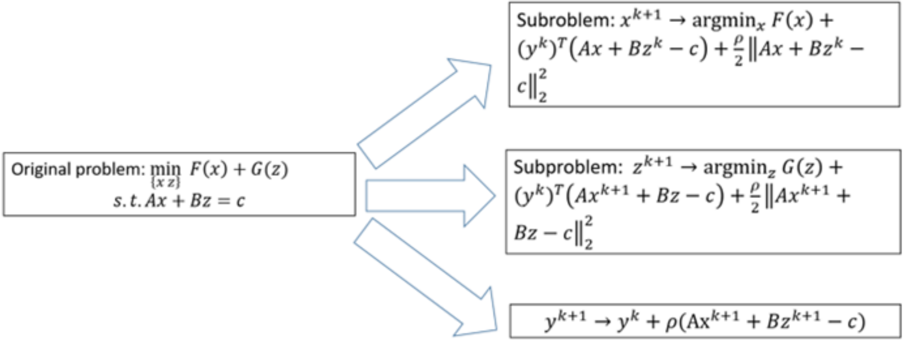


Figure: ADMM Framework.

GA-MLP Training Problem: Feature Augmentation

Consider a graph $G = (V, E)$, where V and E are sets of nodes and edges, respectively, let $\Psi = \{\psi_1(A), \cdots, \psi_K(A)\}$ be a set of (usually multi-hop) operators $\psi_i(A) : \mathbb{R}^{|V|} \rightarrow \mathbb{R}^{|V|} (i = 1, \cdots, K)$ that are functions of the adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$, where $\mathbb{R}^{|V|}$ is the domain. $X_k = H\psi_k(A)$ is the augmentation of node features by the k-hop operator, where $H \in \mathbb{R}^{d \times |V|}$ is a matrix of node features, and d is the dimension of features. Then the input is stacked into

$$X = [X_1; \cdots; X_K].$$

Ψ can be considered as a prepossessing step to augment node features via A, and hence it is predefined. One common choice can be $\Psi = \{I, A, A^2, \cdots, A^{K-1}\}$.

GA-MLP Training Problem: Problem Formulation

The GA-MLP training problem is formulated as follows:

Problem

$$\min_{W_l,b_l,z_l,p_l} R(z_L; y),$$

$$s.t.\ z_l = W_l p_l + b_l,\ \ p_{l+1} = f_l(z_l)(l = 1, \cdots, L-1),$$

Notations	Descriptions
L	Number of layers.
W_l	The weight for the l-th layer.
b_l	The bias for the l-th layer.
z_l	The auxiliary variable of the linear mapping for the l-th layer.
$f_l(z_l)$	The nonlinear activation function for the l-th layer.
p_l	The input for the l-th layer ($p_0 = X$).
q_l	The output for the l-th layer.
X	The node representation of the graph.
A	The adjacency matrix of the graph.
y	The predefined label vector.
$R(z_L, y)$	The risk function for the L-th layer.
n_l	The number of neurons for the l-th layer.

Table: Important Notations

Table of Contents

pdADMM-G: Overview

The proposed parallel graph deep learning Alternating Direction Method of Multipliers (pdADMM-G) splits a GA-MLP model into layerwise components, each of which can be trained via an independent client.

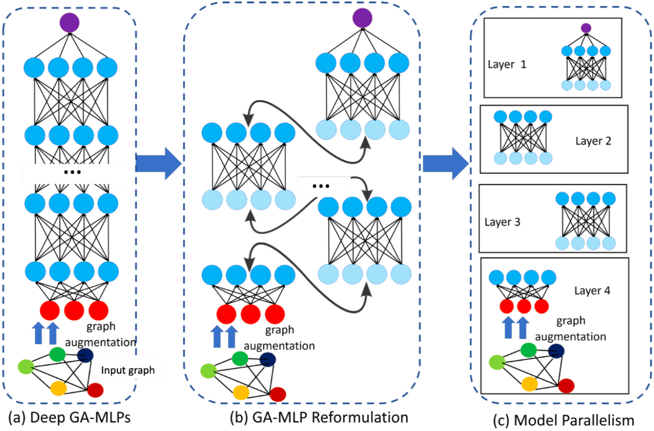


Figure: An Overview of the pdADMM-G.

pdADMM-G: Problem Relaxation

In order to address layer dependency, we relax GA-MLP training problem to the following:

Problem (Relaxed Problem)

$$\min_{\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}} F(\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}) = R(z_L; y)$$
$$+ (\nu/2)(\sum\nolimits_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum\nolimits_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2),$$
$$s.t.\ p_{l+1} = q_l,$$

where $\mathbf{p} = \{p_l\}_{l=1}^{L}$, $\mathbf{W} = \{W_l\}_{l=1}^{L}$, $\mathbf{b} = \{b_l\}_{l=1}^{L}$, $\mathbf{z} = \{z_l\}_{l=1}^{L}$, $\mathbf{q} = \{q_l\}_{l=1}^{L-1}$, and $\nu > 0$ is a tuning parameter. We reduce layer dependency by splitting the output of the l-th layer and the input of the $(l+1)$-th layer into two variables p_{l+1} and q_l, respectively.

