CLASSICAL LINEAR PROGRAMMING TECHNIQUES IN MODERN AI:

A REVIEW OF BIG M AND TWO-PHASE METHODS

Monika Sai Murari[1],
[1] Assistant Professor, Department of Statistics, St. Ann's College for Women,
Mehdipatnam, Hyderabad, Telangana, India
ORCID: 0009-0006-4477-8801, +91 8501801452,

P. Deepika[2],
[2] Assistant Professor, Department of Statistics, St. Ann's College for Women,
Mehdipatnam, Hyderabad, Telangana, India
+91 9640418792,

Abstract

Linear programming (LP) has long been a fundamental tool in optimization, with classical methods such as the Big M and Two-Phase techniques playing key roles in solving constrained problems. Although modern artificial intelligence (AI) typically relies on gradient-based and heuristic algorithms, LP methods still offer practical advantages in specific areas where rules, constraints, and interpretability are critical. This review explores the working principles of the Big M and Two-Phase methods and highlights their relevance in AI applications such as rule-based reasoning, feature selection, task scheduling, and hybrid symbolic-ML systems. By comparing these classical techniques with modern optimization approaches, the paper outlines their strengths, limitations, and suitable use cases. The review concludes by discussing the potential for these LP methods to support the development of interpretable and constraint-aware AI systems in the future.

Keywords: Linear Programming, Big M Method, Two Phase Method, Constraint Handling, Artificial Intelligence

1. Introduction

1.1 The Need for Classical Optimization in AI

As artificial intelligence (AI) continues to evolve, the majority of optimization efforts have shifted toward data-driven methods such as stochastic gradient descent (SGD), evolutionary algorithms, swarm intelligence, and reinforcement learning. These techniques are well-suited for training complex, high-dimensional models—such as deep neural networks—where closed-form solutions are impractical and exact constraint satisfaction is not required. However, despite their success in pattern recognition, recommendation systems, and autonomous agents, these modern

optimization tools often function as “black boxes,” making it difficult to impose hard constraints or explain decision boundaries.

In contrast, many emerging AI applications demand structured, rule-based reasoning, hard constraints, and explicitly interpretable outcomes. Domains such as legal-tech, healthcare diagnostics, supply chain scheduling, and fairness-aware systems require strict compliance with external rules and policies. For instance, an AI model used for loan approvals must respect anti-discrimination laws, while a robotic system navigating a factory floor must stay within defined operational zones. These are not merely optimization problems—they are constraint-satisfaction problems, where the solution must meet clearly defined conditions without exception.

This growing need for constrained optimization has reignited interest in classical linear programming (LP) techniques, especially in areas of AI where interpretability, rule enforcement, or deterministic outcomes are crucial. In particular, the Big M method and Two-Phase method offer reliable ways to handle equality and inequality constraints, model logical conditions, and solve feasibility problems where standard machine learning optimizers struggle.

The Big M method introduces artificial variables with large penalty coefficients to enforce constraints, allowing models to simulate logical “if-then” conditions within an LP framework. The Two-Phase method first finds a feasible starting point (Phase I), then solves the actual optimization problem (Phase II), ensuring that the model respects the constraint structure before optimizing for performance. These techniques are highly suitable for AI subfields where constraints are not negotiable, and where solutions must be explainable, reproducible, and auditable.

Moreover, in low-data environments or rule-driven systems where training examples are limited or not statistically representative, classical methods can provide robust and mathematically grounded alternatives to data-heavy learning approaches. By formulating AI tasks as LP problems with structured objective functions and bounded variables, developers can maintain full control over the behavior of the model, something often lost in neural or evolutionary architectures.

Ultimately, the resurgence of interest in classical optimization reflects a broader trend: a shift from purely statistical modeling to hybrid systems that integrate learning with logic, rules, and structure. In this context, Big M and Two-Phase methods remain powerful tools that complement modern AI, enabling models that are not only intelligent but also accountable, interpretable, and compliant with real-world constraints.

