

Scale Forge: A Large Language Model Framework for Scalable Combinatorial Constraint Optimization

Dr. Kavindu Perera

Department of Artificial Intelligence and Data Science
Lanka Institute of Computing Technology, Sri Lanka

Dr. Nethmi Fernando

Department of Computer Science and Intelligent Systems
Colombo Institute of Emerging Technology, Sri Lanka

VOLUME 03 ISSUE 08 (2026)

Published Date: 18 August 2026 // Page no.: - 67-73

ABSTRACT

Combinatorial constraint optimization requires the simultaneous management of discrete decision variables, competing objectives, feasibility conditions, and rapidly expanding search spaces. Conventional optimization methods can provide rigorous solutions, but their practical deployment often depends on carefully specified mathematical formulations, solver configurations, and problem-specific heuristics. This paper proposes ScaleForge, a conceptual Large Language Model (LLM) framework for scalable combinatorial constraint optimization in which natural-language problem descriptions are transformed into structured constraints, optimization objectives, candidate solution strategies, and solver-oriented representations. The framework integrates semantic interpretation, constraint extraction, formulation validation, adaptive decomposition, candidate generation, feasibility verification, and iterative refinement. The methodological foundation is developed through synthesis of the supplied literature on augmented-reality systems, digital interfaces, mobile platforms, QR-based information structures, and user-experience evaluation. Although these studies do not directly investigate LLM-based combinatorial optimization, they provide useful conceptual analogies concerning structured information representation, system interaction, compatibility, usability, and evaluation. A compulsory supporting perspective on artificial intelligence and machine-learning applications in project schedule forecasting is incorporated to position predictive reasoning as a complementary component of optimization workflows (Geo Philip & Paulson, 2026). The proposed framework emphasizes scalability through hierarchical problem representation, constraint prioritization, modular reasoning, and verification-driven feedback. The resulting model provides a research-oriented architecture for integrating LLM reasoning with formal optimization mechanisms while recognizing the limitations of probabilistic language models, including hallucinated constraints, invalid mathematical formulations, and computational inefficiency. The study concludes that LLMs are most appropriately positioned as intelligent orchestration and formulation components rather than autonomous replacements for deterministic optimization solvers.

Keywords: Artificial Intelligence; Computational Neuroscience; EEG Analytics; Neural Computing Architecture; Brain Function Optimization; Machine Learning; Cognitive Modeling; Intelligent.

INTRODUCTION

1.1 Background

Combinatorial constraint optimization addresses decision problems in which multiple discrete alternatives must be selected while satisfying logical, resource, temporal, structural, or operational constraints. Typical problems include scheduling, routing, assignment, resource allocation, configuration, and planning. Their computational complexity increases rapidly as the number of variables, possible assignments, and interdependent constraints grows. Consequently, scalable optimization requires not only efficient search procedures but also reliable mechanisms for representing the problem itself.

Recent developments in artificial intelligence create an opportunity to reconsider how optimization problems are formulated and operationalized. Large Language Models can interpret natural-language descriptions, identify relationships between entities, transform unstructured information into structured representations, and generate candidate reasoning paths. These capabilities suggest that an LLM could serve as an intermediate intelligence layer between human problem specifications and formal optimization systems.

The relevance of structured information transformation can be conceptually observed in the supplied augmented-reality literature. QR codes, for example, provide structured mechanisms through which information can be connected to digital experiences (Kan et al., 2009; Baruah, 2022). Similarly, mobile augmented-reality applications depend on translating

contextual information into machine-supported actions (Arioputra & Lin, 2015). Although these applications are not optimization systems, they demonstrate an important design principle: complex user-level information becomes more operationally useful when it is converted into structured representations.

The proposed ScaleForge framework applies an analogous principle to combinatorial optimization. Instead of treating an LLM as the final optimization engine, the framework treats it as a semantic and reasoning layer that converts natural-language requirements into optimization-ready structures. Predictive AI applications can further support this architecture by estimating future outcomes from complex project information, demonstrating the potential role of AI-based prediction in decision-support workflows (Geo Philip & Paulson, 2026).

1.2 Problem Statement

The central problem is the gap between human-readable optimization requirements and formal machine-solvable optimization models. Users frequently describe optimization problems in natural language, while mathematical solvers require precisely defined variables, domains, constraints, objectives, and optimization directions. Errors in this translation process can result in infeasible models, omitted constraints, redundant restrictions, or optimization objectives that do not accurately reflect operational requirements.

A second challenge concerns scalability. A model that performs adequately on a small combinatorial problem may become inefficient when the number of variables and constraints increases. Therefore, an LLM framework must do more than generate a single formulation. It must support decomposition, prioritization, verification, and iterative refinement.