pdADMM-G: The Augmented Lagrangian

The Augmented Lagrangian is formulated mathematically as follows:

$$
\begin{aligned}
&L_\rho(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u}) \\
&= F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}) + \sum\nolimits_{l=1}^{L-1} (u_l^T (p_{l+1} - q_l) + (\rho/2)\|p_{l+1} - q_l\|_2^2) \\
&= R(z_L; y) + \phi(p_1, W_1, b_1, z_1) + \sum\nolimits_{l=2}^{L} \phi(p_l, W_l, b_l, z_l, q_{l-1}, u_{l-1}) \\
&+ (\nu/2) \sum\nolimits_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2,
\end{aligned}
$$

where $\phi(p_1, W_1, b_1, z_1) = (\nu/2)\|z_1 - W_1 p_1 - b_1\|_2^2$,
$\phi(p_l, W_l, b_l, z_l, q_{l-1}, u_{l-1}) = (\nu/2)\|z_l - W_l p_l - b_l\|_2^2 + u_{l-1}^T (p_l - q_{l-1})$
$+(\rho/2)\|p_l - q_{l-1}\|_2^2 (l = 2, \cdots, L)$, $u_l (l = 1, \cdots, L-1)$ are dual variables,
$\rho > 0$ is a hyperparameter, and $\mathbf{u} = \{u_l\}_{l=1}^{L-1}$.

pdADMM-G: Pseudocode

Algorithm The pdADMM-G Algorithm

Require: y, $p_1 = X$, ρ, ν.
Ensure: $\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}$.

Initialize $k = 0$.
while $\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k$ not converged **do**
  Update p_l^{k+1} of different l in parallel.
  Update W_l^{k+1} of different l in parallel.
  Update b_l^{k+1} of different l in parallel.
  Update z_l^{k+1} of different l in parallel.
  Update q_l^{k+1} of different l in parallel.
  $r_l^k \leftarrow p_{l+1}^{k+1} - q_l^{k+1} (l = 1, \cdots, L)$ in parallel # Compute residuals.
  Update u_l^{k+1} of different l in parallel.
  $k \leftarrow k + 1$.
end while
Output $\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}$.

Table of Contents

Motivation: Heavy Communication Overheads

In the proposed pdADMM-G algorithm, p_l and q_l are transmitted back and forth among layers (i.e. clients). However, the communication overheads surge for a large-scale graph G.

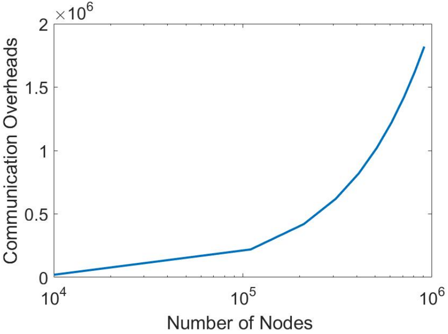


Figure: Heavy communication overheads for millions of nodes.

Solution: Quantization Technique

The quantization technique is commonly utilized to reduce communication overheads by mapping continuous values into a discrete set. To achieve this, p_l is required to fit into a countable set Δ, which is shown as follows:

Problem (Quantized GA-MLP Training Problem)

$$\min_{\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}} F(\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}) = R(z_L; y)$$
$$+ (\nu/2)(\sum\nolimits_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum\nolimits_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2),$$
$$s.t.\ p_{l+1} = q_l,\ \ p_l \in \Delta = \{\delta_1, \cdots, \delta_m\},$$

where $\delta_i (i = 1, \cdots, m) \in \Delta$ are quantized values, which can be integers or low-precision values. $m = |\Delta|$ is the cardinality of Δ.

pdADMM-G-Q: Problem Transformation and Algorithm

To address this problem, we rewrite it into the following form:

$$\min_{\mathbf{p},\mathbf{W},\mathbf{b},\mathbf{z},\mathbf{q}} R(z_L; y) + \sum\nolimits_{l=2}^{L} \mathbb{I}(p_l)$$
$$+ (\nu/2)(\sum\nolimits_{l=1}^{L} \|z_l - W_l p_l - b_l\|_2^2 + \sum\nolimits_{l=1}^{L-1} \|q_l - f_l(z_l)\|_2^2),$$
$$s.t.\ p_{l+1} = q_l,$$

where the indicator function $\mathbb{I}(p_l)$ is defined as follows: $\mathbb{I}(p_l) = 0$ if $p_l \in \Delta$, and $\mathbb{I}(p_l) = +\infty$ if $p_l \notin \Delta$. The augmented Lagrangian of the pdADMM-G-Q is shown as follows:

$$\beta_\rho(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u}) = L_\rho(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u}) + \sum\nolimits_{l=2}^{L} \mathbb{I}(p_l)$$

where L_ρ is the augmented Lagrangian of the pdADMM-G algorithm. The pdADMM-G-Q algorithm follows the same routine of the pdADMM-G algorithm.

