Minimax Theory and its Applications
Volume 11 (2026), No. 4s, 101 to 144

Computational complexity analysis of large-scale mathematical optimization algorithms in engineering applications

Dr.P. Sathish Kumar[1], P.R. Abhinav[2], Dr. Jyoti Upadhyay[3], Dr.D. Venkatesa Prabu[4], Sunil Kumar Yadav[5], K. Lavanya[6]

[1]Assistant Professor, Department of Information Technology, J.J. College of Engineering and Technology, Tiruchirappalli, Tamil Nadu, India. sathishkumarp@jjcet.ac.in, https://orcid.org/0000-0002-5885-5808

[2]Assistant Professor, Department of Computer Science and Engineering, K.S. Rangasamy College of Technology, Namakkal, Tamil Nadu, India. eabhinavpr@ksrct.ac.in, https://orcid.org/0009-0007-0867-536X

[3]Associate Professor, Department of Computer Science and Engineering, Rungta College of Engineering and Technology, Rungta International Skills University, Bhilai, Chhattisgarh, India. jyoti.upadhyay@rungta.org, https://orcid.org/0000-0003-0640-5378

[4]Assistant Professor, Department of Mechatronics Engineering, Sona College of Technology, Salem, Tamil Nadu, India. venkatesa.mct@sonatech.ac.in, https://orcid.org/0000-0002-3213-317X

[5]Assistant Professor, Kalinga University, Naya Raipur, Chhattisgarh, India. ku.sunilkumaryadav@kalingauniversity.ac.in, https://orcid.org/0009-0004-0034-277X

[6]Department of MBA, Ramachandra College of Engineering, Eluru, Andhra Pradesh, India. lavanyak@rcee.ac.in

Abstract:

The mathematical optimization algorithms represent an essential tool in solving complicated engineering problems with high-dimensional space of parameters and nonlinear constraints. However, the increase of problem dimension leads to specific difficulties related to execution time, convergence rate, memory and scalability. This research proposes the framework for the complexity analysis of the optimization algorithms for the engineering application. The complexity of the gradient descent, accelerated gradient, quasi-Newton, Newton optimization, interior point and alternating direction method of multipliers is estimated using such criteria as iteration number, computational costs, memory requirements and impact of the sparsity structure. The numerical tests are performed for synthetic problems of strongly convex quadratic optimization with dimensions from 50 to 1600 and condition number from 5 to 400. It is shown that for increasing dimensions and fixed $\kappa = 50$, gradient descent requires 807–945 iterations for convergence, while Newton optimization converges after one iteration but with higher computation cost. Accelerated gradient optimization decreases iteration number from 7356 to 420 for increasing $\kappa = 400$ (17.5 times less iterations than conventional gradient descent). BFGS optimization converges with a stable number of iterations (from 60 to 98) for dimensions from 50 to 800. Sparsity structure analysis shows that sparse solver achieves 0.0196 seconds execution time at $n = 51{,}200$ while dense factorization execution time is more than 30 seconds beyond $n \approx 5000$.

Keywords: Computational complexity analysis, large-scale optimization, mathematical optimization algorithms, engineering applications, scalability analysis, convergence analysis, metaheuristic optimization.

1. Introduction

Due to the increasing complexity of modern engineering systems, which may include energy systems, production facilities, communication channels, and computerized environments, there arise numerous problems related to development of highly effective mathematical optimization algorithms. Optimization tasks in engineering usually require dealing with multidimensional data, non-linear constraints, multiple criteria and even dynamics of operations, which requires using some sophisticated computation methods in order to solve these problems accurately and effectively. It has been shown that modern mathematical modeling and optimization technologies can be applied successfully to solve engineering problems due to the improvement in decision making and system performance (Abhang et al., 2026).

Optimization on a large-scale needs complexity assessment tools to assess the performance and scalability of an algorithm, computational requirements, and so on. Complexity assessment of evolutionary algorithms and other metaheuristic approaches has emphasized the need for assessing the computational and execution efficiency when dealing with optimization problems on a large scale (Sari & Yildirim, 2026). The use of artificial intelligence for optimization has great potential in improving the energy management of systems through intelligent decision-making and resource utilization in smart grids. Decomposition optimization has also been considered to make the problem manageable by splitting it into several smaller problems (Chuong et al., 2025). Indexed tree sorting offers a more efficient way of sorting very large data sets because it enhances time efficiency and lowers computing costs, and is applicable to large scale optimization in distributed engineering networks (Agrawal et al., 2012).

The optimization of the power system and engineering design has relied on optimization algorithms for enhancing system reliability and efficiency. Optimal capacitors in power distribution network have been optimized using advanced optimization techniques for voltage stability and reduction of power losses while ensuring computational efficiency (Monrazeri et al., 2025). The human-inspired metaheuristic algorithms have made more contributions in solving the difficult engineering optimization problems with flexible search techniques and better exploration abilities (Sharifi et al., 2025). The applications of large-scale networks like the satellite communication network need efficient computations and optimization processes in order to meet the rising resource requirements (Guo et al., 2025). In the same way, optimization of distributed generation has proved that comparative metaheuristics can solve the difficult placement problems efficiently (Hamadneh et al., 2025).

Multi-objective optimization has become a crucial element of engineering applications, which require the simultaneous optimization of multiple objectives. Multi-objective optimization using genetic algorithms has been considered in order to obtain better trade-offs among conflicting objectives in complex engineering systems. The development of reinforcement learning-based optimization techniques has proven to be an efficient method of dealing with big data processing and dynamic optimization problems, thanks to the adaptability of the system (Yang & Liu, 2025). Moreover, advanced many-objective optimization techniques have been developed, which help to solve engineering design problems with a large number of objectives (Kalita et al., 2024). The recent research on optimization techniques has focused on the development of intelligent algorithms (Selvarajan, 2024). Healthcare services that make use of cyber-physical systems that have AI integration are likely to enhance the monitoring of the patients' vital signs in an efficient manner (Shrivastava et al., 2026).

The development of intelligent computing frameworks has paved the way for innovative ways of achieving increased efficiency in the optimization process in the context of real-time and large-scale engineering. Optimization approaches based on DNA computing framework have been developed for smart factory scheduling process, indicating the ability of unconventional computing frameworks to deal with difficult scheduling problems. Enhanced topology optimization frameworks have been explored for minimizing computational costs in engineering designs and increasing efficiency of the solution process (Mukherjee et al., 2021). Innovative optimization algorithms including arithmetic and evolutionary methods have been shown to achieve better performance in solving constrained multi-objective engineering problems due to increased diversity maintenance and validation procedures (Premkumar et al., 2021). Geometric programming-based optimization techniques have also proven their ability in dealing with complex decision-making situations with uncertainty and numerous constraints.