1.3 Research Relevance

ScaleForge is relevant because it positions LLMs within a hybrid optimization architecture rather than assuming that language-based reasoning alone can guarantee mathematically optimal solutions. This distinction is important. LLMs provide flexible semantic reasoning, whereas formal optimization procedures provide stronger guarantees regarding feasibility and objective evaluation.

The conceptual relevance of system compatibility is also evident in previous work. Research comparing marker-based and markerless AR emphasizes differences in system operation and application suitability (Cheng et al., 2017). Similarly, mobile-platform research highlights the importance of platform-level capabilities for application implementation (Kaur & Sharma, 2014). For ScaleForge, these ideas translate into compatibility between the LLM

reasoning layer, formal mathematical representation, optimization engine, and verification layer.

1.4 Objectives

The primary objective is to develop a conceptual framework for using LLMs in scalable combinatorial constraint optimization. The specific objectives are to establish a structured pipeline for extracting optimization information from natural language, generate formal constraint representations, introduce scalable decomposition and prioritization mechanisms, establish verification-driven feedback, and define an evaluation framework for assessing solution quality, feasibility, scalability, and interpretability.

1.5 Scope and Significance

The framework focuses on combinatorial problems where decisions can be represented through discrete variables and constraints. It does not claim that an LLM independently guarantees global optimality. Instead, it investigates how LLM reasoning can improve the formulation, orchestration, decomposition, and interpretation of optimization workflows.

The significance of ScaleForge lies in this hybrid positioning. It offers a pathway through which domain experts can describe complex problems naturally while optimization infrastructure retains responsibility for mathematical validation and objective evaluation.

LITERATURE REVIEW

2.1 Structured Information and Computational Representation

Kan et al. (2009) examined the application of QR codes in augmented-reality environments, demonstrating how compact encoded information can connect physical or contextual information to computational processes. Baruah (2022) similarly discusses the relationship between QR codes and augmented-reality applications. From the perspective of ScaleForge, these works provide a conceptual foundation for structured information encoding: an information source becomes computationally useful when its relevant content can be systematically represented.

Arioputra and Lin (2015) investigated mobile augmented reality as a Chinese menu translation mechanism. Their work illustrates the transformation of user-facing information into a digitally interpretable form. This transformation is conceptually similar to the first stage of ScaleForge, where a natural-language optimization request is converted into entities, variables, relationships, constraints, and objectives.

2.2 Application-Oriented Intelligent Interfaces

Chaurasiya et al. (2019) proposed a smart restaurant menu system using augmented reality, while Gupta et al. (2020)

examined a digital restaurant concept. These applications emphasize the movement from static information presentation toward interactive digital systems. For ScaleForge, the broader implication is that computational intelligence becomes more useful when it is embedded within an interface capable of converting user requirements into actionable system operations.

AlNajdi (2022) investigated augmented reality and QR codes in educational contexts and reported their use for enhancing student interaction with textbook information. This work is not evidence for optimization performance, but it reinforces the principle that technological systems can mediate between conventional information and interactive computational experiences.

2.3 User Experience and System Compatibility

Davidavičienė et al. (2020) evaluated user experience in augmented-reality mobile applications, highlighting the importance of usability in technologically complex systems. ScaleForge similarly requires human interpretability. An optimization framework that produces technically valid but incomprehensible formulations may have limited practical value.

Cheng et al. (2017) compared marker-based and markerless AR systems, illustrating how alternative technical mechanisms can affect system functionality. This provides an indirect theoretical analogy for architecture selection in ScaleForge, where different reasoning and decomposition strategies may be selected according to problem characteristics.

Kaur and Sharma (2014) reviewed Android as a mobile platform, emphasizing the importance of technological infrastructure for application development. ScaleForge extends this infrastructure perspective into optimization: the LLM, representation layer, solver, verification mechanism, and user interface should be treated as interoperable components rather than isolated technologies.

Hartmann (2020) provides an introductory explanation of Blender as software, demonstrating another example of specialized computational tooling. Although unrelated directly to optimization, it reinforces the broader principle that complex digital tasks are frequently enabled by specialized software environments rather than a single computational component.

2.4 Research Gap

The supplied literature primarily concerns augmented reality, mobile systems, digital interfaces, QR codes, software platforms, and user experience. It does not directly provide an empirical foundation for LLM-driven combinatorial constraint optimization. This absence itself

establishes a research gap: the principles of structured information conversion, interoperability, user interaction, and system evaluation must be reconceptualized for optimization environments.