1.2 Why Big M and Two-Phase Methods?

The Big M method introduces large penalty constants to incorporate inequality constraints into LP problems, while the Two-Phase method ensures a feasible starting solution by solving an auxiliary LP before the main optimization. These techniques are particularly beneficial in AI contexts where:

- Constraints cannot be violated (e.g., logical rules or regulatory limits).
- Interpretability matters (clear-cut, rule-based decisions are needed).
- Limited data is available, making classical solvers more adequate than data-hungry heuristics.
- Integration with symbolic modules (e.g., combining LP solvers with knowledge bases or logic rules).

1.3 Literature Review

Linear programming (LP) has long been a foundational technique in operations research and optimization. While many modern AI systems rely on gradient-based or heuristic optimization, classical LP methods particularly the Big M and Two-Phase approaches retain critical relevance where hard constraints, logical dependencies, and interpretability are essential. This section reviews key studies that apply or adapt these classical methods to AI-related problems.

Grimstad and Andersson (2019) investigated the use of LP and Mixed-Integer Linear Programming (MILP) in the verification of neural networks with ReLU activations.[1] Their work employs Big M formulations to encode the piecewise-linear nature of ReLU, and they emphasize the impact of "tight" M-values on solver efficiency and numerical stability. This represents a growing trend where LP tools are applied not to train models, but to verify or constrain them post-training, contributing to the safety and robustness of AI systems.

Similarly, Bunel et al. (2020) proposed a unified MILP framework for verifying piecewise-linear neural networks.[2] Their model also builds on Big M constraints to translate neural network operations into an LP-compatible form. These applications demonstrate how classical LP methods support model interpretability and robustness checking goals that are increasingly important in AI safety and regulation.

Beyond neural verification, Two-Phase LP methods have been applied in practical decision-making systems. For instance, Sultana (2022) explored the use of the Two-Phase Simplex method in e-commerce advertising optimization.[3] Their approach ensures feasibility under strict budget and platform constraints before optimizing for reach, highlighting how traditional LP steps align with real-world AI-driven decision pipelines.

In industrial and planning domains, Timpe (2002) [4] demonstrated how MILP combined with constraint programming (CP) could solve complex scheduling problems.[4] It employs Big M-style constraints to model precedence and allocation logic. Such techniques are directly transferable to AI systems focused on autonomous planning, multi-agent scheduling, or logistics.

Koberstein and Suhl (2007) expanded on the stability of LP solvers by developing improved dual Phase-I algorithms for large-scale LP problems.[5] Their research contributes tools that are applicable to AI tasks involving feasibility detection and constraint satisfaction, such as structured decision-making systems or symbolic AI frameworks.

These studies show that while Big M and Two-Phase methods are classical, they are far from obsolete. Instead, they serve specific and growing needs in AI namely: interpretable modeling, constraint handling, symbolic reasoning, and formal verification. However, there is still a lack of unified literature discussing their comparative strengths and best practices within AI pipelines. This review paper aims to bridge that gap.

2. Overview of Big M and Two-Phase Methods

2.1 Basics of Linear Programming (LP)

Linear programming (LP) is a mathematical approach used to determine the optimal outcome of a linear objective function, subject to a set of linear equality and inequality constraints. It is expressed as:

Maximize or Minimize: $Z=c_1x_1+c_2x_2+\ldots+c_nx_n$

Subject to constraints $a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n \leq/ \geq/= b_1$

$a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n \leq/ \geq/= b_2$

$a_{m1}x_1+a_{m2}x_2+\ldots+a_{mn}x_n \leq/ \geq/= b_m$

satisfying non-negative restrictions $x1,x2,\ldots,xn\geq0$

where $b_1, b_2, \ldots, b_m$ are constants

These problems are typically solved using the Simplex method, which moves from one vertex (or corner point) of the feasible region to another until it finds the optimal solution. However, if the LP includes equality constraints or "≥" constraints, a basic feasible solution may not be readily available. In such cases, Big M and Two-Phase methods are introduced to facilitate the solution process.

