A Novel Dual-Stream Stochastic Gradient Descent Algorithm for Highly-Efficient Factorization of Large-scale Incomplete Matrices on GPUs Supplementary File

Qicong Hu, *Graduate Student Member, IEEE,* Wen Qin, Hao Wu, *Member, IEEE*, Xin Luo, *Fellow, IEEE*

I. Introduction

THIS is the supplementary file for paper entitled *"A Novel Dual-Stream Stochastic Gradient Descent Algorithm for Highly-Efficient Factorization of Large-scale Incomplete Matrices on GPUs"*. It provides the convergence proof of the DS-SGD algorithm and the memory usage comparison.

II. Convergence Proof

A. Proof for Lemma 1

Considering two LF vectors $\mathbf{m}_{u1}, \mathbf{m}_{u2} \in \mathbb{R}^f$, we apply the second-order Taylor approximation of $\mathcal{L}_{uv}(\mathbf{m}_{u1})$ around $\mathbf{m}_{u2}$:

$$\begin{aligned} &\mathcal{L}_{uv}\left(\mathbf{m}_{u1}\right) = \mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right) + \nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T + \frac{1}{2}\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\nabla^2\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T \\ \Rightarrow\ &\mathcal{L}_{uv}\left(\mathbf{m}_{u1}\right) - \mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right) = \nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T + \frac{1}{2}\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\nabla^2\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T. \end{aligned} \tag{S1}$$

From **Definition 3**, if $\mathcal{L}_{uv}$ is strongly convex, we obtain the inequality:

$$\mathcal{L}_{uv}\left(\mathbf{m}_{u1}\right) - \mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right) \geq \nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T + \frac{\mu}{2}\left\|\mathbf{m}_{u1}-\mathbf{m}_{u2}\right\|^2. \tag{S2}$$

Substituting equation (S1) into (S2) yields:

$$\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\nabla^2\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T \geq \mu\left\|\mathbf{m}_{u1}-\mathbf{m}_{u2}\right\|^2. \tag{S3}$$

Given that $\nabla^2\mathcal{L}_{uv}(\mathbf{m}_{u2}) = \mathbf{n}_v^T\mathbf{n}_v + \lambda I_f$, it follows that:

$$\begin{aligned} &\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\left(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T \geq \mu\left\|\mathbf{m}_{u1}-\mathbf{m}_{u2}\right\|^2 \\ &\Rightarrow \left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\left(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f - \mu I_f\right)\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)^T \geq 0. \end{aligned} \tag{S4}$$

Inequality (S4) is guaranteed provided that $(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f - \mu I_f)$ remains positive semi-definite. This property is ensured when μ equals the smallest eigenvalue of $(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f)$. Consequently, this completes the *proof* of **Lemma 1**.

B. Proof of Lemma 2

Considering two LF vectors $\mathbf{m}_{u1}$ and $\mathbf{m}_{u2}$, we have:

$$\begin{aligned} &\nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u1}\right) - \nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right) = \left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\left(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f\right) \\ &\Rightarrow \left\|\nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u1}\right) - \nabla\mathcal{L}_{uv}\left(\mathbf{m}_{u2}\right)\right\|^2 = \left\|\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\left(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f\right)\right\|^2 \\ &\qquad\qquad \leq \left\|\left(\mathbf{m}_{u1}-\mathbf{m}_{u2}\right)\right\|^2\left\|\left(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f\right)\right\|^2, \end{aligned} \tag{S5}$$

where $\left\|\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f\right\|^2$ represents the greatest eigenvalue of $(\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f)$. Let $L = \left\|\mathbf{n}_v^T\mathbf{n}_v + \lambda I_f\right\|^2$, **Lemma 2** follows.

C. Proof of Theorem 1

Based on the M-oriented learning scheme in (6), we obtain:

$$\begin{aligned}\left\|\mathbf{m}_u^{(t)}-\mathbf{m}_u^*\right\|^2 &= \left\|\mathbf{m}_u^{(t-1)}-\eta\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)-\mathbf{m}_u^*\right\|^2 \\ &= \left\|\mathbf{m}_u^{(t-1)}-\mathbf{m}_u^*\right\|^2 - 2\eta\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)\left(\mathbf{m}_u^{(t-1)}-\mathbf{m}_u^*\right)^T + \eta^2\left\|\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)\right\|^2.\end{aligned} \tag{S6}$$