Some of the recent advancements in areas like artificial intelligence and computational modeling have broadened the applicability area of large-scale optimization. Some geometric neural operator models have offered efficient approaches for large-scale partial differential equation problems that combine physical knowledge with machine learning models (Li et al., 2023). There have been bio-inspired optimization methods that can help improve solution quality for numerical optimization and industrial engineering problems because of efficient exploration and exploitation techniques (Bouaouda et al., 2024). There have been nature-inspired optimization algorithms like cheetah-based optimization models that have shown good results in solving large-scale optimization problems with very high-dimensional solution space (Akbari et al., 2022). There have been software packages for large-scale constrained nonlinear optimization problems (Wu et al., 2020).

Key Contributions

- A framework that offers an analysis of the computational complexity of large-scale optimization algorithms is formulated in this study through an analysis of execution time, convergence, scalability, and computational resource requirements as the dimension of the problem increases.
- The analysis framework evaluates various optimization methods, such as evolutionary, metaheuristic, decomposition, and intelligent optimization methods, to determine which can be applied to large-scale optimization problems within engineering applications.
- Insights on the scalability of optimization algorithms are generated in this study to select appropriate computational methods for large-scale engineering applications.

The rest of the paper is structured as follows. In section II introduce the basic notations and preliminaries on optimization theory. In section III, describe the problem formulation and structural regimes for large scale engineering optimization. In section IV, provide the theoretical complexity analysis and convergence properties of various optimization algorithms. In section V, show the computational complexity analysis, memory requirements, and scalability analysis. In section VI, discuss the engineering examples, whereas in section VII, provide the numerical results and analysis. The implications of the results are discussed in section VIII. The conclusion of the paper is given in section IX.

2. Notation and preliminaries

To synthesize these foundational optimization definitions into a cohesive text, consider an objective function $f: \mathbb{R}^n \to \mathbb{R}$ assumed to be twice continuously differentiable ($f \in C^2$), where $n \in \mathbb{N}$ denotes the problem dimension and $m \in \mathbb{N}$ specifies the number of constraints. The first- and second-order local behaviors of f are characterized by its gradient vector $\nabla f(x) = (\partial f/\partial x_1, \dots, \partial f/\partial x_n)^T \in \mathbb{R}^n$ and Hessian matrix $\nabla^2 f(x) = \left[\partial^2 f/\partial x_i \partial x_j\right]_{i,j=1}^n \in \mathbb{R}^{n\times n}$. Standard spatial measures rely on the Euclidean vector norm $\|x\| = \sqrt{\sum_{i=1}^n x_i^2}$, while matrix operations utilize the induced spectral norm $\|A\|_2 = \sigma_{\max}(A)$, the Frobenius norm $\|A\|_F = \sqrt{\mathrm{Tr}(A^T A)}$, and structural sparsity measured by nonzeros $\mathrm{nnz}(A) = |\{(i,j): A_{ij} \neq 0\}|$.

The analytical regularity of the objective is governed by smoothness and convexity parameters. The function exhibits L-smoothness if its gradient is L-Lipschitz continuous, satisfying $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^n$ with $L > 0$. Furthermore, f is μ-strongly convex ($\mu \geq 0$) if it obeys $f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{\mu}{2}\|y - x\|^2$, where setting $\mu = 0$ yields ordinary convexity. The ill-conditioning of a strongly convex, smooth objective is quantified by its condition number $\kappa = L/\mu$, which for quadratic forms $f(x) = \frac{1}{2}x^T Ax - b^T x$ simplifies to the ratio of extreme eigenvalues $\kappa = \lambda_{\max}(A)/\lambda_{\min}(A)$.

Algorithmic performance and convergence are evaluated relative to optimal benchmarks and target tolerances. The optimal solution is denoted by $x^* = \arg\min_{x\in\mathbb{R}^n} f(x)$, achieving the minimum value $f^* = f(x^*)$. Iterative solvers aim to compute an estimate x_k satisfying a target accuracy $\varepsilon > 0$ under criteria such as objective error $f(x_k) - f^* \leq \varepsilon$, distance to optimality $\|x_k - x^*\| \leq \varepsilon$, or stationarity $\|\nabla f(x_k)\| \leq \varepsilon$. Asymptotic computational complexity across key parameters $(n, \kappa, 1/\varepsilon)$ is expressed using standard big-O notation $O(\cdot)$.

2.1. Convexity, smoothness and optimality conditions

To present these fundamental optimization principles as a unified text, begin with the first-order optimality criteria for differentiable convex functions. For an unconstrained objective $f: \mathbb{R}^n \to \mathbb{R}$, a point $x^* \in \mathbb{R}^n$ is a global minimizer if and only if it satisfies the stationarity condition $\nabla f(x^*) = 0$. When the domain is restricted to a

closed convex set $\mathcal{C} \subseteq \mathbb{R}^n$, optimality of $x^* \in \mathcal{C}$ is characterized by the first-order variational inequality $\langle \nabla f(x^*), x - x^* \rangle \geq 0$ for all $x \in \mathcal{C}$. For general constrained programs with differentiable objective f and constraints $g_i(x) \leq 0$ $(i = 1, \dots, m)$ and $h_j(x) = 0$ $(j = 1, \dots, p)$, under a suitable constraint qualification (such as Slater's condition), x^* is optimal if and only if there exist dual vectors $\mu \in \mathbb{R}_+^m$ and $\lambda \in \mathbb{R}^p$ satisfying the Karush–Kuhn–Tucker (KKT) system: stationarity $\nabla f(x^*) + \sum_{i=1}^m \mu_i \nabla g_i(x^*) + \sum_{j=1}^p \lambda_j \nabla h_j(x^*) = 0$, primal feasibility $g_i(x^*) \leq 0$ and $h_j(x^*) = 0$, dual feasibility $\mu_i \geq 0$, and complementary slackness $\mu_i g_i(x^*) = 0$ for all i.

The analytic properties of twice-differentiable objectives ($f \in C^2$) are linked directly to spectral bounds on the Hessian operator $\nabla^2 f(x)$. Specifically, L-smoothness and μ-strong convexity are equivalent to the matrix inequality $\mu I \preceq \nabla^2 f(x) \preceq LI$ holding for all $x \in \mathbb{R}^n$, where I is the identity matrix, $\mu \geq 0$, and $L > 0$. In the special case of a quadratic objective $f(x) = \frac{1}{2} x^T A x - b^T x$ with a symmetric positive definite matrix $A \succ 0$, the Hessian is constant ($\nabla^2 f(x) = A$). Consequently, the parameters reduce to the extreme eigenvalues of A, with $L = \lambda_{\max}(A)$ and $\mu = \lambda_{\min}(A)$, rendering the function condition number $\kappa = L/\mu$ identical to the matrix condition number $\kappa(A) = \lambda_{\max}(A)/\lambda_{\min}(A) = \|A\|_2 \|A^{-1}\|_2$.

