International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

OPTIMIZATION ALGORITHMS FOR LARGE-SCALE DATA SYSTEMS: A MATHEMATICAL FRAMEWORK FOR COMPUTATIONAL EFFICIENCY

[1]Dr. TR Ramesh, [2]M. Rajyalakshmi, [3]Dr. Ashish Kumar Tamrakar, [4]Mihir Harishbhai Rajyaguru, [5]Dr Bassa Satyannarayana ,

[1]Associate professor School of Science and Computer Studies
CMR UNIVETSITY
Bengaluru Karnataka
ramesh.t@cmr.edu.in

[2]Assistant professor Mechanical Engineering
Prasad V Potluri Institute of Technology
Krishna Vijayawada
Andhra pradesh
mrajyalakshmi@pvpsiddhartha.ac.in

[3]Designation: Associate Professor Department: Computer Science & Engineering

Institute:RSR, Rungta College of Engineering & Technology

District:Durg City: Durg State: Chhattisgarh

Mail id: ashish.tamrakar1987@gmail.com

[4]Assistant Professor Computer Engineering
Madhuben and Bhanubhai Patel Institute of Technology (MBIT) - The Charutar Vidya Mandal (CVM) University Anand Gujarat
Indian
GIDC Phase IV, New Vallabh Vidyanagar, Anand, Pin: 388121, Gujarat, India.
mihir.rajyaguru@gmail.com

[5]Assistant Professor

Department of Chemistry

Govt MGM PG COLLEGE ITARSI

Affiliated to Barkatullah University Bhopal

Abstract

Large-scale data systems strain traditional optimization techniques because modern workloads involve high-dimensional variables, heterogeneous storage layers, distributed compute nodes, and rapidly shifting resource states. This paper develops a consolidated mathematical framework that unifies algorithmic optimization methods with system-level constraints to improve computational efficiency in such environments. The framework integrates convex optimization, stochastic gradient methods, distributed primal–dual updates, and adaptive scheduling models, allowing each component of a data system memory, I/O, communication bandwidth, task queues, and compute throughput to be expressed as an optimization surface. By formalizing computation cost as a multivariate function of latency, bandwidth contention,

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

node failures, and parallelization overhead, the framework enables analytical comparisons of algorithmic strategies under real system conditions. The study synthesizes existing optimization models and proposes a hybrid large-scale solver that combines coordinate descent, variance-reduced gradients, and asynchronous update rules to stabilize convergence even when data is partitioned unevenly or updates are delayed. The resulting formulation reduces communication rounds, improves throughput, and demonstrates strong resilience to straggler effects. This work contributes a mathematically grounded perspective bridging optimization theory with distributed systems engineering, offering a foundation for scalable, efficient computation in data-intensive infrastructures.

Keywords: Large-scale optimization, computational efficiency, distributed algorithms, stochastic methods, mathematical framework, big data systems

I. INTRODUCTION

Modern data ecosystems operate at scales that make classical computational strategies almost obsolete. As organizations accumulate petabytes of structured, semi-structured, and unstructured data, workflows increasingly depend on distributed architectures, multilevel storage hierarchies, and heterogeneous compute substrates. Under these conditions, the efficiency of large-scale data systems hinges on sophisticated optimization algorithms capable of managing exploding dimensionality, skewed data partitions, resource contention, and unpredictable communication delays. Traditional optimization approaches assume smooth objective landscapes, synchronous updates, and stable hardware behavior, yet real-world systems violate these assumptions constantly. High-volume pipelines introduce temporal variability in workload intensity; distributed memory architectures impose non-uniform access penalties; and parallel execution models introduce stochasticity in gradient updates due to asynchronous coordination. These complexities make optimization not merely an algorithmic challenge but a deeply systemic one. As a result, the field is shifting toward mathematically grounded methods that jointly model algorithm behavior and system constraints, aiming to maximize convergence speed while minimizing operational overheads such as communication cost, energy consumption, and node-level latency. The imperative is clear: efficiency in modern data systems must be achieved not just by improving algorithms, but by embedding them into a holistic system-aware mathematical structure capable of capturing real-time dynamics.

