Journal Name

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

Hierarchical Attention Graph Learning with LLMs Enhancement for Molecular Solubility Prediction

Yangxin Fan,[a] Yinghui Wu,[a] Roger H. French,[b] Danny Perez ,[c] Michael G. Taylor,[c] and Ping Yang,[c*]

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

1 Supplementary Information

1.1 Proof

Theorem 1.1. *The complexity of HASolGNN is $O(ek|E|dF^2 + ek|V|F^2)$, where e, k, $|E|$, $|V|$, F, and d denotes the number of training epochs, the number of iterations in AE Block, $max(|E_1|,|E_2|)$, $max(|V_1|,|V_2|)$, number of features per node, and maximum degree of solute graph G_1 and solvent graph G_2.*

Proof. We prove above by first demonstrating that the following two arguments hold: (1) GAT component in the AE Block asymptotically dominates over its counterparts in the ME Block and IE Block; and (2) for all embedding blocks, the GAT component asymptotically dominates over the LSTM component. For (1), AE Block operates directly on molecular graphs $\hat{G}$ while ME Block operators on synthetic graph $\hat{G}^M$ and IG operators on the interaction graph IG. Since the size of molecular graph is $|\hat{G}| = |V| + |E|$, the size of synthetic graph is $|\hat{G}^M| = 2|V| + 1$, and the size of interaction graph $|IG|$ is a constant, both $|\hat{G}^M|$ and $|IG|$ are bounded by $|\hat{G}|$. Hence, GAT component in the AE Block asymptotically dominates over its counterparts in the ME Block and IE Block. For (2), within each embedding block, each GAT is followed by a LSTM layer as illustrated in Fig. **??**. Assume the embedding block operators on computation graph $G = (V,E)$, each LSTM performs gates computation (four gates) which is $O(|V|F^2)$ and element wise operator is $O(|V|F)$. This cost of LSTM is $O(|V|F^2)$. The cost of a single GAT layer is $O(|E|dF^2 + |V|F^2)$[1]. This means that within each embedding block, the GAT component asymptotically dominates over the LSTM component. Besides, one can easily verify that the complexity of the fusion mechanism is $O(F^2)$. The inference complexity of GAT is bounded by one training iteration. Therefore, the training complexity of HASolGNN is $O(ek|E|dF^2 + ek|V|F^2)$ and the inference complexity is $O(k|E|dF^2 + k|V|F^2)$.

1.2 Para-HASolGNN

The design of HASolGNN is parallel-friendly[2]. We introduce a parallel algorithm for HASolGNN training, denoted as Para-HASolGNN, illustrated in Fig. 1, to scale the training of HASolGNN to large-scale solubility dataset. Para-HASolGNN exploits the following three levels of parallelism. We assume the followings: (1) the main coordinator P_0 has the access to all Level I coordinators P_i; and (2) each P_i has information access to all level II workers $P_{i,j}$.

Level I: Model Parallelism. HASolGNN contains two parallel MFGM modules. This presents opportunities for parallelizing the training by distributing solute and solvent MFGM among the processors. In each epoch, Para-HASolGNN executes model parallelism where P_i initializes parallel jobs J_i, $\forall i \in [1,2]$. Within each the level I execution, it forward propagates $MFGM_i$ using Φ_i. At each P_i, $MFGM_i$ backpropagates independently and updates their gradients in parallel after receiving messages from P_0. The output of each module will be assembled and forwarded to $IG \in M$ by the coordinator processor P_0. P_0 calculates global loss in Eqn. **??** and updates IG.

Level II: Data Parallelism. Within the solute or solvent $MFGM$, Para-HASolGNN takes a sequence of molecular graphs as the input. At the level II, Para-HASolGNN initializes the level II parallel jobs $J_i^j, \forall j \in [1, \lceil \frac{T}{L} \rceil]$ and processes the components of the input with a fixed batch size L in parallel. The batch size is determined by available computational resources and the total workload. Besides, Para-HASolGNN optionally exploits the mini-batch data parallelism[3] to achieve even larger speed-up. It splits the information propagation of the large-scale molecular graph into parallelly computed message flow graphs induced by mutually disjoint node batches.