2.2. Self-concordant barriers

Let $\mathcal{K} \subset \mathbb{R}^n$ be a closed convex set with nonempty interior $\text{int}(\mathcal{K})$. A convex function ϕ:$\text{int}(\mathcal{K}) \to \mathbb{R}$ is defined as a ν-self-concordant barrier for $\mathcal{K}$ if it approaches infinity as x approaches the boundary $\partial\mathcal{K}$ ($\lim_{x \to \partial\mathcal{K}} \phi(x) = +\infty$), is three-times continuously differentiable ($\phi \in C^3$), and satisfies two core analytical inequalities for all $x \in \text{int}(\mathcal{K})$. First, in the sense of Nesterov and Nemirovski, ϕ must be self-concordant such that its third directional derivative is bounded by its second directional derivative to the $3/2$ power: $|D^3\phi(x)[h,h,h]| \leq 2(D^2\phi(x)[h,h])^{3/2}$ for all directions $h \in \mathbb{R}^n$, or equivalently $\|\nabla^3 \phi(x)\|_{\nabla^2\phi(x)} \leq 2$. Second, the local dual norm of its gradient with respect to the inverse Hessian metric must satisfy the bounded parameter condition:

$$\langle \nabla\phi(x), [\nabla^2\phi(x)]^{-1} \nabla\phi(x) \rangle = \|\nabla\phi(x)\|^2_{\nabla^2\phi(x)^{-1}} \leq \nu$$

The barrier parameter $\nu \geq 1$ directly determines the iteration complexity of interior-point optimization methods. Standard canonical implementations include the logarithmic barrier for the nonnegative orthant $\mathbb{R}_+^m$, defined by $\phi(x) = -\sum_{i=1}^m \log(x_i)$ with parameter $\nu = m$ for m inequality constraints, and the log-determinant barrier for the cone of $p \times p$ symmetric positive definite matrices $\mathbf{S}_+^p$, defined by $\phi(X) = -\log\det(X)$ with parameter $\nu = p$.

2.3. Problem classes

An optimization problem $\min_{x \in \mathbb{R}^n} f(x)$subject to $Ax = b$and $g_i(x) \leq 0$for $i = 1, \dots, m$(with $A \in \mathbb{R}^{p \times n}$, $b \in \mathbb{R}^p$, and g: $\mathbb{R}^n \to \mathbb{R}^m$) is defined as a convex program if the objective f: $\mathbb{R}^n \to \mathbb{R}$and all inequality constraint components $g_i$are convex functions, while equality constraints are affine. When the objective is a convex quadratic function $f(x) = \frac{1}{2} x^T Q x + c^T x$with $Q \succeq 0$and $c \in \mathbb{R}^n$, and all inequality constraints are linear ($g(x) = Gx - h \leq 0$), the formulation is classified as a convex quadratic program (QP):

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T Q x + c^T x \quad \text{subject to} \quad Gx \leq h, \quad Ax = b$$

High-dimensional problems of dimension $n$frequently exhibit structured sparsity dictated by underlying system topology. A problem possesses bandwidth-b sparsity if its Hessian operator $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$or KKT matrix $K = \begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}$contains at most $\mathcal{O}(b)$nonzeros per row around the diagonal. Alternatively, mesh sparsity arises when the nonzero pattern matches the adjacency graph of a d-dimensional physical or network mesh, yielding a total non-zero count of $\text{nnz} = \mathcal{O}(n)$. Systems displaying mesh sparsity admit direct solution via sparse Cholesky factorizations, achieving iteration computational complexities of $\mathcal{O}(n^{1.5})$for two-dimensional structures ($d = 2$) and $\mathcal{O}(n^2)$for three-dimensional structures ($d = 3$).

3. Problem formulation

In this study three canonical classes of optimization formulations: the unconstrained problem (P1): $\min_{x\in\mathbb{R}^n} f(x)$, the linearly-constrained problem (P2): $\min_{x\in\mathbb{R}^n} f(x)$ s.t. $Ax = b, x \in \mathcal{K}$ where $\mathcal{K} \subseteq \mathbb{R}^n$ is a closed convex set (such as $\mathbb{R}^n$, the nonnegative orthant $\mathbb{R}^n_+$, a box, or a second-order cone), and the linearly-coupled separable problem (P3): $\min_{x,z} h(x) + r(z)$ s.t. $Bx + Cz = d$, where h and r are convex functions with r allowed to be non-smooth. Formulations (P1) and (P2) encompass compliance minimization and DC optimal power flow—where f is a convex quadratic and power flow adds convex quadratic inequality constraints—while receding-horizon trajectory optimization reduces to a sequence of (P2) instances featuring block-banded constraint matrices, and its distributed variant maps directly to (P3). For complexity analysis purposes, distinguish three structural regimes that dictate algorithmic behavior: (i) the dense, unstructured regime, in which the constraint matrix A or Hessian $\nabla^2 f(x)$ lacks exploitable sparsity and requires standard dense linear algebra; (ii) the mesh/network-sparse regime (Definition 3), where non-zero patterns match the adjacency structure of finite-element models or power grids; and (iii) the block-separable regime of (P3), which isolates system components across multi-agent, multi-period, or multi-region engineering formulations to enable distributed decomposition methods.

4. Main theoretical results

This section states and proves the iteration-complexity results used in Section 5. Throughout, x_k denotes the k-th iterate and x* a minimizer of the relevant problem.

4.1. Gradient descent-convex, L-smooth case

Theorem 1 (Convergence of Gradient Descent for Smooth Convex Functions)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be convex and L-smooth ($L > 0$), and consider the gradient descent iteration:

$$x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k), \quad k \geq 0$$

starting from an initial point $x_0 \in \mathbb{R}^n$. Then for every $k \geq 1$, the suboptimality gap satisfies:

$$f(x_k) - f^* \leq \frac{L\|x_0 - x^*\|^2}{2k}$$

Consequently, obtaining an ε-optimal solution such that $f(x_k) - f^* \leq \varepsilon$ requires at most:

$$k = \mathcal{O}\left(\frac{L\|x_0 - x^*\|^2}{\varepsilon}\right) \text{ iterations}$$

where each iteration requires evaluating $\nabla f(x_k)$, costing $\mathcal{O}(n)$ arithmetic operations in the dense unstructured regime (and less under structured or sparse settings).

Proof

Descent Lemma Step: By L-smoothness of f, the descent lemma yields:

$$f(x_{k+1}) \leq f(x_k) + \langle \nabla f(x_k), x_{k+1} - x_k \rangle + \frac{L}{2}\|x_{k+1} - x_k\|^2$$

Substituting the update step $x_{k+1} - x_k = -\frac{1}{L}\nabla f(x_k)$ simplifies this to:

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2$$

Distance to Optimality & Non-expansiveness: By convexity of f, have:

$$f(x_k) - f^* \leq \langle \nabla f(x_k), x_k - x^* \rangle \leq \|\nabla f(x_k)\|\|x_k - x^*\|$$

Using the step size $\alpha = 1/L$, non-expansiveness dictates that $\|x_{k+1} - x^*\| \leq \|x_k - x^*\|$, which by induction establishes:

$$\|x_k - x^*\| \leq \|x_0 - x^*\|, \quad \forall k \geq 0$$

Telescoping **Sum & Rate Derivation:** Combining the single-step descent inequality with the distance bound and summing over $k = 0, \ldots, K-1$ yields the telescoping sum that proves the $\mathcal{O}(1/k)$ rate (for full details, see Nesterov, 2018, Thm. 2.1.14).