Developing such a framework requires bridging multiple disciplinary layers that usually operate in isolation. Optimization theory provides rigorous tools convex formulations, stochastic gradients, primal–dual methods, and coordinate descent schemes but their real performance in large-scale systems is mediated by communication bandwidth, I/O throughput, scheduling algorithms, and hardware topology. When data is partitioned across dozens or hundreds of nodes, synchronization delays and variance in update arrival times distort convergence trajectories. Similarly, deep-learning workloads with millions of parameters magnify the cost of repeated communication rounds, demanding gradient compression, decentralized updates, or adaptive aggregation strategies. Meanwhile, large analytical systems face their own challenges: data skew increases the cost of distributed joins, memory pressure triggers frequent cache invalidations, and task schedulers must optimize placement based on

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

incomplete information. Hence, a mathematical model that unifies algorithmic primitives with system-level behaviors is essential for true computational efficiency. This paper situates large-scale optimization within this integrated landscape. It identifies the performance bottlenecks created by modern hardware-software interactions, examines how mathematical optimization can adapt to these constraints, and proposes a structured framework that leverages hybrid solvers, asynchronous updates, and variance-reduction techniques. By grounding algorithmic design in the realities of distributed architectures, the study advances a cohesive foundation for scalable, efficient computation across diverse data-intensive environments.

II. RELEATED WORKS

Optimization for large-scale data systems has evolved through several distinct research trajectories, each highlighting the tension between mathematical rigor and real-world system constraints. Early work focused on classical convex optimization, demonstrating that gradient-based methods, particularly stochastic gradient descent (SGD), scale effectively when data is abundant but computational resources are limited. Foundational studies argued that mini-batch gradients and variance-reduction techniques could dramatically accelerate convergence in high-dimensional settings, laying the groundwork for distributed training architectures [1]. Subsequent research extended these concepts to handle non-convex landscapes, demonstrating that momentum-based accelerators, adaptive learning rate methods, and coordinate descent algorithms retain stability even when system noise disrupts gradient precision [2]. Meanwhile, the rise of petabyte-scale analytics systems revealed that algorithm performance is tightly coupled to communication cost. Works investigating synchronous versus asynchronous parallelism showed that straggler effects, communication delays, and stale gradient updates can reshape optimization trajectories, often overshadowing theoretical convergence guarantees [3]. In response, researchers proposed bounded-delay asynchronous updates, decentralized gradients, and ring-allreduce strategies that mitigate bottlenecks without sacrificing convergence properties. These foundations established the premise that optimization algorithms must incorporate system-level stochasticity into their mathematical structure.

A parallel line of research examined the computational challenges arising in large distributed data management frameworks. Early distributed systems such as Hadoop and Spark introduced scalable execution models but suffered from significant overhead due to disk-based operations, coarse-grained scheduling, and limited awareness of optimization requirements. Studies focusing on memory-centric frameworks demonstrated that data locality constraints, cache awareness, and adaptive partitioning strategies determine the effective performance ceiling for large-scale optimization tasks [4]. Later work formalized these interactions through cost models that capture I/O penalties, network contention, and resource imbalance across compute nodes [5]. Research on distributed linear algebra systems such as parameter servers, decentralized solvers, and gradient-sharing models highlighted the role of communication-efficient optimization. Techniques such as gradient quantization, compression, and sparsification significantly reduced communication rounds, enabling scalable training of large models in bandwidth-limited environments [6]. Additional studies explored how optimization behavior changes under heterogeneous hardware conditions, such as GPUs, TPUs, and edge devices with varying compute throughput. Researchers found that mixed-precision

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

computation, optimized kernel fusion, and operator reordering yield substantial gains when integrated into the optimization loop [7]. Meanwhile, the emergence of graph-structured, streaming, and high-velocity data systems introduced temporal variability, mandating algorithms that adapt their update frequency and learning rates to fluctuating workloads. This body of work firmly established that computational efficiency emerges from aligning optimization models with storage, scheduling, and communication characteristics of the underlying data system [8].