Level III: Pipeline Parallelism. Each MFGM comprises two ME

a Department of Computer Science, Case Western Reserve University, Cleveland, OH, USA, E-mail: yxf451@case.edu; yxw1650@case.edu.
b Department of Material Science, Case Western Reserve University, Cleveland, OH, USA, E-mail: rxf131@case.edu.
c Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM, USA, E-mail: danny_perez@lanl.gov; mgt16@lanl.gov; pyang@lanl.gov.

Algorithm 1 : Para-HASolGNN

Input: A batch of solute and solvent graph pairs $\Phi = \{(G_1^1, G_2^1), \ldots, (G_1^T, G_2^T)\}$, a randomly initialized HASolGNN Model M, batch size L, a coordinator P_0, a set of sub-coordinators P_i, a set of workers $P_{i,j}$, the number of epochs e;
Output: Incrementally trained M upon batch Φ.
for $m = 1$ **to** e **do**
    / executes model parallelism */*
    Para_Model $(J_i(MFGM_i, \Phi_i))$, $\forall i \in [1,2]$;
    P_0.forward(IG);
    P_0 updates $IG \in M$;
return M from P_0;
procedure PARA_MODEL(J_i)
    / executes data parallelism */*
    Para_Data $(J_i^j(MFGM_i^j, \Phi_i^j))$, $\forall j \in [1, \lceil \frac{T}{L} \rceil]$;
    P_i receives M_i from P_0;
    P_i updates $MFGM_i \in M$;
procedure PARA_DATA(J_i^j)
    / executes pipeline parallelism (lines 4-5) */*
    B_1.forward$(\Phi_i^j), B_1 \in MFGM_{i,j}$;
    B_2.forward$(\Phi_i^j), B_2 \in MFGM_{i,j}$;
    $P_{i,j}$ receives $M_{i,j}$ from P_i;
    $P_{i,j}$ updates $MFGM_{i,j}$;

Fig. 1 Para-HASolGNN: Three-level Parallel Training.

Blocks and one AE Block. We adopt an asynchronous macro-pipeline parallelism schema[4] to parallelize the computation of the two independent branches B_1 and B_2. B_1 consists of a AE Block followed by a ME Block while B_2 comprises a single ME Block. In this way, the forward message passing of both B_1 and B_2 are parallelly computed. It eliminates the inter-pipeline synchronization (w/o information loss since the batches from level II are independent of each other).

Complexity Analysis of Para-HASolGNN. The total cost of Para-HASolGNN is $O\left(\frac{ek|E|dF^2+ek|V|F^2}{|P|} + f(\theta)\right)$. Given the input Φ and a model M, we denote the total training cost of HASolGNN using a single worker as $T(\Phi, M)$ which is $O(ek|E|dF^2 + ek|V|F^2)$. We show that for each level of parallelism, the parallel cost is in inverse proportion to the number of the workers $|P|$. We denote the communication overhead among the coordinators and workers as $f(\theta)$ which is $O(\lceil \frac{T}{L} \rceil e)$. Since L and e are hyper-parameters only relevant to the model M, $f(\theta)$ accounts for a communication overhead that is independent of the size of Φ but only dependent of the selection of parameters of M. With the level I and II parallelisms, the communication cost can be further reduced to $O(e)$. Therefore, the total cost of Para-GTrend is $O\left(\frac{ek|E|dF^2+ek|V|F^2}{|P|} + f(\theta)\right)$. $f(\theta)$ is a linear function independent of size of Φ.

1.3 Supplementary Experimental Results

Notes and references

1 M. Besta and T. Hoefler, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.
2 Y. Fan, R. Wieser, L. S. Bruckman, R. H. French and Y. Wu, Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, New York, NY, USA, 2024, p. 4470–4478.
3 D. Zheng, C. Ma, M. Wang, J. Zhou, Q. Su, X. Song, Q. Gan, Z. Zhang and G. Karypis, 2020 IEEE/ACM 10th Workshop on Irregular Applications: Architectures and Algorithms (IA3), 2020, pp. 36–44.
4 D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons and M. Zaharia, Proceedings of the 27th ACM Symposium on Operating Systems Principles, 2019, pp. 1–15.