2.2 The Big M Method

The Big M method incorporates artificial variables into the LP problem when a feasible starting solution is not obvious. It assigns large penalties (denoted as M) to these artificial variables in the objective function, discouraging their presence in the final solution.

Example Concept: Consider objective function $Z= x_1+2x_2$

To convert an equality constraint: $x_1+x_2=5$

into a simplex-compatible form, we introduce an artificial variable A_1: $x_1+x_2+A_1=5$

Then, modify the objective function: $Z = x_1+2x_2+MA_1$

where M is a large positive number. The simplex method will aim to minimize A_1 due to its high penalty, thus forcing it toward zero and maintaining feasibility.

Advantages:

- One-step approach (no separate phase for feasibility).
- Easy to implement for small or moderately sized problems.

Limitations: Choosing the right value for M is critical. If it's too large, numerical instability can occur. If it's too small, the artificial variable might remain in the final solution, violating feasibility.

2.3 The Two-Phase Method

The Two-Phase method avoids the use of an arbitrary penalty constant like M. Instead, it breaks the solution into two parts:

- Phase I: Introduces artificial variables and minimizes their sum to find a feasible starting solution.
- Phase II: Uses that feasible point as a starting solution to solve the original problem.

Procedure:

1. Add artificial variables where needed.
2. Define a new objective: minimize the sum of all artificial variables.
3. Solve Phase I. If the minimum value is zero, proceed to Phase II.
4. Remove artificial variables and solve the original objective.

Advantages:

- More numerically stable than Big M.
- Clearly separates feasibility and optimization.

Limitations:

- Requires solving two problems instead of one (computationally heavier).
- Slightly more complex to implement.

2.4 Relevance in AI Applications

Both Big M and Two-Phase methods have direct applications in modern AI systems where constraints are non-negotiable, such as:

- MILP-based neural network verification (Big M)
- Logic-driven AI systems requiring feasibility before optimization (Two-Phase)
- Rule-based decision-making and explainable AI pipelines
- Task scheduling in robotics and operations where constraints define legality

These methods allow AI engineers to encode domain-specific logic, policies, or operational constraints directly into optimization frameworks, bridging the gap between symbolic reasoning and numeric optimization.

3. Applications in AI Systems

Classical linear programming methods like the Big M and Two-Phase approaches have found renewed utility in modern AI systems that require deterministic behavior, interpretability, and strict constraint adherence. While these methods are not typically used for training large-scale models like deep neural networks, they are essential in several niche yet growing subfields of AI where hard constraints must be encoded and enforced.

3.1 Neural Network Verification and Safety

One of the most impactful applications of the Big M method is in the verification of piecewise-linear neural networks, such as those using ReLU activations. Neural networks can be encoded into a mixed-integer linear programming (MILP) framework using Big M constraints to represent activation behavior.

- Use Case: Verifying whether a neural network will behave correctly under adversarial inputs.

- Example: Grimstad and Andersson (2019) used Big M-based MILP models to check robustness of ReLU networks [1]. Bunel et al. (2018) [2], Fischetti and Jo (2017) [6], and Schwan et al. (2022) [7] extended this to formal safety verification pipelines.
- Impact: Enables safety-critical AI systems (e.g., in autonomous driving or medical AI) to meet formal certification requirements [10].

3.2 Rule-Constrained Decision Making

In domains like legal-tech, policy-driven automation, and ethical AI, models must make decisions under hard rules (e.g., "if age < 18, deny access"). These rules can be embedded into LP or MILP models using Big M-style logic constraints.

- Use Case: Building AI systems for government, HR, or financial compliance.
- Example: Logical implication constraints (e.g., "if A then B") are modeled using MILP formulations with Big M terms, as seen in verification frameworks [10] and logic-enhanced ML models [16].
- Benefit: Guarantees rule compliance and produces auditable, interpretable models[8], [9].