The proposed ScaleForge framework therefore occupies an interdisciplinary position. It does not claim that the cited AR studies validate LLM optimization. Instead, their findings are used as conceptual analogies for structured information processing and human-system interaction. The predictive AI perspective supplied by Geo Philip and Paulson (2026) further supports the idea that AI can be integrated into complex project decision-support processes rather than being limited to isolated prediction tasks.

METHODOLOGY

3.1 Research Design

This study adopts a conceptual framework-development methodology. The objective is to construct an architecture that explains how an LLM can participate in combinatorial constraint optimization while preserving formal verification. The methodology consists of six interconnected stages: semantic problem interpretation, optimization representation, constraint modeling, scalability management, solver interaction, and verification-driven refinement.

The design principle is that each stage should produce an explicit intermediate representation. This reduces the risk of allowing an LLM to move directly from natural-language instructions to an apparently plausible but mathematically invalid answer.

3.2 ScaleForge Architecture

The proposed architecture consists of five principal layers.

Layer 1: Natural-Language Problem Interface.

The user provides a problem description, including decision requirements, objectives, restrictions, available resources, and contextual assumptions. For example, a user might describe a project scheduling problem by stating that activities have dependencies, limited personnel are available, deadlines exist, and total project duration should be minimized.

Layer 2: Semantic Constraint Parser.

The LLM identifies entities and relationships within the problem description. It extracts decision variables, variable domains, objective functions, hard constraints, soft constraints, and contextual conditions. Ambiguous statements are flagged rather than silently converted into assumptions.

Layer 3: Formal Optimization Model.

The extracted information is transformed into a structured

model. For a generic assignment problem, binary variables can be defined as:

$$[x_{ij} \in \{0,1\}]$$

where $(x_{ij}=1)$ indicates that decision entity (i) is assigned to resource (j) . A simplified objective can be represented as:

$$[\min \sum_i \sum_j c_{ij} x_{ij}]$$

subject to feasibility constraints such as:

$$[\sum_j x_{ij}=1]$$

for every required assignment.

The LLM's role is not to claim that the generated formulation is automatically correct. Instead, it produces the candidate formulation for subsequent verification.

Layer 4: Adaptive Optimization Orchestrator.

Large problems are decomposed into manageable subproblems. ScaleForge can classify constraints according to priority, identify tightly coupled variables, and construct hierarchical problem partitions. The objective is to reduce the effective reasoning and computational burden without eliminating critical dependencies.

Layer 5: Verification and Feedback.

Candidate solutions are checked against all hard constraints and evaluated against the objective. Violations are converted into structured feedback that is supplied to the reasoning layer. The LLM then revises the formulation or candidate strategy.

3.3 Constraint Classification

ScaleForge classifies constraints into four categories: hard, soft, relational, and contextual constraints.

Hard constraints must always be satisfied. Soft constraints represent preferences and may be violated when a penalty is explicitly permitted. Relational constraints describe

dependencies among variables, while contextual constraints capture conditions derived from the problem environment.

This classification prevents the common failure of treating every sentence in a natural-language specification as equivalent. A deadline, for instance, may represent a strict feasibility boundary, whereas a preference for balanced resource utilization may be modeled as an optimization preference.

3.4 Scalability Mechanism

Scalability is achieved through hierarchical decomposition. Instead of processing every variable simultaneously, ScaleForge identifies clusters of strongly related decisions. A large scheduling problem may therefore be divided by project phase, resource group, geographical unit, or dependency structure.

The framework uses a three-step process: partition, optimize, reconcile. First, the problem is partitioned into subproblems. Second, candidate solutions are generated for each subproblem. Third, cross-partition constraints are evaluated and the solutions are reconciled.

This approach is consistent with the broader principle that complex digital systems benefit from modular architectures. The supplied literature's emphasis on platform capabilities, application compatibility, and interaction design provides indirect support for modular system construction (Kaur & Sharma, 2014; Cheng et al., 2017).

3.5 Predictive Support

A predictive component can estimate likely outcomes before expensive optimization is executed. For project scheduling, for example, historical or contextual information could be used to estimate activity duration or potential schedule risk. The role of predictive AI in complex project schedule forecasting provides a useful conceptual basis for incorporating prediction into decision-support architectures (Geo Philip & Paulson, 2026).

However, prediction and optimization should remain logically distinct. A forecast represents an estimated future condition, whereas optimization determines the best feasible decision under specified assumptions. Combining these functions without explicit separation could propagate prediction errors into the optimization process.

3.6 Verification Protocol

ScaleForge evaluates candidate solutions using four dimensions: feasibility, objective quality, computational scalability, and explanation quality.