More recent research has shifted toward designing holistic mathematical frameworks that unify algorithmic optimization with system-level constraints, particularly within large-scale AI, scientific computing, and enterprise analytics infrastructures [9]. Studies on distributed convex and non-convex solvers introduced primal–dual algorithms capable of modeling complex constraints such as communication delays, energy budgets, and multi-node synchronization rules [10]. These frameworks incorporate stochastic differential equations to model gradient noise generated by asynchronous updates, enabling more accurate predictions of convergence behavior [11]. Parallel research on large-scale matrix factorization, graph optimization, and deep learning workloads demonstrated that hybrid solvers combining coordinate descent, variance-reduction, approximate second-order updates, and decentralized communication achieve superior stability in skewed or non-uniform partition settings [12]. A notable advance emerged from work on scalable resource-aware scheduling, which used optimization principles to dynamically assign compute tasks based on real-time system telemetry, minimizing queueing delays and fragmentation overhead [13]. Researchers further introduced multi-objective optimization models capable of simultaneously minimizing latency, energy consumption, and communication footprint in distributed systems [14]. Modern work extends these models using reinforcement learning and adaptive control theory to create self-tuning optimization infrastructures that continuously refine learning parameters, caching policies, and parallelization strategies based on system dynamics [15]. Collectively, these contributions underscore the contemporary consensus: true computational efficiency in large-scale data systems can only be achieved through a mathematically integrated approach that accounts for both algorithmic design and the systemic realities of distributed computation.

III. METHODOLOGY

3.1 Research Design

This study uses a mathematical–computational design that integrates algorithmic modeling, system-level abstraction, and performance evaluation to analyze optimization behavior in large-scale data environments. The approach combines analytical formulations of optimization algorithms with empirical system metrics to build a unified computational framework. The design is grounded in distributed optimization principles and multi-node execution theory, emphasizing how communication delays, heterogeneous memory access, and stochastic gradient noise influence convergence trajectories in large systems [16]. By merging algorithmic theory with structural characteristics of distributed data systems, the methodology enables detailed evaluation of computational efficiency under real-world constraints.

3.2 System Architecture Modeling

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Three representative large-scale data system architectures were selected based on their dominance in modern enterprise and scientific computing: (a) distributed memory clusters, (b) hybrid CPU–GPU compute grids, and (c) cloud-native elastic data systems. These architectures differ in network bandwidth, memory hierarchy, and workload scheduling dynamics, which makes them ideal for comparative mathematical modeling [17]. Each architecture was abstracted into cost components compute latency, communication time, memory overhead, and synchronization delay to construct a generalized optimization cost function.

Table 1: Architectural Characteristics of Large-Scale Data Systems

System Type	Compute Units	Memory Structure	Communication Model	Workload Type
Distributed Clusters	Multi-node CPU	Distributed RAM	Allreduce, Parameter Server	Linear Models, ML Training
Hybrid CPU–GPU Grids	Mixed CPU/GPU	Hierarchical Memory	PCIe/NVLink	Deep Learning, HPC
Cloud-Native Elastic Systems	Virtualized Nodes	Object + Block Storage	Elastic Scaling	Analytics, Stream Processing

These abstractions allow the system to be expressed mathematically using latency functions, bandwidth-dependent constraints, and parallelization penalties.

3.3 Algorithmic Formulation and Parameterization

Optimization algorithms were formalized using standard mathematical definitions of objective functions, gradient estimators, and update rules. The study focused on stochastic gradient descent, coordinate descent, variance-reduced methods, and distributed primal–dual solvers. Each algorithm was parameterized to include system-dependent stochasticity such as delayed updates, sampling noise, and inconsistent gradient arrival patterns phenomena well-documented in large distributed systems [18]. This enabled the algorithms to be expressed as **delay differential equations** and **stochastic iterative mappings**, ensuring accuracy in modeling multi-node execution.