3.3 Feature Selection with Logical Constraints

Feature selection in machine learning can be modeled as a constrained optimization problem. Using LP and Big M, one can enforce rules like mutual exclusivity, required combinations, or budget limits on selected features.

- Use Case: Building compact, interpretable models for healthcare or fraud detection.
- Approach: Formulate selection as 0-1 integer program with Big M constraints [11], [12].
- Advantage: Enables explainable AI by controlling feature inclusion rules.

3.4 Task Scheduling and Resource Allocation

The Two-Phase method is widely used in AI planning, job-shop scheduling, and multi-agent coordination—especially in robotics and operations research settings.

- Use Case: Assigning tasks to robots in a factory while respecting time, energy, and resource constraints.
- Example: Timpe (2002) applied MILP with logical constraints to optimize production scheduling [4], and Sultana (2016) applied Two-Phase LP to advertising budget allocation in a commercial decision model [3].
- How it helps: Phase I ensures feasibility of all constraints before optimizing task distribution in Phase II [5].

3.5 Explainable AI and Logic-Based Inference Systems

Classical LP methods are used in symbolic AI and explainable inference systems, where the goal is to reason over facts and constraints.

- Use Case: Designing transparent AI systems that mimic legal reasoning or policy enforcement.
- Approach: Translate inference logic into LP constraints (e.g., facts, max flow, boolean implications) [10], [13].
- Method: Two-Phase LP ensures feasibility of complex logic chains before computing results [5].

3.6 Hybrid Symbolic–Neural Systems

In emerging neuro-symbolic architectures, LP solvers are integrated into deep learning pipelines to enforce external rules or structural constraints.

- Use Case: Enforcing high-level rules in a reinforcement learning environment.
- Role of LP: Acts as a logic-checking layer between neural prediction and action execution [15].
- Technique: LP solver (with Big M) evaluates symbolic logic during or after learning, often using pre-processing or model reduction techniques to speed up solving [14], [16].

AI Domain	LP Method Used	Purpose
Neural Verification	Big M (MILP)	Encode ReLU logic, ensure robustness
Rule-Constrained Decision	Big M	Enforce logic/policy rules
Feature Selection	Big M	Select features under logical constraints
Scheduling & Planning	Two-Phase	Ensure feasibility, then optimize
Explainable Logic Models	Two-Phase / Big M	Model and evaluate structured inference
Neuro-Symbolic Systems	Big M	Enforce symbolic rules within learning

4. Comparative Analysis of Classical vs. Modern Optimization Methods

Modern AI systems are predominantly trained using gradient-based or heuristic optimization techniques such as stochastic gradient descent (SGD), Adam optimizer, genetic algorithms (GA), and particle swarm optimization (PSO). While these methods are highly effective for learning from large-scale data and optimizing non-linear, high-dimensional functions, they do not provide built-in mechanisms to enforce hard constraints, logical dependencies, or feasibility guarantees. This is where classical linear programming techniques such as Big M and Two-Phase methods maintain their relevance.

4.1 Transparency and Interpretability

- Classical LP Methods: Solutions derived from Big M and Two-Phase formulations are inherently interpretable. Each variable, constraint, and result has a defined role, making them suitable for applications that demand transparency such as law, healthcare, and finance.
- Modern AI Optimizers: Neural networks optimized using SGD or Adam are often considered black-box models. While accurate, they provide little insight into why or how a decision was made, making them less suitable for regulated environments.

4.2 Constraint Handling

- Big M and Two-Phase: These methods are designed to solve constrained problems directly. They can handle equality, inequality, and logical constraints explicitly, which is essential in symbolic reasoning and structured planning.
- SGD and Evolutionary Algorithms: Constraints are typically handled indirectly (e.g., through penalty functions or soft constraints). There is no guarantee that constraints will be strictly satisfied, especially under noise or limited data.

