Proceedings of the Design Society, Volume 5: ICED25
https://doi.org/10.1017/pds.2025.10234

Large language models for combinatorial optimization of design structure matrix

Shuo Jiang ✉, Min Xie and Jianxi Luo

City University of Hong Kong, Hong Kong

✉ shuo.jiang@cityu.edu.hk

ABSTRACT: Combinatorial optimization (CO) is essential for improving efficiency and performance in engineering applications. Traditional algorithms based on pure mathematical reasoning are limited and incapable to capture the contextual nuances for optimization. This study explores the potential of Large Language Models (LLMs) in solving engineering CO problems by leveraging their reasoning power and contextual knowledge. We propose a novel LLM-based framework that integrates network topology and contextual domain knowledge to optimize the sequencing of Design Structure Matrix (DSM) —a common CO problem. Our experiments on various DSM cases demonstrate that the proposed method achieves faster convergence and higher solution quality than benchmark methods. Moreover, results show that incorporating contextual domain knowledge significantly improves performance despite the choice of LLMs.

KEYWORDS: large language models, combinatorial optimization, systems engineering (SE), concurrent engineering (CE), design management

1. Introduction

Combinatorial optimization (CO) problems are ubiquitous across fields, where finding an optimal solution from a finite set often drives improvements in efficiency, cost, and performance (Korte, et al. 2011). For instance, applications such as DNA barcoding and DNA assembly in synthetic biology (Naseri and Koffas 2020), as well as job scheduling in manufacturing (Xidias and Azariadis 2019), rely heavily on effective CO solutions. However, due to their NP-hard nature, these problems present substantial challenges, especially as complexity increases with larger problem sizes and more intricate dependencies. Traditionally, CO problems in engineering are usually approached through the following process: the problem is first modelled mathematically, then solved using specific algorithms or heuristics, and finally interpreted within the context of practical engineering (Pinedo 2012). This separation of problem-solving and interpretation stages is limited and incapable to capture the contextual nuances necessary for optimization of real-world problems.

Recent advancements in Large Language Models (LLMs) have demonstrated their powerful capabilities in natural language generation, semantic understanding, instruction following, and complex reasoning (Wei, et al. 2022) (Chang, et al. 2024). Furthermore, pioneering studies have shown that LLMs can be used for continuous and concrete optimization (Romera-Paredes, et al. 2024) (Yang, et al. 2024) (Liu, et al. 2024). For instance, researchers from DeepMind utilized LLMs as optimizers and evaluated their effectiveness on classic CO problems (Yang, et al. 2024), such as Traveling Salesman Problems (TSP). Additionally, prior studies also highlight that LLMs possess extensive domain knowledge pretrained across a wide range of engineering-related data, which enhances their applicability in engineering fields (Zhu and Luo 2023) (Makatura, et al. 2024) (Jiang and Luo 2024) (Mei, et al. 2024). Therefore, the ability of LLMs to combine mathematical and semantic reasoning, along with their possession of extensive knowledge, motivated us to explore their potential for solving engineering CO problems while integrating contextual domain knowledge relevant to their network typology. Our hypotheses are: (1)

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

LLMs can be effectively applied to solve CO problems in engineering, and (2) incorporating contextual domain knowledge can further enhance LLM performance by supporting mathematical reasoning with semantic insights. This paradigm, which leverages both semantic and mathematical reasoning, introduces a novel approach to combinatorial optimization that traditional pure mathematical methods cannot achieve for empirical problems. On this basis, we propose a novel LLM-based framework that integrates both network topology and domain context into the optimization process.

To evaluate our proposed method, we focus on the Design Structure Matrix (DSM) sequencing task, as an example of CO problems. DSM is a modelling tool in engineering design, which represents dependency relationships among tasks or components within a system (Steward 1981). Reordering the node sequence of DSMs can significantly reduce feedback loops and improve modularization (Choi 2011) (Eppinger and Browning 2012). The DSM sequencing is also an NP-hard problem, and traditional methods typically approach it using heuristics-based algorithms (Eppinger, Whitney, et al. 1994) (Qian, et al. 2011). Figure 1 illustrates a design activity DSM before and after sequencing (Amen, Rask and Sunnersjö 1999). In this paper, we conduct extensive experiments on various DSM cases to demonstrate that our LLM-based method achieves better convergence speed and solution quality compared to benchmark methods. Notably, results show that incorporating contextual domain knowledge significantly enhances the performance despite the choice of backbone LLMs.