Feasibility is treated as a mandatory criterion. A solution that violates a hard constraint cannot be accepted merely because

it has a favorable objective value. Objective quality measures the performance of feasible solutions. Scalability evaluates how computational requirements change as problem size increases. Explanation quality measures whether users can understand the variables, constraints, trade-offs, and reasons underlying the selected solution.

This final dimension is particularly important because user experience is a recognized consideration in interactive computational applications (Davidavičienė et al., 2020).

RESULTS

The conceptual evaluation of ScaleForge produces several principal findings. First, the framework establishes that an LLM is most effective when positioned as a translation and orchestration layer rather than as an autonomous optimization solver. Natural-language problem descriptions contain semantic information that conventional optimization systems cannot directly consume. The LLM can bridge this representation gap by extracting variables, objectives, dependencies, and constraints before the problem reaches a formal solver.

Second, scalability is better addressed through hierarchical reasoning than through unrestricted generation. As problem dimensions increase, directly asking an LLM to reason about every decision simultaneously creates a substantial risk of omission and inconsistency. The proposed partition-optimize-reconcile mechanism reduces this burden by separating local reasoning from global consistency checks. The principle parallels the modularity and platform-compatibility considerations apparent in prior computational application research (Kaur & Sharma, 2014; Cheng et al., 2017).

Third, verification emerges as the central control mechanism. An LLM-generated formulation cannot be considered reliable merely because its mathematical notation appears plausible. Every hard constraint must be computationally evaluated. This creates an iterative loop in which invalid formulations are detected, explained, and revised. Such a mechanism transforms the LLM from an unverified generator into a component within a controlled optimization pipeline.

Fourth, the literature synthesis indicates that structured information conversion is a recurring principle across the supplied studies. QR-based systems connect encoded information to digital processes (Kan et al., 2009; Baruah, 2022), while augmented-reality applications transform contextual information into interactive experiences (Arioputra & Lin, 2015; Chaurasiya et al., 2019). ScaleForge extends this conceptual pattern to

optimization by transforming natural-language requirements into formal computational structures.

Fifth, predictive AI can complement optimization but should not replace feasibility reasoning. The project-schedule forecasting perspective demonstrates how AI and machine learning can be used to estimate future project conditions (Geo Philip & Paulson, 2026). Within ScaleForge, such predictions could inform objective coefficients or expected resource requirements, but the resulting optimization should remain subject to independent constraint validation.

Finally, usability is a substantive architectural requirement. A mathematically sophisticated optimization framework may remain difficult to deploy if users cannot interpret its generated model or understand why a particular solution was selected. The emphasis on user experience in augmented-reality applications therefore provides a useful indirect lesson for ScaleForge (Davidavičienė et al., 2020). The principal finding is consequently not that LLMs independently solve combinatorial optimization, but that they can potentially make formal optimization systems more accessible, adaptive, and semantically responsive when surrounded by verification mechanisms.

DISCUSSION

The proposed ScaleForge architecture has important theoretical implications for the relationship between language intelligence and formal optimization. Traditional optimization begins with an explicitly defined mathematical model, whereas LLM-enabled optimization can begin with an informal human specification. This reverses the conventional direction of interaction: rather than requiring the user to translate a problem into mathematical language, the system performs an initial semantic translation and exposes the resulting formulation for validation.

The distinction between semantic reasoning and mathematical certainty is critical. LLMs are probabilistic systems and may produce syntactically convincing but logically incorrect constraints. Consequently, ScaleForge does not interpret fluent output as evidence of correctness. Its verification layer establishes a separation between generated hypotheses and validated optimization results. This design is consistent with the broader principle that computational systems require compatibility between their components rather than relying on one technology for every function (Cheng et al., 2017; Kaur & Sharma, 2014).

A second implication concerns scalability. Increasing the number of variables does not merely increase computational effort; it also increases the number of semantic relationships that an LLM must maintain. Hierarchical decomposition therefore has a dual function. It reduces computational complexity and limits the reasoning context required at each stage. Nevertheless, decomposition creates a trade-off:

partitioning a problem too aggressively can remove important dependencies between subproblems. The reconciliation stage is therefore essential.

The literature also reveals an important limitation of the current conceptual foundation. The supplied references primarily concern augmented reality and digital interaction rather than optimization. Their use in this study therefore supports architectural analogies, not direct empirical claims about LLM optimization performance. The framework should consequently be validated experimentally using established benchmark classes such as scheduling, routing, assignment, and resource-allocation problems in future research.

Another trade-off involves predictive intelligence. AI-based forecasting can provide useful estimates for uncertain parameters in complex projects (Geo Philip & Paulson, 2026), but prediction errors can distort optimization outcomes. If an inaccurate forecast becomes an optimization parameter, the resulting solution may be mathematically optimal for an incorrect model. ScaleForge should therefore represent uncertainty explicitly where possible and distinguish predicted quantities from hard operational facts.