4.2 Gradient Descent-strongly convex case

Theorem 2 (Linear Convergence of Gradient Descent for Strongly Convex Functions)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be μ-strongly convex ($\mu > 0$) and L-smooth ($L > 0$) with condition number $\kappa = L/\mu \geq 1$, and consider the gradient descent iteration $x_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$ for $k \geq 0$ starting from an initial point $x_0 \in \mathbb{R}^n$. Then for every $k \geq 1$, the distance to the unique minimizer x^* satisfies $\|x_k - x^*\|^2 \leq \left(1 - \frac{1}{\kappa}\right)^k \|x_0 - x^*\|^2$, which implies that obtaining an ε-accurate solution satisfying $\|x_k - x^*\| \leq \varepsilon$ requires at most $k = \mathcal{O}\left(\kappa \log\left(\frac{\|x_0 - x^*\|}{\varepsilon}\right)\right)$ iterations, demonstrating that gradient descent achieves linear (geometric) convergence with a rate governed directly by the condition number κ.

Proof

1. **Co-coercivity Bound:** Combining μ-strong convexity and L-smoothness yields the tight co-coercivity property $\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq \frac{\mu L}{\mu + L}\|x - y\|^2 + \frac{1}{\mu + L}\|\nabla f(x) - \nabla f(y)\|^2$ for all $x, y \in \mathbb{R}^n$.

2. **Single-Step Contraction:** Expanding the squared distance $\|x_{k+1} - x^*\|^2 = \left\|x_k - \frac{1}{L}\nabla f(x_k) - x^*\right\|^2 = \|x_k - x^*\|^2 - \frac{2}{L}\langle \nabla f(x_k), x_k - x^* \rangle + \frac{1}{L^2}\|\nabla f(x_k)\|^2$ and applying the co-coercivity inequality at (x_k, x^*) along with the first-order optimality condition $\nabla f(x^*) = 0$ yields the one-step contraction $\|x_{k+1} - x^*\|^2 \leq \left(1 - \frac{\mu}{L}\right)\|x_k - x^*\|^2 = \left(1 - \frac{1}{\kappa}\right)\|x_k - x^*\|^2$.

3. **Iterative Contraction & Complexity:** Iterating this contraction from $k = 0$ yields the geometric bound $\|x_k - x^*\|^2 \leq (1 - 1/\kappa)^k \|x_0 - x^*\|^2$, from which taking logarithms and solving for k completes the proof of the iteration count.

4.3. Accelerated gradient descent (Neserov)

Theorem 3 (Nesterov accelerated gradient descent)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be μ-strongly convex ($\mu > 0$) and L-smooth ($L > 0$) with condition number $\kappa = L/\mu \geq 1$. The Nesterov accelerated gradient iteration defined by $y_k = x_k + \beta(x_k - x_{k-1})$, $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$, with momentum parameter $\beta = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$, satisfies $f(x_k) - f^* \leq \frac{L + \mu}{2}\|x_0 - x^*\|^2 \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k$. Consequently, achieving an ε-optimal solution $f(x_k) - f^* \leq \varepsilon$ requires at most $k = \mathcal{O}\left(\sqrt{\kappa} \log\left(\frac{1}{\varepsilon}\right)\right)$ iterations, representing a quadratic improvement in condition-number dependence over the $\mathcal{O}\left(\kappa \log\left(\frac{1}{\varepsilon}\right)\right)$ rate of standard gradient descent while maintaining the identical $\mathcal{O}(n)$ per-iteration arithmetic cost.

Proof (Sketch)

1. **Lyapunov Estimate Sequence:** The proof constructs an estimate sequence $\Phi_k(x) = f(x_k) + \frac{\gamma_k}{2}\|x - v_k\|^2$ anchored by an auxiliary sequence $v_k \in \mathbb{R}^n$ and decreasing weights $\gamma_k > 0$.

2. **Inductive Contraction:** By establishing that $f(x_k) - f^* \leq \Phi_k(x^*) - f^* \leq \lambda_k(\Phi_0(x^*) - f^*)$ inductively, the contraction factor accelerates to $\lambda_k = \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k$ under the optimal choice of momentum β.

3. **Optimality and Derivation:** The precise derivation of β ensures optimal rate contraction, as detailed in Nesterov (2018, §2.2.1) and Bubeck (2015, Thm. 3.18).

Remarks & optimality bounds

Theorem 3 is of central practical significance because the $\mathcal{O}\left(\sqrt{\kappa}\log\left(\frac{1}{\varepsilon}\right)\right)$ iteration complexity is information-theoretically optimal among all first-order methods accessing f solely through gradient evaluations. Consequently, no first-order algorithm can achieve better than a $\sqrt{\kappa}$ scaling in the worst case, establishing acceleration as the benchmark remedy for ill-conditioned systems when second-order information is unavailable or computationally prohibitive.

4.4. Quasi-newton methods

Theorem 4 (Superlinear convergence of BFGS quasi-newton method)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable ($f \in C^2$) and μ-strongly convex with a Lipschitz-continuous Hessian operator. Consider the quasi-Newton iteration $x_{k+1} = x_k - H_k \nabla f(x_k)$, where $H_k \in \mathbb{R}^{n\times n}$ is the BFGS approximation of $[\nabla^2 f(x_k)]^{-1}$ updated via the standard secant (Sherman–Morrison) formula. Under these regularity conditions, and assuming x_0 and the initial matrix H_0 are chosen sufficiently close to x^* and $[\nabla^2 f(x^*)]^{-1}$ respectively, the sequence of iterates x_k converges superlinearly to the unique minimizer x^*:

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|} = 0$$

Consequently, the iteration complexity required to achieve an error tolerance $\|x_k - x^*\| \le \varepsilon$ scales as $\mathcal{O}\left(\log\left(\frac{1}{\varepsilon}\right)\right)$, with a rate constant that continuously decreases as iteration count k increases. Each iteration incurs a computational cost of $\mathcal{O}(n^2)$ arithmetic operations for the dense secant matrix update and matrix–vector product.

Proof (Sketch)

1. **Dennis–Moré Characterization:** By the classical Dennis–Moré condition, a quasi-Newton iteration $x_{k+1} = x_k - H_k \nabla f(x_k)$ converges superlinearly if and only if the step direction satisfies $\lim_{k\to\infty} \frac{\|\left(H_k - [\nabla^2 f(x^*)]^{-1}\right)(x_{k+1} - x_k)\|}{\|x_{k+1} - x_k\|} = 0.$

2. **Secant Property & Bounded Deterioration:** For the BFGS update, enforcing the secant equation $H_{k+1}\left(\nabla f(x_{k+1}) - \nabla f(x_k)\right) = x_{k+1} - x_k$ guarantees that H_k tracks the true inverse Hessian along the step directions. The bounded deterioration property ensures H_k does not drift significantly from $[\nabla^2 f(x^*)]^{-1}$ over successive updates.