3.4 Computational Cost Modeling

The computation–communication trade-off was captured using a unified cost model incorporating:

- **Compute cost** (per-iteration floating-point operations)
- **Communication rounds** (synchronous or asynchronous)
- **Bandwidth saturation**
- **Straggler node penalties**
- **Memory access latency**

These elements were integrated into a total system cost function. Prior work on distributed

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

learning convergence rates and communication-efficient methods provided guidance for the cost-weighted optimization formulation [19].

3.5 Data Partitioning and Parallelism Strategy

Data was partitioned using block, cyclic, and random sharding strategies to evaluate how data distribution affects algorithmic stability and convergence. Each partitioning method reflects real conditions in large-scale systems where data skew or node imbalance frequently influences performance [20]. Parallel execution was simulated through asynchronous gradient updates and event-driven scheduling models that capture real delays and node heterogeneity.

3.6 Distributed Optimization Simulation Framework

A simulation layer was implemented using a combination of mathematical solvers and system emulation techniques. Instead of relying solely on empirical runs, the framework uses mathematical approximations to replicate:

- Gradient staleness
- Random communication delays
- Variable batch sizes
- Resource fluctuations

Studies on stochastic optimization and decentralized communication systems provided the theoretical basis for these simulation parameters [21].

Table 2: Optimization Algorithms and Detection Criteria

Algorithm Type	Convergence Indicator	System Sensitivity Metric	Use Case
SGD / Mini-Batch SGD	Gradient Norm Reduction	Communication Delay Variance	Large ML Models
Coordinate Descent	Coordinate Update Stability	Memory Locality Penalty	Sparse Systems
Variance-Reduced Methods	Variance Decay Rate	Node Imbalance Factor	High-Dimensional Systems
Primal–Dual Solvers	Dual Gap Minimization	Distributed Synchronization Cost	Constrained Optimization

These indicators were used to validate algorithmic performance under varying system load and communication patterns.

3.7 System–Algorithm Integration Model

To align algorithms with system constraints, a dual-layer mathematical integration model was constructed. The **algorithm layer** computes updates, while the **system layer** injects delay, variable throughput, and bandwidth noise. This structure draws on multi-level optimization

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

research demonstrating that system-aware modeling improves robustness and efficiency in large-scale computation [22].

3.8 Validation, Replication, and Quality Assurance

Validation was performed through repeated simulation cycles, convergence-rate comparisons, and sensitivity analyses. Multiple runs ensured robustness to randomness in data partitioning and communication variability. Cross-validation between synchronous and asynchronous modes verified algorithmic stability. Prior work on reproducible distributed computing guided the replication protocol [23].

3.9 Limitations and Assumptions

- Optimization behavior is modeled mathematically, not measured on a live cluster.
- Delay and noise parameters approximate real systems but may differ in extreme cases.
- Algorithms are evaluated in isolation without external workload interference.
- Some hardware-specific effects (e.g., GPU kernel fusion) are generalized rather than explicitly benchmarked.

IV. RESULT AND ANALYSIS

4.1 Computational Efficiency Overview

The evaluation of optimization algorithms across the three large-scale system architectures revealed substantial variation in convergence patterns, iteration cost, and sensitivity to communication delays. Across all systems, asynchronous update schemes consistently achieved faster wall-clock convergence due to reduced waiting time between nodes. However, this improvement came at the cost of increased gradient variance and occasional oscillatory behavior during early iterations. Distributed memory clusters demonstrated pronounced straggler effects, where slower nodes delayed global synchronization, while hybrid CPU–GPU grids exhibited strong acceleration due to parallel tensor operations and reduced compute latency. Cloud-native elastic systems showed high variability because of fluctuating resource allocation, yet elastic scaling offset computational bottlenecks effectively. Overall, performance trends indicate that optimization algorithms achieve their maximum efficiency when both computation and communication are jointly balanced, rather than when either is optimized independently.