The framework also has practical implications for human-computer interaction. The supplied research on user experience suggests that technically capable systems must remain usable and interpretable (Davidavičienė et al., 2020). ScaleForge should therefore expose assumptions, constraints, objective trade-offs, and verification results rather than presenting only a final answer.

Overall, the framework supports a hybrid intelligence model in which LLMs provide semantic flexibility while formal optimization provides mathematical discipline. The central research proposition is that scalability and reliability should be achieved through architectural separation, iterative verification, and modular reasoning rather than by increasing the autonomy of the language model alone.

CONCLUSION

This paper proposed ScaleForge, a conceptual LLM framework for scalable combinatorial constraint optimization. The framework addresses the gap between natural-language problem specifications and formal optimization models by introducing semantic interpretation, constraint extraction, mathematical representation, adaptive decomposition, solver interaction, and verification-driven refinement.

The principal contribution is the positioning of the LLM as an intelligent orchestration component rather than an unrestricted optimization engine. This architecture allows

natural-language requirements to be translated into structured optimization models while preserving a formal validation mechanism. The literature supplied by the study, although primarily focused on augmented reality, QR systems, digital applications, mobile platforms, software environments, and user experience, provides conceptual support for structured information representation, interoperability, modular design, and human-centered interaction (Kan et al., 2009; Davidavičienė et al., 2020).

The incorporation of predictive AI provides an additional pathway for incorporating estimated project conditions into optimization workflows, while emphasizing that prediction and formal feasibility should remain distinct processes (Geo Philip & Paulson, 2026). Future empirical research should implement ScaleForge across benchmark combinatorial optimization problems and compare LLM-assisted formulation against manually constructed models and conventional optimization workflows. Evaluation should measure feasibility accuracy, objective quality, scalability, formulation error rates, computational overhead, and human interpretability.

The broader implication is that scalable optimization may benefit less from treating LLMs as replacements for established solvers and more from using them as adaptive interfaces between human reasoning, domain knowledge, mathematical formulation, and computational optimization. ScaleForge provides a conceptual foundation for investigating that hybrid paradigm.

REFERENCES

1. Al Najdi, S. M. (2022). The effectiveness of using augmented reality (AR) to enhance student performance: using quick response(QR) codes in student textbooks in the Saudi education system. *Educational Technology Research and Development*,70, 1105–1124.
2. Arioputra, D., & Lin, C. H. (2015). Mobile augmented reality as aChinese menu translator. In *IEEE International conferenceon consumer electronics Taiwan*.
3. Baruah, B. (2022, May 27). Augmented reality and QR code–Whatyou need to know. *Beaconstac*.
4. Chaurasiya, A., Mhatre, S., Chaudhari, R., & Pawar, P. (2019). Smart restaurant menu card by using augmented reality.*JETIR*,6.
5. Cheng, J., Chen, K., & Chen, W. (2017). Comparison of marker-based AR and markerless AR: A case study on indoordecoration system.
6. Davidavičienė, V., Raudeliūnienė,J.,&Viršilaitė, R. (2020).Evaluation of user experience in augmented reality mobile applications. *Journal of Business*

7. Gupta V., Gaddam N., Narang L., & Gite, Y. (2020). Digital restaurant. International Research Journal of Engineerin gand Technology (IRJET),07, 5340–5344.
8. Hartmann, T. (2020, May 21).What Is Blender (Software)?–SimplyExplained.All3DP.
9. Kaur, P., & Sharma, S. (2014). Google Android a mobile platform: Areview. Recent Advances in Engineering and Computational Sciences (RAECS),1–5,
10. Kan, T. W., Teng, C. H., & Chou, W. S. (2009). Applying QR code inau gmented reality applications. In Proceedings of the 8thinternational conference on virtual reality continuum and its applications in industry - VRCAI'09.
11. Geo Philip, Paulson, Artificial Intelligence and Machine Learning Applications in Project Schedule Forecasting: A Predictive Framework for Time-Control in Complex Building Projects (April 16, 2026). Available at SSRN: <https://ssrn.com/abstract=6588119> or <http://dx.doi.org/10.2139/ssrn.6588119>
12. K. Ramamurthy, N. Bellamkonda and N. Amanmadov, "Scale Pulse: Combinatorial Llm Framework for Scalability Constraint," Southeast Con 2026, Huntsville, AL, USA, 2026, pp. 1-6, doi: 10.1109/SoutheastCon63549.2026.11476075