Table of Contents

Several Definitions

Definition (Lipschitz Continuity)

A function $g(x)$ is Lipschitz continuous if there exists a constant $D > 0$ such that $\forall x_1, x_2$, the following holds

$$\|g(x_1) - g(x_2)\| \leq D\|x_1 - x_2\|.$$

Definition (Coercivity)

A function $h(x)$ is coerce over the feasible set $\mathscr{F}$ means that $h(x) \to \infty$ if $x \in \mathscr{F}$ and $\|x\| \to \infty$.

Definition (Quantized Stationary Point)

The p_l is a quantized stationary point of of the quantized GA-MLP training problem if there exists $\tau > 0$ such that

$$p_l \in \arg\min_{\delta \in \Delta} \|\delta - (p_l - \nabla_{p_l} F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q})/\tau)\|$$

Convergence Assumptions

Convergence assumptions are so mild that common losses and activations satisfy our assumptions.

Assumption

$f_l(z_l)$ is Lipschitz continuous with coefficient $S > 0$ and $F(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q})$ is coercive.

Common loss functions such as the cross-entropy loss and the least square loss satisfy it.

Assumption

$\partial f_l(z_l)$ is bounded, i.e. there exists $M > 0$ such that $\|\partial f_l(z_l)\| \leq M$.

Common activations such as ReLU, leaky ReLU, sigmoid and tanh satisfy it.

Convergence Properties

Theorem (Convergence of the pdADMM-G algorithm)

If $\rho > \max(4\nu S^2, (\sqrt{17}+1)\nu/2)$*, then for the variables* $(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u})$*, starting from any* $(\mathbf{p}^0, \mathbf{W}^0, \mathbf{b}^0, \mathbf{z}^0, \mathbf{q}^0, \mathbf{u}^0)$*,* $(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ *has at least a limit point* $(\mathbf{p}^*, \mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{q}^*, \mathbf{u}^*)$*, and any limit point is a stationary point. That is,* $0 \in \partial L_\rho(\mathbf{p}^*, \mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{q}^*, \mathbf{u}^*)$*.*

Theorem (Convergence of the pdADMM-G-Q algorithm)

If $\rho > \max(4\nu S^2, (\sqrt{17}+1)\nu/2)$*, then for the variables* $(\mathbf{p}, \mathbf{W}, \mathbf{b}, \mathbf{z}, \mathbf{q}, \mathbf{u})$*, starting from any* $(\mathbf{p}^0, \mathbf{W}^0, \mathbf{b}^0, \mathbf{z}^0, \mathbf{q}^0, \mathbf{u}^0)$*,* $(\mathbf{p}^k, \mathbf{W}^k, \mathbf{b}^k, \mathbf{z}^k, \mathbf{q}^k, \mathbf{u}^k)$ *has at least a limit point* $(\mathbf{p}^*, \mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{q}^*, \mathbf{u}^*)$*, and any limit point* $(\mathbf{W}^*, \mathbf{b}^*, \mathbf{z}^*, \mathbf{q}^*, \mathbf{u}^*)$ *is a stationary point. Moreover, if* τ_l^{k+1} *is bounded, then* $\mathbf{p}^*$ *is a quantized stationary point. where* τ_l^* *is a limit point of* τ_l^k*.*

The convergence rates of the proposed pdADMM-G and pdADMM-G-Q are $o(1/k)$, where k is the number of iterations.

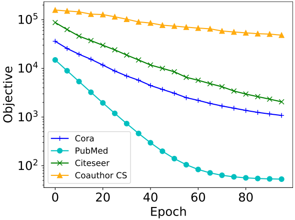
Table of Contents

Datasets

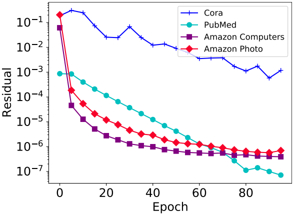
Dataset	Node#	Training Sample#	Test Sample#	Class#	Feature#
Cora	2708	140	1000	7	1433
PubMed	19717	60	1000	3	500
Citeseer	3327	120	1000	6	3703
Amazon Computers	13752	200	1000	10	767
Amazon Photo	7650	160	1000	8	745
Coauthor CS	18333	300	1000	15	6805

Table: Six benchmark datasets

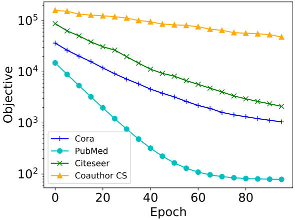
Convergence



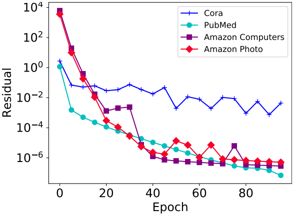
(a). pdADMM-G Objective.