3. **Asymptotics:** Combining bounded deterioration with the Lipschitz continuity of $\nabla^2 f$ establishes that the Dennis–Moré condition holds along the sequence of iterates .

Computational & Algorithmic Trade-offs

The $\mathcal{O}(n^2)$ per-iteration cost of the BFGS algorithm is asymptotically cheaper than full Newton steps, which require an $\mathcal{O}(n^3)$ matrix factorization of $\nabla^2 f(x_k)$ at every iteration. However, this computational savings comes at the trade-off of a superliner rather than quadratic asymptotic rate, often translating to a higher total number of iterations in practice compared to pure Newton steps.

4.5. Newton's method-local quadratic convergence

Theorem 5 (Quadratic Convergence of Newton's Method)

Let $f: \mathbb{R}^n \to \mathbb{R}$ be twice continuously differentiable ($f \in C^2$) with an M-Lipschitz continuous Hessian operator ($\|\nabla^2 f(x) - \nabla^2 f(y)\| \le M\|x - y\|$), and let $\nabla^2 f(x^*) \ge \mu I$ ($\mu > 0$) at the minimizer x^*. There exists a open neighborhood around x^* such that for any initial point x_0 within this neighborhood, the Newton iteration $x_{k+1} = x_k - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$ satisfies:

$$\|x_{k+1} - x^*\| \le \frac{M}{2\mu}\|x_k - x^*\|^2$$

Consequently, once an iterate enters the local region of quadratic convergence, the number of additional iterations

required to reach an accuracy ε scales as $\mathcal{O}\left(\log\log\left(\frac{1}{\varepsilon}\right)\right)$. The per-iteration computational cost is dominated by forming and factoring the $n \times n$ Hessian matrix: $\mathcal{O}(n^3)$ operations for dense unstructured systems, reducing to $\mathcal{O}(n^{1.5})$ for two-dimensional ($d = 2$) and $\mathcal{O}(n^2)$ for three-dimensional ($d = 3$) mesh-sparse or banded Hessians (Definition 3).

Proof

1. **Taylor Expansion with Lipschitz Hessian:** Denoting the error at step k by $e_k = x_k - x^*$, Taylor's theorem applied to $\nabla f(x_k)$ around x^* yields $\nabla f(x_k) = \nabla f(x^*) + \nabla^2 f(x^*)e_k + r_k$, where the remainder is bounded by $\|r_k\| \leq \frac{M}{2}\|e_k\|^2$. Using the first-order optimality condition $\nabla f(x^*) = 0$, this simplifies to $\nabla f(x_k) = \nabla^2 f(x^*)e_k + r_k$.

2. **Error Dynamics Expansion:** Substituting the Newton step into the error expression gives:

$$e_{k+1} = e_k - [\nabla^2 f(x_k)]^{-1}\nabla f(x_k) = [\nabla^2 f(x_k)]^{-1}\left(\nabla^2 f(x_k)e_k - \nabla f(x_k)\right)$$

   Inserting the Taylor expansion for $\nabla f(x_k)$ yields $e_{k+1} = [\nabla^2 f(x_k)]^{-1}\left(\left(\nabla^2 f(x_k) - \nabla^2 f(x^*)\right)e_k - r_k\right)$.

3. **Norm Bounds & Local Quadratic Rate:** Bounding the Hessian difference by $\|\nabla^2 f(x_k) - \nabla^2 f(x^*)\| \leq M\|e_k\|$, and utilizing $\|[\nabla^2 f(x_k)]^{-1}\| \leq \frac{1}{\mu}$ for x_k sufficiently close to x^* (via continuity and strong convexity at x^*), standard symmetrized integral derivations (Nocedal & Wright, 2006, Thm. 3.5) yield the sharp quadratic error bound $\|e_{k+1}\| \leq \frac{M}{2\mu}\|e_k\|^2$.

5. Complexity analysis

Table 1 restates the iteration complexity results from Section 4 as total arithmetic complexity by adding up the iteration number and linear algebra costs per iteration required to attain precision level ε on an n-dimensional problem with condition number κ, and self-concordance parameter ν for the interior point row.

Table 1: Complexity analysis.

Method	Iterations	Cost / iteration	Total complexity
Gradient descent (convex)	$O(1/\varepsilon)$	$O(n)$	$O(n/\varepsilon)$
Gradient descent (strongly convex)	$O(\kappa\, log(1/\varepsilon))$	$O(n)$	$O(n\kappa\, log(1/\varepsilon))$
Accelerated gradient (Nesterov)	$O(\sqrt{\kappa}\, log(1/\varepsilon))$	$O(n)$	$O(n\sqrt{\kappa}\, log(1/\varepsilon))$
Quasi-Newton (BFGS)	$O(log(1/\varepsilon))$ *	$O(n^2)$	$O(n^2\, log(1/\varepsilon))$ *
Newton's method (local)	$O(log\, log(1/\varepsilon))$	$O(n^3)$	$O(n^3\, log\, log(1/\varepsilon))$
Interior-point (self-concordant)	$O(\sqrt{\nu}\, log(1/\varepsilon))$	$O(n^3)$	$O(n^3\sqrt{\nu}\, log(1/\varepsilon))$
ADMM (ergodic)	$O(1/\varepsilon)$	$O(n)\ -\ O(n\text{^}1.5)$	$O(n\text{^}1.5/\varepsilon)$

5.1. Memory complexity

The memory cost may become the bottleneck constraint at large scale for engineering problems, and is often more significant than the arithmetic cost in determining the algorithm to use. For gradient descent and accelerated gradient descent, memory cost is merely O(n) (the current iterates, the gradient, and a few auxiliary vectors). For BFGS in its standard dense version, memory cost is O(n²) due to the storage of the Hessian approximation, which is already impractical for n larger than roughly 10^4–10^5 on current hardware; for the limited-memory version (L-BFGS), which only stores m ≪ n secant pairs, the memory cost is $O(mn)$ but with the loss of the full superlinear convergence guarantee provided by Theorem 4 (L-BFGS can only have a linear local rate of convergence generally speaking, although it does exhibit super linear convergence practically in many problems). Memory cost of Newton's method and interior point methods is $O(n^2)$ due to storage of the dense Hessian/KKT matrix, or $O(nnz)$ for the sparse factorization case, which is $O(n)$ under the mesh sparsity condition of Definition 3.

5.2. The role of sparsity and structure

Table 1 $O(n^3)$ dense-case costs for Newton-type and interior point methods are worst case bounds that almost never apply to engineering problems, as the Hessian or KKT matrix of the discretized physical problem will be sparse due to the sparsity of the mesh, circuit or network graph. If the mesh is sparse as in Definition 3, then sparse Cholesky (or LDL^{T}) factorization takes only $O(n\hat{}\{1.5\})$ work in two dimensions and $O(n^2)$ in three dimensions, via nested-dissection or minimum-degree ordering to bound fill-in, rather than $O(n^3)$. This is by far the most important practical adjustment to the complexity bounds of Section 4, and is why Newton-type and interior point methods scale reasonably well at the size of the problems in Section 6 despite having poor asymptotics in the dense case. This adjustment is made explicit in Section 7.4, where sparse factorization is already three to four orders of magnitude faster than dense factorization at n=3200, and dense factorization is impractical (>30 s on one core) above about n=5000, while sparse factorization is under 0.02 s at n=51200.