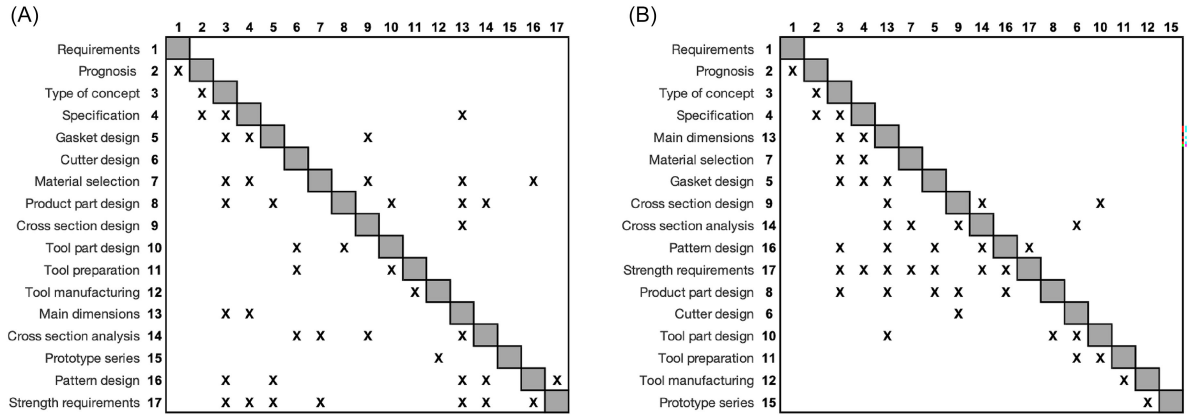


Figure 1. Illustration of a design activity DSM: (A) Pre-sequencing; (B) Post-sequencing

2. Methodology

In this section, we introduce our proposed LLM-based framework for solving CO problems. The framework is designed to harness the generative and reasoning powers of LLMs in combination with domain knowledge and objective evaluation. The framework begins with the initialization of a solution randomly sampled in the total solution space. Each solution is evaluated based on predefined criteria by an evaluator, which quantifies the quality of a solution. Using this evaluation, the framework iteratively updates the solution base through few-shot learning and suggesting new candidates, guided by crafted prompts that include both network information in mathematical form and domain knowledge in natural language description. The newly generated solutions are appended into the solution base, together with their evaluation results. When the iteration time is reached, the solution base returns the best one as the final output. In following, we focus on DSM sequencing as a common CO problem to illustrate the pipeline. The framework is depicted in Figure 2.

Initialization and solution sampling

The Solution Base serves as an essential module to (1) enable storage of explored solutions and their evaluation results, (2) provide historical solutions to the LLM for few-shot learning, and (3) return the top-performing solution when iteration ends.

The initialization involves generating an initial solution that is randomly sampled from the entire solution space. Specific initialization strategies can be designed in future work, as initialization may influence subsequent reasoning. In the DSM sequencing task, a solution represents a complete and non-repetitive sequence of nodes (see Fig. 2). This initial solution is then evaluated and added to the Solution Base for

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

future use. In subsequent iterations, we design a sampling rule that selects K_p top-performance solutions and randomly samples K_q solutions from the remaining $K_n - K_p$ solutions to form a solution set, where K_n is the total number of solutions in the Solution Base. K_p and K_q are adjustable parameters. The obtained solution set is further refined and crafted into prompts.