Through expectation calculation on both sides of (S6):

$$\begin{aligned}\mathbb{E}\left[\left\|\mathbf{m}_u^{(t)}-\mathbf{m}_u^*\right\|^2\right] = \mathbb{E}&\left[\left\|\mathbf{m}_u^{(t-1)}-\mathbf{m}_u^*\right\|^2\right] \\ &- 2\eta\mathbb{E}\left[\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)\left(\mathbf{m}_u^{(t-1)}-\mathbf{m}_u^*\right)^T\right] \\ &+ \eta^2\mathbb{E}\left[\left\|\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)\right\|^2\right].\end{aligned} \tag{S7}$$

According to **Definition 3** and **Lemma 1**, we have:

$$\begin{cases}\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right) \geq \mathcal{L}_{uv}\left(\mathbf{m}_u^{t-1}\right) + \nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{t-1}\right)\left(\mathbf{m}_u^*-\mathbf{m}_u^{t-1}\right)^T + \frac{\mu}{2}\left\|\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right\|^2; \\ \mathcal{L}_{uv}\left(\mathbf{m}_u^{t-1}\right) \geq \mathcal{L}_{uv}\left(\mathbf{m}_u^*\right) + \nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\left(\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right)^T + \frac{\mu}{2}\left\|\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right\|^2.\end{cases} \tag{S8}$$

Joining these inequalities leads to:

$$\left[\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{t-1}\right)-\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\right]\left(\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right)^T \geq \mu\left\|\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right\|^2. \tag{S9}$$

Given that $\mathbf{m}_u^*$ represents the optimal value of $\mathbf{m}_u$, we have:

$$\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right) = 0. \tag{S10}$$

Therefore, (S9) can be rewritten as:

$$\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{t-1}\right)\left(\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right)^T \geq \mu\left\|\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right\|^2. \tag{S11}$$

Substituting (S11) into (S8) gives:

$$\begin{aligned}\mathbb{E}\left[\left\|\mathbf{m}_u^{(t)}-\mathbf{m}_u^*\right\|^2\right] \leq \mathbb{E}&\left[\left\|\mathbf{m}_u^{(t-1)}-\mathbf{m}_u^*\right\|^2\right] \\ &- 2\eta\mu\mathbb{E}\left[\left\|\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right\|^2\right] \\ &+ \eta^2\mathbb{E}\left[\left\|\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)\right\|^2\right].\end{aligned} \tag{S12}$$

Based on **Lemmas 1** and **Lemmas 2**,the subsequent corollary [27] follows:

$$\mathbb{E}\left[\left\|\nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^{(t-1)}\right)\right\|^2\right] \leq G^2. \tag{S13}$$

Substituting (S13) into (S12) yields:

$$\mathbb{E}\left[\left\|\mathbf{m}_u^{(t)}-\mathbf{m}_u^*\right\|^2\right] \leq (1-2\eta\mu)\,\mathbb{E}\left[\left\|\mathbf{m}_u^{t-1}-\mathbf{m}_u^*\right\|^2\right] + \eta^2G^2. \tag{S14}$$

Using inductive reasoning over t, we find:

$$\mathbb{E}\left[\left\|\mathbf{m}_u^{(t)}-\mathbf{m}_u^*\right\|^2\right] \leq (1-2\eta\mu)^{t-1}\mathbb{E}\left[\left\|\mathbf{m}_u^1-\mathbf{m}_u^*\right\|^2\right] + \sum_{j=0}^{t-2}(1-2\eta\mu)^j(\eta G)^2. \tag{S15}$$

Under **Theorem 1**'s constraint $0 < \eta < 1/(2\mu)$, it follows that:

$$\sum_{j=0}^{t-2}(1-2\eta\mu)^j < \sum_{j=0}^{\infty}(1-2\eta\mu)^j = \frac{1}{2\eta\mu}. \tag{S16}$$

Substituting (S16) into (S15) yields:

$$\mathbb{E}\left[\left\|\mathbf{m}_u^{(t)}-\mathbf{m}_u^*\right\|^2\right] \leq (1-2\eta\mu)^{t-1}\mathbb{E}\left[\left\|\mathbf{m}_u^1-\mathbf{m}_u^*\right\|^2\right]+\frac{\eta G^2}{2\mu}. \tag{S17}$$

Based on the quadratic upper bound property of the L-smoothness function [27], we have:

$$\begin{aligned}\mathcal{L}_{uv}\left(\mathbf{m}_u^t\right)-\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right) &\leq \nabla\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\left(\mathbf{m}_u^t-\mathbf{m}_u^*\right)^T+\frac{L}{2}\left\|\mathbf{m}_u^t-\mathbf{m}_u^*\right\|^2 \\ &\leq \frac{L}{2}\left\|\mathbf{m}_u^t-\mathbf{m}_u^*\right\|^2.\end{aligned} \tag{S18}$$