5.3. Summary comparison

Three practical observations can be made based on table 1 and the preceding discussion of sparsity. First, gradient and accelerated gradient methods have a linear per-iteration complexity in terms of n but require an iteration count that scales with either κ or $\sqrt{\kappa}$; work best when n is very large, acceptable accuracy is sufficient, and gradient computations are inexpensive compared to solving linear systems — the domain of large-scale surrogate modeling optimization and many machine learning-inspired engineering problems. Second, Newton, quasi-Newton, and interior-point methods require a super-linear complexity per iteration ($O(n^2)\ to\ O(n^3)$ in dense form, $O(n^{1.5})\ to\ O(n^2)$ in sparse form) but just a few iterations up to several dozen, scaling well for medium to large values of n with sparse structure and high accuracy necessary — the usual domain of structural design verification, power system state estimation, and any application where certified KKT optimality is needed. Third, the ADMM's $O(1/\varepsilon)$ ergodic complexity is asymptotically worse in accuracy than Newton or interior-point methods but the decomposable and cheap-to-compute subproblems allow it to outperform others in case of naturally decomposable problem into multiple subsystems and when wall-clock time (rather than iteration count) governs the cost — distributed multi-region and multi-agent optimization.

6. Engineering case studies

6.1. Structural compliance minimization

As regards SIMP-based topology optimization, where the compliance objective $c(\rho) = f^{\mathrm{T}}u(\rho)$, where $u(\rho)$ is the displacement obtained as the solution of the discretized equilibrium equation $K(\rho)u = f$, after reformulation as in standard sensitivity-based topology optimization with a fixed mesh, is locally approximated by a quadratic in the design variables for small design steps, the bound results of Section 4 directly apply to the optimization of each sub-step (say within an SQP or method of moving asymptotes framework). In this case n is the number of finite elements (typically 10^5–10^7 in realistic 3-D cases), while the stiffness matrix K is sparse and its bandwidth depends on the mesh connectivity. This means that Definition 3 applies for d = 2 or d = 3. Theorem 5 (Newton's method) and the discussion on sparse factorization in Section 5.2 clarify why Newtonian (SQP/interior point) algorithms remain the preferred approach for this type of problems, even though their cost is $O(n^3)$, thanks to sparse factorization, which keeps the actual cost per iteration between $O(n\hat{}\{1.5\})$ and $O(n^2)$, together with quadratic local convergence, which leaves the number of Newton steps practically invariant to n. For larger problems, which can be solved via sparse direct factorization, first order and accelerated first order methods (Theorems 1–3) are used.

6.2. DC optimal power flow

The DC optimal power flow (OPF) problem minimizes a convex (usually quadratic) generator-cost function under linear power-balance and line-flow constraints — an example of (P2) where $\mathcal{K}$ is a box (bounds on generator and line flows). Contemporary power transmission systems have n (generator and bus variables) in the range of thousands to tens of thousands with a sparsity pattern in its constraint matrix matching the sparsity pattern of its graph, which is typically sparse and nearly planar (Definition 3). In Theorem 6 predict $O(\sqrt{v}\ log(1/\varepsilon))$ iterations of the interior-point method, each dominated by a linear system solving step exploiting sparsity structure of the network graph; it is consistent with contemporary OPF solvers, which regularly solve the problem in 15 to 40 interior-point iterations independent of the size of the network, with most of the computational effort spent on solving a sparse linear system (KKT factorization) rather than iterations number. In Theorem 7 predict that a regional decomposition of the problem by ADMM where each region runs its own OPF instance (standard in multi-utility coordination problem) will trade this fast convergence for a fast, yet local problem solving at each control area; it is consistent with the $O(1/K)$ rate proved above.

6.3. Receding-horizon trajectory optimization

Indeed, in LQR MPC, at every control iteration one solves the finite-horizon optimal control problem $min \sum_\{t = 0\}^\{N - 1\} (x_t^{\mathrm{T}} Q x_t + u_t^{\mathrm{T}} R u_t) + x_N^{\mathrm{T}} P x_N$ under the linear dynamics $x_\{t + 1\} = A_dx_t + B_du_t$ and box constraints on states and inputs, i.e. a convex QP of the block-tridiagonal type $(P2)$, with $n = N(n_x + n_u)$ increasing with the prediction horizon N and the state/input sizes n_x, n_u. Block-tridiagonal KKT system corresponds to mesh sparsity (Definition 3) of the bandwidth $O(n_x + n_u)$ independent of N; hence, the cost of sparse solving using Riccati recursion is $O(N)$, not $O(N^3)$ of the dense factorization; in other words, it represents an extreme example of sparsity correction from Section 5.2. This is precisely why interior-point and active-set QP solvers, and not first-order methods in general, dominate real-time MPC with modest horizon lengths: Theorem 6 together with the O(N) cost of a single Riccati solve results in the total complexity of $O(N\sqrt{\nu}\ log(1/\varepsilon))$, linear in N. For very small sample times and very large state dimension (e.g., in distributed MPC over large sensor networks), one resorts to operator splitting methods from Theorem 7, solved block-wise along the horizon or across space, respectively.

7. Numerical results

Herein present four numerical experiments supporting the predicted scaling by Theorems 1, 2, 3, and 6 (using the sparsity argument from Section 5.2). Unless otherwise noted, the test problems involved strongly convex quadratic optimization problems of the form min_x ½$x^{\mathrm{T}} Ax - b^{\mathrm{T}} x$, where A was chosen to be the orthogonal similarity transformation of a diagonal matrix whose eigenvalues ranged between 1 and κ and b was random. All computations were carried out using double precision on a single processor.

7.1. Scaling with problem dimension n (fixed k=50)

Gradient descent (step size 1/L) and Newton's method were applied up to a gradient-norm accuracy of 10^{-8} for quadratics of varying dimensions n = 50 to 1600 for a fixed condition number κ = 50 in table2.

Theorem 2 predicted that the number of iterations in the gradient-descent algorithm would remain essentially constant (807 to 945 iterations) as n increased from 50 to 1600 with fixed $\kappa = 50$, since the number of iterations was independent of n with the bound $O(\kappa\ log(1/\varepsilon))$. However, wall time increases with the $O(n)$ per-iteration cost of matrix-vector multiplication. Newton's algorithm takes only one iteration for these exactly-quadratic problems (as Theorem 5 predicted since for quadratics the Hessian matrix is constant and the convergence bound reduces to exact convergence after one iteration), but the wall time of Newton's algorithm increased substantially faster than that of gradient descent with increasing n, reflecting its $O(n^3)$ per-iteration cost relative to $O(n)$ per-iteration cost of gradient descent.