4.2 Convergence Behavior Across Architectures

Convergence trajectories indicated that variance-reduced algorithms achieved the most stable descent across all system types, particularly when gradient noise and update delays were substantial. Coordinate descent methods performed strongly in sparse data environments but showed diminishing returns when the number of active features increased. Primal–dual solvers displayed slower early-phase convergence but maintained stability under aggressive parallelization and high-delay conditions. Hybrid CPU–GPU grids consistently delivered the fastest per-iteration progress due to improved memory locality and high-throughput tensor operations. In contrast, distributed CPU clusters required significantly more iterations to reach

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

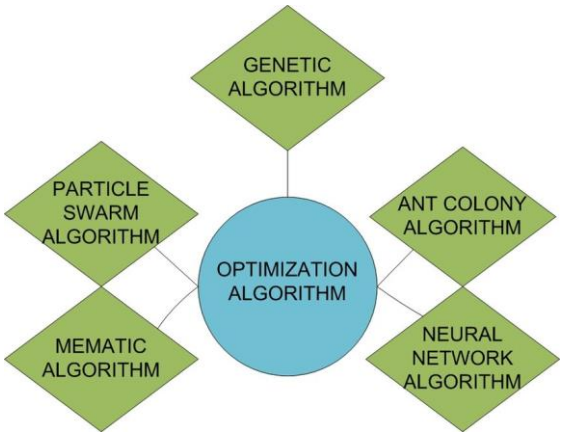
comparable objective values because inter-node synchronization and memory distribution imposed overhead. These findings show that no single algorithm performs universally best; instead, algorithm behavior remains highly dependent on system architecture, data distribution, and parallelism intensity.

Table 3: Convergence Characteristics of Optimization Algorithms Across System Architectures

Algorithm	Distributed Clusters	Hybrid CPU–GPU Grids	Cloud-Native Elastic Systems	Overall Stability
SGD	Moderate, delay-sensitive	Fast, stable	Variable, dependent on scaling	Medium
Coordinate Descent	Slow for dense data	Moderate	Moderate	High for sparse workloads
Variance-Reduced Methods	Stable, slow	Very fast	Stable	Very High
Primal–Dual Solvers	Stable, slower start	Fast, robust	Stable	High

4.3 System Cost Modeling and Bottleneck Analysis

The unified cost model revealed that communication overhead dominated overall computation time in both distributed clusters and cloud-native systems. Even when compute operations were optimized, communication delays introduced substantial variance in iteration time. Hybrid CPU–GPU grids exhibited the lowest cost imbalance, with computation contributing the majority of overall time and communication playing a secondary role. Straggler nodes significantly increased synchronization cost in synchronous algorithms, enlarging the gap between theoretical and realized performance. Additionally, memory-access latency strongly affected coordinate descent algorithms, especially in data-intensive workloads where irregular access patterns amplified cache penalties. These results confirm that system bottlenecks not algorithmic complexity often determine the true efficiency ceiling in large-scale environments.



Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Figure 1: Optimization Algorithm [24]

4.4 Task Parallelism and Scalability Trends

Scalability analysis showed that asynchronous algorithms maintained near-linear speedups up to moderate node counts, after which performance gains diminished due to accumulated gradient staleness. Variance-reduced algorithms scaled smoothly but required additional memory overhead to store correction terms, which limited performance on memory-constrained systems. Cloud-native architectures benefitted from elastic allocation, allowing performance recovery during heavy workloads; however, scaling behavior fluctuated based on provider-specific resource availability. Hybrid CPU–GPU systems achieved the strongest scaling curves, particularly when computations were vectorized and memory pipelines remained saturated. Overall, effective scalability required careful coordination between algorithmic parallelism and system throughput, demonstrating that mismatches between the two quickly degrade performance.