4.3 Applicability to Data-Scarce Environments

- Classical LP: Effective even in low-data or rule-driven scenarios where the problem can be fully described using mathematical constraints. They do not require training data to perform optimization.
- Modern Methods: Require large datasets for effective training and generalization. In data-scarce conditions, performance degrades or models overfit.

4.4 Computational Efficiency and Scalability

- Big M: Solves small to medium LP problems efficiently but can face numerical instability with large M values or when problem dimensions increase.
- Two-Phase: More stable for large-scale LP problems but involves two separate stages, increasing computation time.
- Modern Methods: Gradient-based techniques like Adam scale well with millions of parameters (e.g., in deep learning). However, they trade off precision and constraint compliance.

4.5 Flexibility and Learning Capability

- Modern Optimizers excel at pattern recognition, function approximation, and reinforcement learning, where the optimization landscape is non-linear and non-convex.

- LP Methods are restricted to linear models and constraints. While highly reliable within their scope, they cannot approximate complex functions like neural networks.

4.6 Integration Possibilities

Rather than viewing classical and modern methods as mutually exclusive, many recent AI systems combine both. For example:

- A neural network predicts likely outcomes, and a LP model refines the decision under hard rules.
- Reinforcement learning policies are filtered using LP solvers to enforce safety limits.
- Feature selection or model compression is handled using MILP with Big M constraints before model training.

These hybrid strategies open a path for constraint-aware AI, balancing learning power with rule compliance and interpretability.

5. Conclusion and Future Work

5.1 Conclusion

Classical linear programming techniques, particularly the Big M and Two-Phase methods, have demonstrated lasting value in artificial intelligence systems where constraint satisfaction, transparency, and deterministic behavior are non-negotiable. Although modern AI has largely shifted toward data-driven, gradient-based optimization, this review highlights how classical LP methods continue to support critical applications such as neural network verification, symbolic reasoning, rule-constrained decision-making, and task scheduling.

The Big M method is particularly useful for encoding logical rules and constraints into optimization frameworks commonly applied in mixed-integer linear programming (MILP) formulations. Meanwhile, the Two-Phase method provides a numerically stable approach for finding feasible solutions in complex constrained systems. Their inherent interpretability and formal structure make them indispensable in AI domains that prioritize fairness, compliance, and auditability.

As AI systems become more integrated into sensitive and regulated environments, the need for explainable and constraint-aware decision-making will continue to grow. Classical LP methods provide a mathematical backbone for such systems, ensuring that they remain accountable and safe.

5.2 Future Work

While this review has outlined the relevance and utility of Big M and Two-Phase methods in AI, several open areas remain for future exploration:

Scalability Improvements: Research is needed to improve the scalability of LP solvers for handling high-dimensional AI problems, especially when integrating logic-based constraints in large models.

Hybrid Architectures: Future systems can more deeply integrate LP solvers with neural networks, reinforcement learning agents, and symbolic logic frameworks to enforce rules in real time.

Numerical Stability Enhancements: Developing adaptive methods to dynamically tune or eliminate large penalty values (as in Big M) could improve solver performance and reliability.

Explainability Tools: LP-based methods could serve as the foundation for new explainable AI models that provide traceable decision paths, especially when fairness and regulatory compliance are required.

Application Expansion: Additional use cases in healthcare, legal-tech, ethical AI, and smart infrastructure systems can benefit from further empirical research on classical LP integration.

In conclusion, Big M and Two-Phase methods are not relics of the past but rather underutilized enablers of modern, constraint-conscious AI. Their systematic, rule-driven nature makes them vital tools for building transparent, robust, and legally compliant intelligent systems in the years to come.