Table 2: Scaling with problem dimension.

n	κ	GD iters	GD time (s)	GD rel. err	Newton iters	Newton time (s)
50	50.0	807	0.0042	1.51e-08	1	0.0001
100	50.0	829	0.0054	7.49e-09	1	0.0001
200	50.0	945	0.0117	3.32e-09	1	0.0005
400	50.0	854	0.0372	2.67e-09	1	0.0032
800	50.0	898	0.2536	2.62e-09	1	0.0135
1600	50.0	887	1.0604	1.57e-09	1	0.0758

7.2. Scaling with condition number k (fixed n=400)

Table 3: Condition number scaling for plain gradient descent n=400.

κ	Gradient-descent iterations to convergence
5	84
10	169
25	442
50	890
100	1575
200	3700
400	6724

The number of iterations increases linearly with respect to κ, ranging from 84 at $\kappa = 5$ to 6724 at $\kappa = 400$, confirming the $O(\kappa\, log(1/\varepsilon))$ complexity of Theorem 2: going from κ = 200 to 400 causes the number of iterations to roughly double (from 3700 to 6724) in table3. It thus shows that the conditioning is the bottleneck rather than the intrinsic dimensionality in first-order methods in this family of problems, which constitutes the rationale for preconditioning in practical large-scale engineering systems where A tends to be inherently ill-conditioned.

7.3. Effect of acceleration

The condition number analysis from Section 7.2 of the Nesterov acceleration scheme (Theorem 3 with momentum parameter $\beta = (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1)$) versus gradient descent for a fixed value of n = 400.

Table 4: Plain Vs accelerated gradient descent n=400.

κ	Plain GD iterations	Accelerated GD iterations
5	86	37
10	182	59
25	451	96
50	889	138
100	1809	202
200	3454	278
400	7356	420

The empirical number of iterations matches the predicted $O(\kappa)$ compared to $O(\sqrt{\kappa})$ very well, where for $\kappa = 400$, gradient descent needs 7356 iterations compared to 420 iterations for the accelerated method, resulting in a factor 17.5 improvement that is in line with the $\sqrt{400} = 20$ factor expected asymptotically from comparison of Theorems 2 and 3. As κ increases, so does the advantage of acceleration, where for example for $\kappa = 5$ there is only a 2.3× improvement but for $\kappa = 400$ there is 17.5× improvement, which clearly shows that the accelerated approach is most beneficial exactly when gradient descent struggles in table4.

7.4. Quasi-newton behavior (BFGS)

the same class of quadratics (κ = 50) with dense BFGS (using SciPy's implementation), giving the number of iterations, elapsed time, and the relative error at convergence for n from 50 to 800. Consistent with Theorem 4, the number of BFGS iterations does not depend on n (60-98 iterations for an order-of-magnitude difference in n) and, importantly, is significantly lower than the $O(\kappa \log(1/\varepsilon)) \approx 890$ iterations required by gradient descent alone when the same κ = 50 is chosen (Table 5), thereby demonstrating the relevance of the super linear convergence rate despite the lack of access to exact second derivatives by BFGS. Nevertheless, wall clock time increases super linearly inn, consistent with the $O(n^2)$ complexity of each iteration in terms of the dense secant update and Hessian-vector product from Theorem 4.

Table 5: BFS behavior with dimensions K=50.

n	BFGS iterations	BFGS time (s)	BFGS rel. error
50	60	0.0111	2.81e-11
100	86	0.0261	1.53e-08
200	98	0.1179	9.39e-09
400	96	0.6356	1.86e-08
800	97	3.9126	8.15e-09

7.5. Sparse versus dense factorization cost

For the purpose of separating the effect of sparsity described in Section 5.2, considered banded (tridiagonal) symmetric positive definite matrices which are generated in the case of the 1-D discretization of elliptic equations — the simplest possible example of the mesh sparsity (Definition 3), with sizes n from 200 up to 51,200, comparing a sparse direct solver (sparse LU/Cholesky solver provided by SciPy) to a dense direct solver (dense solver provided by NumPy) at the sizes where the latter was computationally feasible.

Table 6: Sparse Vs dense linear system solve time on banded SPD systems.

n	Sparse solve (s)	Dense solve (s)
200	0.00115	0.00084
800	0.00046	0.01565
3200	0.00120	0.75571
12800	0.00561	n/a (> 30 s)
51200	0.01964	n/a (> 30 s)

This analysis quantifies the sparsity adjustment of Section 5.2. By n = 800, the sparse solver is about 34 times faster than the dense solver (0.00046 s versus 0.01565 s); by n = 3200, the difference is even more pronounced, around 630 times (0.0012 s versus 0.756 s), since the growth of cost in the dense solver is in $O(n^3)$ while it is near-linear in the sparse banded solver. For values larger than $n \approx 5000$, the dense factorization took more than 30 seconds and thus it is omitted, while the sparse solver continued to perform well, taking 0.0056 s by n = 12,800 and 0.0196 s by n = 51,200in table 6. This is the quantitative background to the statement in Sections 5.2 and 6, that sparsity utilization and not merely choice of method is responsible for the tractability of Newtonian and interior point methods on structured engineering problems in practice; a dense-only interior point method would become impractical in n roughly two orders of magnitude earlier than the sparse one.

8. Discussion

In summary, obtain a set of sufficient conditions for various optimization algorithms, leading to an optimal choice guideline in engineering practice (without specifying an algorithm), as presented below. Large n, cheap gradient evaluations, medium precision, well-conditioned or easily preconditioned: use accelerated gradient descent (Theorem 3); this approach is superior to plain gradient descent wherever κ is not near 1 at the same O(n) cost and memory footprint. Medium to large n, sparse problem structure, high precision required, KKT optimality certificate needed: use Newton-type or interior point methods (Theorems 5–6) with either a sparse direct or iterative KKT system solver – Section 7.5 proves that this remains feasible up to a much larger scale than the naive dense approach would allow. Medium n, memory available for $O(n^2)$ storage, non-expensive to evaluate objectives, costly to differentiate: use (L-)BFGS (Theorem 4) as an intermediate solution between O(n) and slow κ-dependent convergence of first-order methods and $O(n^3)$ and rapid convergence of full Newton. Naturally block-separable or multi-agent/multi-regions problems or time is limited by communication/parallelization cost rather than iterations count: use ADMM or another splitting approach (Theorem 7). Finally, this paper was restricted to deterministic full batch optimization. In the setting of estimation based on noisy sensor data, simulation Monte Carlo samples or streaming measurements (which occurs increasingly frequently in data-driven digital-twin and online estimation), stochastic first order methods, namely stochastic gradient descent and variance-reduced versions thereof, show a different complexity landscape: iteration complexity $O(1/\varepsilon)$ or $O(1/\varepsilon^2)$ in the noisy-gradient setting instead of $O(\log(1/\varepsilon))$ of the deterministic strongly convex one; a comprehensive treatment of the latter case is a subject of future research.