By joining (S18) and (S17), it follows that:

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}_{uv}\left(\mathbf{m}_u^t\right)-\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\right] &\leq \frac{L}{2}\mathbb{E}\left[\left\|\mathbf{m}_u^t-\mathbf{m}_u^*\right\|^2\right] \\ &\leq \frac{L}{2}\left[(1-2\eta\mu)^{t-1}\mathbb{E}\left[\left\|\mathbf{m}_u^1-\mathbf{m}_u^*\right\|^2\right]+\frac{\eta G^2}{2\mu}\right].\end{aligned} \tag{S19}$$

When the iteration count t is sufficiently large (i.e., as $t\to\infty$):

$$\mathbb{E}\left[\mathcal{L}_{uv}\left(\mathbf{m}_u^t\right)-\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\right] \leq \frac{L\eta G^2}{4\mu}. \tag{S20}$$

Let $\Omega(u)$ represents the set of known entries related to node u, then (S20) can be reformulated as:

$$\mathbb{E}\left[\sum_{r_{uv}\in\Lambda(u)}\mathcal{L}_{uv}\left(\mathbf{m}_u^t\right)-\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\right] \leq \frac{|\Lambda(u)|\,L\eta G^2}{4\mu}. \tag{S21}$$

For all $u\in U$, equation (S21) can be extended to:

$$\mathbb{E}\left[\sum_{u\in U}\sum_{r_{uv}\in\Lambda(u)}\mathcal{L}_{uv}\left(\mathbf{m}_u^t\right)-\mathcal{L}_{uv}\left(\mathbf{m}_u^*\right)\right] \leq \frac{|\Lambda|\,L\eta G^2}{4\mu}. \tag{S22}$$

This completes the *proof* of **Theorem 1**.

III. Memory Usage Comparison

TABLE S1
GPU Memory Usage Across Datasets

Algorithm	ML25M	ML32M	Yahoo!Music	Netflix
FAMPSGD	1092MB	1282MB	4226MB	2988MB
MSGD	900MB	1038MB	3274MB	2324MB
CUMF_CCD	974MB	1140MB	3450MB	2436MB
DS-SGD-C	900MB	1038MB	3274MB	2324MB
DS-SGD-D	788MB	888MB	2564MB	1870MB

This table reports the GPU memory usage of DS-SGD-C, DS-SGD-D, and other GPU-based baselines across various datasets. For DS-SGD, we set the batch sizes to 5,000,000 (ML25M), 6,000,000 (ML32M), 30,000,000 (Yahoo!Music), and 20,000,000 (Netflix), such that the number of batches is consistently 4. The results show that DS-SGD-C consumes at most 3274MB of GPU memory, which is well within the device's capacity. In this case, introducing batching is unnecessary, as the cost of data transfer may exceed the compute time of each batch, preventing effective overlap. Furthermore, both DS-SGD-D and DS-SGD-C exhibit the first and second lowest memory usage among all methods, respectively, indicating that DS-SGD-D's batch-wise mechanism is effective in reducing memory footprint.

IV. Related Work

Recent parallel optimization algorithms for the factorization of LIM can be categorized into CPU-based and GPU-based approaches.

Niu *et al.* [1] introduce Hogwild!, a lock-free asynchronous SGD algorithm in which multiple workers randomly sample a entry of the LIM R and update shared parameters without synchronization, tolerating overwrites through sparsity assumptions. Gemulla *et al.* [2] then propose DSGD, which partitions R into a $p\times p$ grid and assigns each worker one block per "stratum" such that no two simultaneously updated blocks share a row or column. However, DSGD still incurs synchronization overhead when strata alternate. Luo *et al.* [3] approach the conflict problem in Hogwild! via parameter decoupling, alternately updating M

and N in parallel, but this scheme introduces explicit barriers when alternating. Pilászy *et al.* [4] introduce a CPU-based CCD algorithm that computes the closed-form solution of a single scalar in the LF matrix in parallel along the node dimension, holding all other parameters constant. CCD++ [5], [6] is an extension of CCD that updates along the feature dimension, achieving faster convergence.

CPU Baseline. FPSGD proposed by Zhuang *et al.* [7], [8] is a widely adopted CPU baseline, which partition R into at least $(p+1) \times (p+1)$ blocks, and each worker dynamically acquires a conflict-free block via a scheduler to achieve asynchronous updates. More recently, APSGD [9] extends ASGD's decoupling idea by exploiting shared-memory broadcasting of parameters, enabling truly asynchronous alternating updates of both factor matrices.