References

1. B. Grimstad and H. Andersson, “RELU Networks as Surrogate Models in Mixed-Integer Linear Programs,” Computers & Chemical Engineering, vol. 131, 2019, doi: 10.1016/j.compchemeng.2019.106580
2. R. Bunel, I. Turkaslan, P. H. S. Torr, P. Kohli, and M. P. Kumar, “A Unified View of Piecewise-Linear Neural Network Verification,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 31, 2018, pp. 4790–4801, doi: 10.48550/arXiv.1711.00455
3. N. Sultana, “Application of the Two-Phase Simplex Method for Optimal Advertisement Decision: A Study on an E-commerce Company,” IIUC Business Review, vol. 5, Dec 2016, pp. 59–76, doi: 10.3329/iiucs.v14i1.37651.

4. C. Timpe, “Solving planning and scheduling problems with combined integer and constraint programming,” OR Spectrum, vol. 24, no. 4, pp. 431–448, 2002, doi: 10.1007/s00291-002-0107-1.
5. A. Koberstein and L. Suhl, “An advanced implementation of a dual simplex Phase-I algorithm,” Computational Optimization and Applications, vol. 36, nos. 2–3, pp. 319–348, 2007, doi: 10.1007/s10589-007-9022-3.
6. M. Fischetti and J. Jo, “Deep Neural Networks as 0-1 Mixed Integer Linear Programs: A Feasibility Study,” CoRR, vol. abs/1712.06174, 2017, doi: 10.48550/arXiv.1712.06174.
7. R. Schwan, C. N. Jones & D. Kuhn, “Stability Verification of Neural Network Controllers using Mixed-Integer Programming,” arXiv, vol. abs/2206.13374, 2022, doi: 10.48550/arXiv.2206.13374.
8. R. Ehlers, “Formal Verification of Piece-Wise Linear Feed-Forward Neural Networks,” arXiv preprint arXiv:1705.01320, 2017, doi: 10.48550/arXiv.1705.01320.
9. C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer, “Algorithms for Verifying Deep Neural Networks,” arXiv preprint arXiv:1903.06758, 2019, doi: 10.48550/arXiv.1903.06758.
10. G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, “Reluplex: A Simplex-Based SMT Solver for Verifying Deep Neural Networks,” Formal Methods in System Design, vol. 60, no. 1, 2022, pp. 87–115, doi: 10.1007/s10703-021-00363-7.
11. S. Maldonado, L. I. Martínez-Merino, and A. M. Rodríguez-Chía, “Feature Selection for Support Vector Machines via Mixed Integer Linear Programming,” Information Sciences, vol. 279, pp. 163–175, 2014, doi: 10.1016/j.ins.2014.03.110.
12. F. Labbé, L. I. Martínez-Merino, and A. M. Rodríguez-Chía, “Mixed Integer Linear Programming for Feature Selection in Support Vector Machine,” arXiv preprint arXiv:1808.02435, 2018, doi: 10.48550/arXiv.1808.02435.
13. J. Zhang, C. Liu, J. Yan, X. Li, H.-L. Zhen, and M. Yuan, “A Survey for Solving Mixed Integer Programming via Machine Learning,” CoRR, vol. abs/2203.02878, 2022.
14. L. Scavuzzo, K. Aardal, A. Lodi, and N. Yorke-Smith, “Machine learning augmented branch and bound for mixed integer linear programming,” Mathematical Programming, 2024, doi: 10.1007/s10107-024-02130-y.
15. Y. Li, C. Chen, J. Li, J. Duan, X. Han, T. Zhong, V. Chau, W. Wu, and W. Wang, “Fast and Interpretable Mixed-Integer Linear Program Solving by Learning Model Reduction,” AAAI Conference on Artificial Intelligence, vol. 39, no. 25, pp. 27090–27098, 2025, doi: 10.1609/aaai.v39i25.34916.
16. A. Kiruluta and A. Lemos, “Unsupervised Machine Learning Hybrid Approach Integrating Linear Programming in Loss Function: A Robust Optimization Technique,” arXiv preprint arXiv:2408.09967, 2024, doi:10.48550/arXiv.2408.09967