9. Conclusion

In this paper, a framework for computational complexity analysis of large-scale mathematical optimization algorithms used in engineering was developed based on execution time, convergence properties, scalability, and computing resource needs. The analysis included a number of algorithms, such as gradient descent, accelerated gradient, quasi-Newton, Newton, interior point optimization, and ADMM, in terms of their computational complexity features. Numerical simulations with strongly convex quadratic optimization problems, with the dimension ranging from 50 to 1600 and the condition number from 5 to 400, were performed. It was found that the gradient descent needed 807-945 iterations for increasing dimension at $\kappa = 50$, while the Newton optimization converged in a single iteration but had higher computational costs. The accelerated gradient optimization decreased the number of iterations from 7356 to 420 at $\kappa = 400$, which is 17.5× faster than the gradient descent optimization. The BFGS method had stable convergence with iterations ranging between 60–98 iterations over dimensions from 50 to 800, illustrating its efficiency as a method that combines both first-order and second-order optimizations techniques. Analysis using sparse factorization showed that the sparse method attained an execution time of 0.0196 s when n = 51,200, while dense factorization took more than 30 s for n>5000, emphasizing the need to harness the power of sparsity in large scale engineering optimizations. The scope of future studies will be on developing the proposed framework for nonconvex optimization problems, stochastic optimization settings, realistic 3D finite element models, nonlinear power system optimization problems, and data-driven digital twin's platform. Moreover, adaptive preconditioning, distributed optimization, GPU computing, and hybrid artificial intelligence aided optimization algorithms will be considered in order to enhance scalability and real-time decision-making capabilities of next generation large-scale engineering systems.

References

1. Abhang, L. B., Patel, M., & Bansode, V. H. (2026). Advanced Mathematical Modeling AND Optimization Techniques FOR Solving Complex Engineering Problems. *International Journal of Advances in Signal and Image Sciences*, 1437-1444. https://doi.org/10.29284/8k920742

2. Akbari, M. A., Zare, M., Azizipanah-Abarghooee, R., Mirjalili, S., & Deriche, M. (2022). The cheetah optimizer: A nature-inspired metaheuristic algorithm for large-scale optimization problems. *Scientific reports*, *12*(1), 10953. https://doi.org/10.1038/s41598-022-14338-z

3. Agrawal, P., Kaur, H., & Singh, G. (2012). Indexed tree sort: An approach to sort huge data with improved time complexity. *International Journal of Computer Applications, 57*(18), 26–32.

4. Bouaouda, A., Hashim, F. A., Sayouti, Y., & Hussien, A. G. (2024). Pied kingfisher optimizer: a new bio-inspired algorithm for solving numerical optimization and industrial engineering problems. *Neural Computing and Applications*, *36*(25), 15455-15513. https://doi.org/10.1007/s00521-024-09879-5

5. Chuong, T. D., Liu, C., & Yu, X. (2025). Decomposition for large-scale optimization problems: An overview. *Artificial Intelligence Science and Engineering*, *1*(3), 157-174. https://doi.org/10.23919/AISE.2025.000012

6. Guo, B., Zhang, Z., Atapattu, S., Pan, M., Yan, Y., Xiong, Z., & Li, H. (2025). Enabling real-time computing and transmission services in large-scale LEO satellite networks. *IEEE Transactions on Vehicular Technology*, *74*(8), 12813-12828. https://doi.org/10.1109/TVT.2025.3550806

7. Hamadneh, T., Batiha, B., & Gharib, G. M. (2025). Enhanced Distributed generation placement using carpet weaver optimization: A comparative metaheuristic approach. *Large Scale Syst*, *18*(22), 10-22266. 10.22266/inassexpress.2025.003

8. Kalita, K., Ramesh, J. V. N., Čep, R., Jangir, P., Pandya, S. B., Ghadai, R. K., & Abualigah, L. (2024). Many-objective whale optimization algorithm for engineering design and large-scale many-objective optimization problems. *International Journal of Computational Intelligence Systems*, *17*(1), 171. https://doi.org/10.1007/s44196-024-00562-0

9. Li, Z., Kovachki, N., Choy, C., Li, B., Kossaifi, J., Otta, S., ... & Anandkumar, A. (2023). Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, *36*, 35836-35854.

10. Monrazeri, Z., Malik, O. P., & Dehghani, M. (2025). Capacitor Placement in Power Distribution Networks for Voltage Profile Improvement and Loss Reduction Using Revolution Optimization Algorithm. *large-scale systems*, *18*, 22. https://doi.org/10.22266/inassexpress.2025.004

11. Mukherjee, S., Lu, D., Raghavan, B., Breitkopf, P., Dutta, S., Xiao, M., & Zhang, W. (2021). Accelerating Large-scale Topology Optimization: State-of-the-Art and Challenges. *Archives of Computational Methods in Engineering*, *28*(7), 4549-4571. https://doi.org/10.1007/s11831-021-09544-3

12. Premkumar, M., Jangir, P., Kumar, B. S., Sowmya, R., Alhelou, H. H., Abualigah, L., ... & Mirjalili, S. (2021). A new arithmetic optimization algorithm for solving real-world multiobjective CEC-2021 constrained optimization problems: diversity analysis and validations. *Ieee Access*, *9*, 84263-84295. https://doi.org/10.1109/ACCESS.2021.3085529

13. Sari, Z., & Yildirim, M. (2026). A Complexity Calculation Method for Large Scale Optimization with Evolutionary Algorithms and Metaheuristics. *Concurrency and Computation: Practice and Experience, 38*(8), e70702. https://doi.org/10.1002/cpe.70702

14. Selvarajan, S. (2024). A comprehensive study on modern optimization techniques for engineering applications. *Artificial Intelligence Review*, *57*(8), 194. https://doi.org/10.1007/s10462-024-10829-9

15. Sharifi, T., Mirsalim, M., Soleimanian Gharehchopogh, F., & Mirjalili, S. (2025). Cultural history optimization algorithm: a new human-inspired metaheuristic algorithm for engineering optimization problems. *Neural Computing and Applications*, *37*(25), 21009-21068. https://doi.org/10.1007/s00521-025-11379-z

16. Shrivastava, P., Mandhare, Jadhav, K. P., Ramtirthkar, C. R., Pal, P., Dilorom, O., & Shojonov, M. (2026). Cyber-Physical Healthcare Systems Using AI for Continuous Vital Sign Monitoring. *International Journal of Computer Information Systems and Industrial Management Applications, 18*(1s), 1–19.

17. Wu, E., Kenway, G., Mader, C. A., Jasa, J., & ra Martins, J. R. (2020). pyOptSparse: A Python framework for large-scale constrained nonlinear optimization of sparse systems. *Journal of Open Source Software*, *5*(54), 2564. https://doi.org/10.21105/joss.02564

18. Yang, D., & Liu, X. (2025). Research on large-scale data processing and dynamic content optimization algorithm based on reinforcement learning. *Procedia Computer Science*, *261*, 458-466. https://doi.org/10.1016/j.procs.2025.04.234