On the GPU side, early work such as GPUSGD [10] aligns the two-level parallelism of CUDA (thread-block and thread) with a two-stage grid partitioning of R, but suffers from severe inter-block synchronization bottlenecks. Tan *et al.* propose CUMF_ALS [11], a CUDA-based implementation of the ALS algorithm on GPUs. However, ALS suffers from high per-epoch computational complexity due to repeated Hermitian matrix operations. CUMF_SGD [12] adopts an optimized GPU kernel and lock-free parallelism via Hogwild!, but suffers from severe overwriting under high GPU parallelism.

GPU Baseline. CUMF_CCD [13] is a GPU-based CCD++ algorithm that leverages loop fusion and tiling to reduce global memory traffic and improve thread load balancing, thereby enhancing the efficiency of LIM factorization. MSGD [14] decouples the dependencies between LF matrices in the SGD-based LIM factorization, enabling their alternate updates across CUDA kernels. Finally, FAMSGD [15] improves upon MSGD by replacing standard gradient updates with fractional-order gradient updates and incorporating an adaError-style adaptive step size, achieving faster and more stable convergence.

REFERENCES

[1] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[2] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 2011, p. 69–77.

[3] X. Luo, H. Liu, G. Gou, Y. Xia, and Q. Zhu, "A parallel matrix factorization based recommender by alternating stochastic gradient decent," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 7, pp. 1403–1412, 2012.

[4] I. Pilászy, D. Zibriczky, and D. Tikk, "Fast als-based matrix factorization for explicit and implicit feedback datasets," in *Proceedings of the Fourth ACM Conference on Recommender Systems*, New York, NY, USA, 2010, p. 71–78.

[5] H.-F. Yu, C.-J. Hsieh, S. Si, and I. Dhillon, "Scalable coordinate descent approaches to parallel matrix factorization for recommender systems," in *2012 IEEE 12th International Conference on Data Mining*, Brussels, Belgium, 2012, pp. 765–774.

[6] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, "Parallel matrix factorization for recommender systems," *Knowledge and Information Systems*, vol. 41, pp. 793–819, 2014.

[7] Y. Zhuang, W.-S. Chin, Y.-C. Juan, and C.-J. Lin, "A fast parallel sgd for matrix factorization in shared memory systems," in *Proceedings of the 7th ACM Conference on Recommender Systems*, New York, NY, USA, 2013, p. 249–256.

[8] W.-S. Chin, Y. Zhuang, Y.-C. Juan, and C.-J. Lin, "A fast parallel stochastic gradient method for matrix factorization in shared memory systems," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 1, 2015.

[9] W. Qin and X. Luo, "Asynchronous parallel fuzzy stochastic gradient descent for high-dimensional incomplete data representation," *IEEE Transactions on Fuzzy Systems*, vol. 32, no. 2, pp. 445–459, 2024.

[10] J. Jin, S. Lai, S. Hu, J. Lin, and X. Lin, "Gpusgd: A gpu-accelerated stochastic gradient descent algorithm for matrix factorization," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 14, pp. 3844–3865, 2016.

[11] W. Tan, L. Cao, and L. Fong, "Faster and cheaper: Parallelizing large-scale matrix factorization on gpus," in *Proceedings of the 25th ACM International Symposium on High-Performance Parallel and Distributed Computing*. New York, NY, USA: Association for Computing Machinery, 2016, p. 219–230.

[12] X. Xie, W. Tan, L. L. Fong, and Y. Liang, "Cumf_sgd: Parallelized stochastic gradient descent for matrix factorization on gpus," in *Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing*, New York, NY, USA, 2017, p. 79–92.

[13] I. Nisa, A. Sukumaran-Rajam, R. Kunchum, and P. Sadayappan, "Parallel ccd++ on gpu for matrix factorization," in *Proceedings of the General Purpose GPUs*, New York, NY, USA, 2017, p. 73–83.

[14] H. Li, K. Li, J. An, and K. Li, "Msgd: A novel matrix factorization approach for large-scale collaborative filtering recommender systems on gpus," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 7, pp. 1530–1544, 2018.

[15] F. Elahi, M. Fazlali, H. T. Malazi, and M. Elahi, "Parallel fractional stochastic gradient descent with adaptive learning for recommender systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 3, pp. 470–483, 2024.