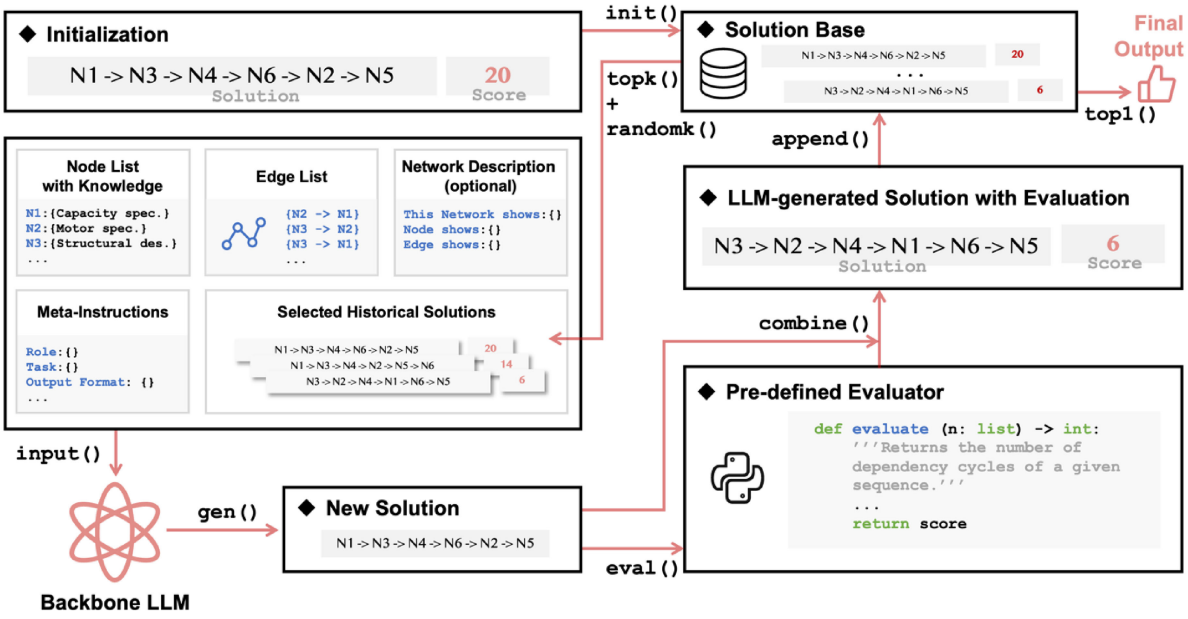


Figure 2. Overview of the proposed framework

LLM-driven optimization using network information and domain knowledge

In each iteration of the optimization process, we prompt backend LLM with the following information: ***(i) Typology Information:*** These two elements complete the mathematical description of a DSM. It is noteworthy that there are multiple equivalent representations for describing a network mathematically, such as an edge list, a dependency relationship list according to node sequence, or an adjacency matrix. In this research, we choose the edge list as the representation of the network's topology and shuffle all edges to avoid any possible bias. ***(ii) Contextual Domain Knowledge:*** This includes the name of each node and an overall description of the network, which conveys the domain knowledge underlying the DSM's mathematical structure to the LLM. For instance, in an activity DSM, each node represents the name of an activity in the entire design process. ***(iii) Meta-instructions:*** We adopt some frequently used prompt engineering strategies (Zhao, et al. 2023), including role-playing, task specification, and output format specification. These strategies allow the LLM to follow the guidance to perform reasoning and generate solutions in a specific format. ***(iv) Selected historical solutions:*** As described in the last section, we obtain a set of up to $K_p + K_q$ solutions through sampling from the Solution Base for the LLM to use in few-shot learning.

Once receiving inputs described above, the backend LLM combines the network topology information with domain knowledge to infer and suggest new solutions. The generated solution must pass the checker, which verifies that all nodes are present exactly once in the sequence. Once validated, the solution is then evaluated and appended to the Solution Base. The detailed input prompt is included in Appendix 1.

Evaluation of DSM sequencing solutions

The evaluator is used to quantify each newly generated solution. For the DSM sequencing task, the goal is to reorder the rows and columns of the DSM to minimize feedback loops. To achieve this objective, the evaluator calculates the number of backward dependencies in the corresponding sequence.

We formally describe this problem as follows: Given a sequence s, we can use the edge list of the network to obtain an n-by-n asymmetric adjacency matrix A, representing a directed network of n nodes, where a_{ij} is a typical binary entry. Specifically, $a_{ij} = 1$ represents that node i depends on node j. The task is to reorder the n correlated rows and columns to minimize the number of entries above the main diagonal in the corresponding adjacency matrix. On this basis, the objective function can be formally expressed as:

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

$\min_{S} \sum_{i=1}^{n} \sum_{j=i+1}^{n} a_{ij}$, where $i < j$.

For each sequence generated by the LLM (including the initial randomly created one), we calculate its evaluation score using the formula above. In this task, a lower score indicates a higher quality solution.