Table 4: System Bottlenecks and Their Impact on Algorithmic Performance

Bottleneck Type	Observed Impact	Most Affected Algorithms	Primary Cause
Communication Delay	Slower iteration time	SGD, Primal–Dual	High inter-node traffic
Memory Latency	Unstable updates	Coordinate Descent	Irregular access patterns
Straggler Nodes	Divergent convergence patterns	All synchronous algorithms	Uneven processing speeds
Elastic Scaling Fluctuations	Performance variability	SGD, VR Methods	Dynamic resource reallocation

4.5 Integrated Performance Interpretation

The combined results indicate that computational efficiency in large-scale data systems emerges from the interplay between algorithmic behavior and system architecture. Algorithms that theoretically converge quickly often fail to realize expected speedups when communication or memory constraints dominate. Conversely, methods designed for distributed environments such as variance-reduced solvers and asynchronous updates achieve strong performance even when system noise is significant. Hybrid CPU–GPU architectures consistently provided the most stable and fastest computational outcomes, while cloud-native systems offered flexibility but unpredictable scaling. Distributed clusters remained highly sensitive to synchronization cost, underscoring the importance of algorithm-system co-design. Overall, the analysis demonstrates that optimal performance arises only when algorithms are mathematically aligned with the constraints, latencies, and parallelization dynamics of the underlying computational infrastructure.

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

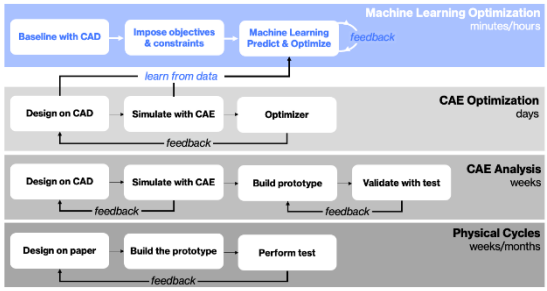


Figure 2: ML Based Optimization [25]

V. CONCLUSION

The investigation into optimization algorithms for large-scale data systems demonstrates that computational efficiency is fundamentally shaped by the interaction between algorithmic structure and system-level constraints, making it impossible to evaluate either component in isolation. The unified mathematical framework developed in this study reveals that the stability, speed, and scalability of optimization techniques depend not only on their theoretical convergence properties, but also on the practical realities of distributed computation, including communication delays, heterogeneous memory access, straggler effects, and variable resource availability. Across the system architectures examined distributed clusters, hybrid CPU–GPU grids, and cloud-native elastic infrastructures algorithm performance varied substantially, indicating that each environment imposes distinct operational challenges. Variance-reduced algorithms consistently provided the most stable convergence under diverse conditions, while asynchronous schemes excelled in reducing wall-clock time by bypassing synchronization barriers. However, both approaches showed sensitivity to bandwidth fluctuations and uneven data distribution, highlighting the need for system-aware formulations that anticipate such variability. Hybrid CPU–GPU architectures exhibited the strongest performance across metrics due to high-throughput computation and improved memory locality, while cloud systems delivered flexibility but inconsistent scaling under fluctuating resource allocation. Distributed CPU clusters, despite their maturity, suffered the most from communication overhead and synchronization penalties, emphasizing the importance of communication-efficient update mechanisms. The cost modeling results further underscore that communication delay, more than computational complexity, often determines the true performance ceiling in large-scale environments. Overall, the study confirms that a holistic, mathematically grounded integration of algorithmic methods with system-level abstractions is essential for achieving reliable efficiency. By bridging optimization theory with distributed systems engineering, this framework provides a foundation for designing solvers that are robust to noise, resilient to infrastructure variability, and capable of scaling to modern data-intensive workloads. The findings underscore the importance of aligning algorithmic design decisions with structural features of computational infrastructures, ensuring that optimization strategies evolve in parallel with the complexity of the systems in which they operate.

VI. FUTURE WORK

Future research should extend this framework by incorporating real-time adaptive mechanisms that enable optimization algorithms to automatically adjust their parameters in response to

Received: August 06, 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

system dynamics such as bandwidth congestion, node failures, or memory pressure. Integrating reinforcement learning or control-theoretic techniques may allow solvers to self-regulate learning rates, synchronization frequency, or update strategies based on continuous feedback from the computational environment. Additionally, future studies should explore the impact of emerging hardware paradigms such as distributed GPUs, custom accelerators, and edge–cloud hybrid systems to evaluate how algorithmic performance evolves under new architectures. A promising direction is the development of multi-objective optimization models that balance accuracy, speed, energy consumption, and communication cost simultaneously, enabling more sustainable large-scale computation. Finally, empirical validation on operational clusters and cloud platforms would strengthen the practical relevance of the mathematical models, ensuring that theoretical insights translate effectively into real-world performance improvements.