(b). pdADMM-G Residual.



(c). pdADMM-G-Q Objective.



(d). pdADMM-G-Q Residual.

Figure: Convergence curves of the pdADMM-G and the pdADMM-G-Q on four datasets: they both converge.

Speedup

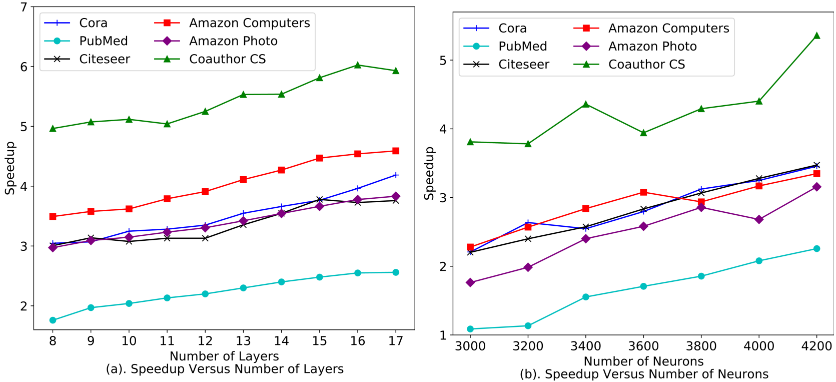
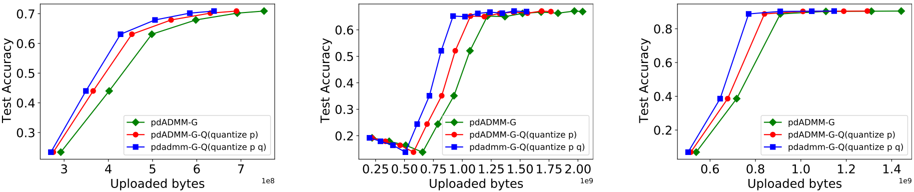


Figure: The relationships between speedup and: (a) the number of layers; (b) the number of neurons in six datasets: the speedup increases linearly with the number of layers and the number of neurons.

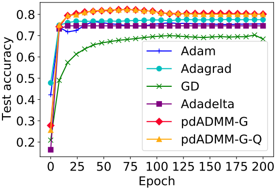
Communication Overheads



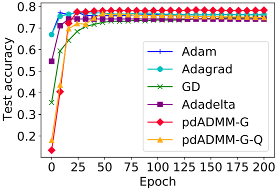
(a). Citeseer (b). Amazon Computers (c). Coauthor CS

Figure: Communication overheads of the pdADMM-G algorithm and the pdADMM-G-Q algorithm in three datasets: the application of the quantization technique on **p** and **q** reduces the communication overheads by 25% without loss of performance.

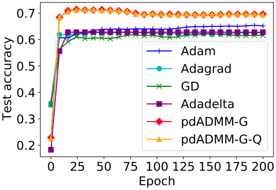
Performance



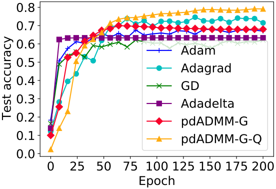
(a). Cora



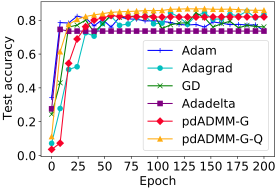
(b). PubMed



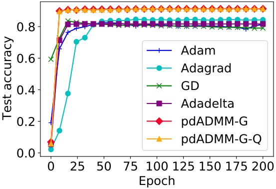
(c). Citeseer



(d). Amazon Computers



(e). Amazon Photo



(f). Coauthor CS

Figure: Test accuracy of all methods: the pdADMM-G algorithm and the pdADMM-G-Q algorithm outperform other comparison methods in six datasets.

Summary

- We propose a novel pdADMM-G framework to train a GA-MLP model in parallel. The extended pdADMM-G-Q algorithm reduces communication costs using the quantization technique.
- The proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm theoretically converge to a stationary point of GA-MLP models with sublinear convergence rates $o(1/k)$.
- Extensive experiments have been conducted to show the convergence, outstanding performance, and the massive speedup of the proposed pdADMM-G algorithm and the pdADMM-G-Q algorithm.

Code Release

pdADMM-G: `https://github.com/xianggebenben/pdADMM-G`

Thank you for your attention!

Reference

F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.