3. Experiments

3.1. Data

We collected four DSM cases for our experiments, which can be categorized into two types (Browning 2001): (1) Activity-based DSMs, which represent the input-output relationships between different tasks or activities within a project; and (2) Parameter-based DSMs, which illustrate the relationships among design parameters of a product. The DSM of the Unmanned Combat Aerial Vehicle (UCAV) includes 12 conceptual design activities conducted at Boeing (Browning 1998). The DSM of the Microfilm Cartridge was derived from Kodak's Cheetah project including 13 major tasks (Ulrich and Eppinger 2016). These two activity-based DSMs were constructed based on interviews with relevant engineers, followed by review, verification, and calibration to ensure accuracy. For the parameter-based DSMs, the Heat Exchanger (Amen, Rask and Sunnersjö 1999) and the Automobile Brake System (Black, Fine and Sachs 1990), researchers first identified the key components from the product and then interviewed the corresponding designers to define design parameters and establish precedence relationships. The Heat Exchanger DSM contains 17 components related to core thermal exchange elements, while the Brake System DSM includes 14 main parameters covering braking mechanisms and their dependencies. From the referential documents, we extracted: (1) the name of each node, (2) the edge list obtained from the adjacency matrix, and (3) the overall description of each network. All data were kept consistent with the original references. The characteristics of four DSMs are summarized in Table 1. In general, the node count (***N***) of DSMs ranges from 12 to 17, and the edge count (***E***) varies between 32 and 47. The network diameter, representing the longest shortest path between any two nodes, spans from 2 to 7. We also present measures such as network density and clustering coefficient to highlight the complexity. For instance, the UCAV DSM has a high network density of 0.712 and a clustering coefficient of 0.773, suggesting strong interconnections, while measures of the Heat Exchanger DSM indicate a sparser network with more distinct relationships.

Table 1. Characteristics of four DSMs

	N	E	Network Diameter	Network Density	Average Degree	Clustering Coefficient	Average Path Length
Activity-Based DSMs							
Unmanned Aerial Vehicle*(Browning 1998)*	12	47	2	0.712	7.833	0.773	1.288
Microfilm Cartridge *(Ulrich and Eppinger 2016)*	13	41	3	0.526	6.308	0.682	1.577
Parameter-Based DSMs							
Heat Exchanger *(Amen, et al. 1999)*	17	41	7	0.302	4.824	0.457	2.397
Automobile Brake System *(Black, et al. 1990)*	14	32	4	0.352	4.571	0.414	1.824

3.2. Experiment setup

Experimental setting

In all our experiments, we ran each method 10 times with different random seeds to ensure robustness. For solution sampling, we set K_p=5 and K_q=5. For the index of each node, we use a string composed of 5 randomly generated characters (a mix of numbers and letters) to represent the node uniquely. The maximum number of iterations was set to 20. We selected *Claude-3.5-Sonnet-20241022* as the backbone LLM and kept all other settings of the LLM as their default values (Anthropic 2024).

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

Comparative methods for benchmarking

We consider two types of approaches for benchmarking.

The first type is stochastic methods, which rely on probabilistic rules to explore the solution space. We chose the classic Genetic Algorithm (GA) for comparison, as it has been successfully applied to various optimization problems (Alhijawi and Awajan 2024). We implemented GA using the DEAP library (Fortin, et al. 2012). We considered three different settings for GA: *exploration-focused*, *exploitation-focused*, and *balanced setting*. Specific parameter settings of three variants are detailed in Appendix 2.