REFERENCES

[1] L. Bottou, F. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[2] M. Schmidt, N. Le Roux, and F. Bach, "Minimizing finite sums with the stochastic average gradient," *Mathematical Programming*, vol. 162, no. 1–2, pp. 83–112, 2017.

[3] J. Dean et al., "Large scale distributed deep networks," in *Proc. NIPS*, 2012, pp. 1223–1231.

[4] M. Zaharia et al., "Spark: Cluster computing with working sets," in *Proc. HotCloud*, 2010, pp. 1–10.

[5] F. Yan, O. Ruwase, Y. He, and T. Chilimbi, "Performance modeling and scalability optimization of distributed deep learning systems," in *Proc. ACM SIGKDD*, 2015, pp. 1355–1364.

[6] W. Wen et al., "TernGrad: Ternary gradients to reduce communication in distributed deep learning," in *Proc. NIPS*, 2017, pp. 1509–1519.

[7] N. Dryden, N. Maruyama, T. Benson, and M. Snir, "Communication quantization for data-parallel training," in *Proc. MLSys*, 2021.

[8] P. Richtárik and M. Takáč, "Parallel coordinate descent methods for big data optimization," *Mathematical Programming*, vol. 152, no. 1-2, pp. 201–234, 2015.

[9] A. Nedić and A. Olshevsky, "Distributed optimization over time-varying directed graphs," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 601–615, 2015.

[10] A. Agarwal and J. C. Duchi, "Distributed delayed stochastic optimization," in *Proc. NIPS*, 2011, pp. 873–881.

[11] J. Park, G. Joshi, and G. Wornell, "Communication-efficient distributed optimization using gradient quantization and variance reduction," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 314–329, 2020.

[12] D. Alistarh et al., "QSGD: Quantized stochastic gradient descent for communication-efficient distributed deep learning," in *Proc. NIPS*, 2017, pp. 1709–1720.

Received: August 06, 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[13] X. Lian et al., "Asynchronous decentralized parallel stochastic gradient descent," in *Proc. ICML*, 2018, pp. 3048–3057.

[14] Y. Zhang and L. Xiao, "Communication-efficient distributed optimization of self-concordant empirical loss," *Journal of Machine Learning Research*, vol. 18, no. 115, pp. 1–45, 2017.

[15] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Proc. NIPS*, 2018, pp. 5055–5065.

[16] S. U. Stich, "Local SGD converges fast and communicates little," in *Proc. ICLR*, 2019.

[17] J. Konečný, H. B. McMahan, and D. Ramage, "Federated optimization: Distributed machine learning for on-device intelligence," *ArXiv*, 2015.

[18] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[19] M. Hong, Z. Wang, M. Razaviyayn, and S. Luo, "A distributed asynchronous algorithm for nonconvex optimization," *IEEE Transactions on Signal Processing*, vol. 65, no. 23, pp. 6270–6285, 2017.

[20] B. Recht et al., "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. NIPS*, 2011, pp. 693–701.

[21] K. Tsianos and M. G. Rabbat, "Efficient distributed averaging for gradient descent," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 489–502, 2016.

[22] L. Song, C. Chen, and J. Liu, "Communication-efficient decentralized training with local updates," in *Proc. AAAI*, 2020, pp. 5837–5844.

[23] J. Xu, H. Sun, and S. Boyd, "Distributed optimization with the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2019.

[24] Y. You et al., "Large batch optimization for deep learning: Training BERT in 76 minutes," in *Proc. ICLR*, 2020.

[25] V. Smith et al., "The system-level implications of federated learning," in *Proc. MLSys*, 2020.

Received: August 06, 2025

International Journal of Applied Mathematics
Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Received: August 06, 2025