The second type is deterministic methods, which rank nodes based on a specific measure and then use this ranking for reordering. Since the calculation of each measure is deterministic, the resulting solution is consistent. We considered five deterministic methods for comparison, each reordering nodes based on different criterion: **(i) Out-In Degree** (Crofts and Higham 2011): Reordering based on the difference between out-degree and in-degree of each node. **(ii) Eigenvector** (Dietzenbacher 1992): Calculating the values of components in the Perron vector of the adjacency matrix and sorting nodes accordingly. **(iii) Walk-based (Exponential)** (Estrada and Rodriguez-Velazquez 2005): Considering in-depth connectivity patterns by involving the power of the adjacency matrix $\boldsymbol{A}$. It is calculated by: $F(\boldsymbol{A}) = \exp(\boldsymbol{A})$. **(iv) Walk-based (Resolvent)** (Estrada and Higham 2010): Calculated by: $F(\boldsymbol{A}) = (1 - \delta\boldsymbol{A})^{-1}$, where represents the probability that a message will successfully traverse an edge. We set $\delta = 0.025$ in experiments following the original reference. **(v) Visibility** (MacCormack, Baldwin and Rusnak 2012): Involving the visibility matrix, showing the dependencies between all system elements for all possible path lengths up to the matrix size. It is calculated by: $F(\boldsymbol{A}) = \sum_{k=0}^{n} \boldsymbol{A}^k$, where n represents the number of nodes. The resulting visibility matrix is then binarized. For methods **(iii)**, **(iv)**, and **(v)**, once the $F(\boldsymbol{A})$ is obtained, nodes are reordered by the sum of rows of $F(\boldsymbol{A})$. If nodes share the same value, they are then reordered based on the sum of columns of $F(\boldsymbol{A})$.

To further investigate the effectiveness of incorporating domain knowledge, we also consider a variant of our proposed method for comparison. In this variant, all settings are kept the same as in the main method, except that the contextual knowledge about nodes and the entire network is removed. Therein, only the topological information of the network is engineered into the input prompt, guiding the LLM to search for an optimal solution solely through mathematical reasoning.

4. Results and discussion

We first compared the convergence speed among our LLM-based methods and the three variants of GA, as shown in Figure 3. In GA, due to different parameter settings (e.g., population size, crossover, and mutation rates), each iteration explores multiple unique solutions. In contrast, our methods suggest only one unique solution per iteration. Therefore, we standardized the iteration time (generation) for GA as the number of unique solutions explored and used this as the measure for convergence speed. Although the maximum number of generations for each GA run was set to 2,000 (refer to Appendix 2), we truncated the comparison to the first 10,000 unique solutions explored for clearer visualization.

Compared to all three GA settings, our methods show significantly higher convergence speed, reaching lower objective values faster across all four cases. Overall, our LLM-based methods demonstrate a consistent advantage over GA, finding high-quality solutions multiple orders of magnitude faster in all cases. In each of the four cases, our methods can always identify an initial solution of good quality within the first attempt and continue optimizing the solution through subsequent iterations, approximating the optimal solution. This demonstrates that LLMs can leverage in-context learning to adapt and improve solutions efficiently.

Moreover, in Figures 3 (A), (C), and (D), we observe that incorporating domain knowledge enhances the LLM's performance in both convergence speed and solution quality, as indicated by the red line consistently lying below the grey line. This suggests that LLMs combining semantic reasoning with mathematical reasoning can solve CO problems more effectively and efficiently. It is noteworthy that in the cartridge development case (Figure 3 (B)), both LLM-based methods (with and without knowledge) found the best solution within the first step. Consequently, the two lines overlap, indicating identical performance in this case.

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

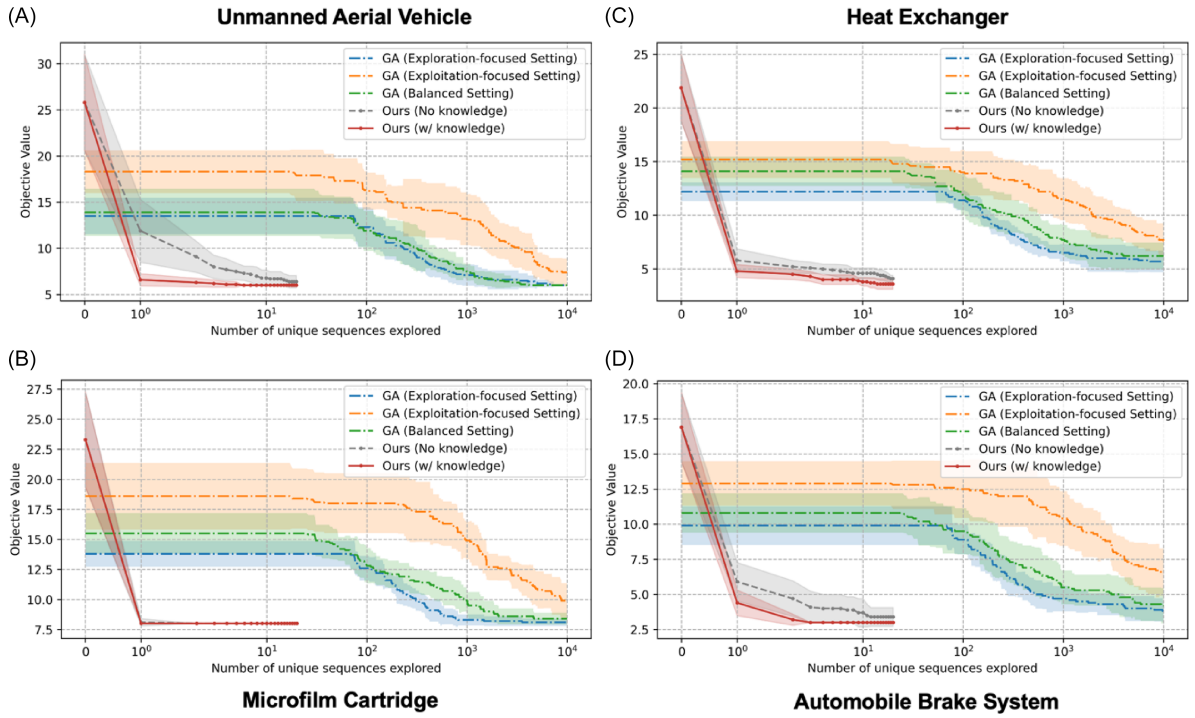


Figure 3. Comparison of convergence speed

We also compared the solution quality among different methods, as presented in Table 2. We experimented with our methods over single-trial, 5-trial, and 20-trial runs to evaluate their effectiveness. Compared to stochastic methods and deterministic methods, our methods significantly outperformed the corresponding benchmarks.

On the one hand, since all deterministic methods can be regarded as single-trial approaches (static algorithms), we first compared our single-trial results with those methods. As shown in Table 2, except for the Visibility-based method finding the optimal solution directly in one case (brake system design DSM), our methods outperformed all other deterministic methods across all cases. Even in the brake system case, our 5-trial method matched the effectiveness of the Visibility-based method.

On the other hand, when comparing with stochastic algorithms, our 20-trial method consistently found better solutions than benchmarks and achieved the optimal solution in 3 out of 4 cases. It is important to note that our experiments only tested up to 20 iterations. If we were to continue with longer iterations, the heat exchanger design case might also reach the optimal solution. Moreover, even with the 5-trial runs, our methods outperformed all three variants of GA in 3 out of 4 cases and were close in the UCAV case (6.1 ± 0.3 vs. 6.0 ± 0.0). This demonstrates that our LLM-based approach, leveraging contextual domain knowledge and iterative in-context learning, can effectively solve CO problems, suggesting high-quality solutions in a few attempts.

To evaluate the effect of different backbone LLMs on our proposed method, we conducted an ablation study. In addition to *Claude-3.5-Sonnet* (Anthropic 2024), we repeated the experiments in Table 2 using three different backbones: two open-source LLMs (*Mixtral-7x8B (Jiang, et al. 2024)* and *Llama3-70B* (Meta 2024)) and one closed-source LLM (*GPT-4-Turbo* (OpenAI 2024)), accessed via its API. We report the results for single-trial, 5-trial, and 20-trial runs, and also examine the impact of removing contextual domain knowledge from the prompts. As indicated by Table 3, *Claude-3.5-Sonnet*, our choice, statistically outperforms the other LLMs across all four cases. *Llama3-70B* follows as the second top performer in most cases, demonstrating that open-source LLMs can be effective alternatives. This suggests that our framework is compatible with different LLMs. We recommend users or researchers may start with *Claude-3.5-Sonnet* for optimal performance, but may consider open-source models like *Llama* for budget or customizability. Furthermore, in all cases, the results across all LLMs show that incorporating contextual domain knowledge consistently improves solution quality compared to the ones without knowledge. This finding aligns with our previous observation in Figure 1, highlighting the robustness of our methods and the significant role of domain knowledge in enhancing LLMs' reasoning performance in such CO tasks with real-world contexts.

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

Table 2. Comparison of solution quality.

Methods	Activity-Based DSMs Unmanned Aerial Vehicle	Microfilm Cartridge	Parameter-Based DSMs Heat Exchanger	Automobile Brake System
Stochastic Methods[1]				
GA (Exploration-focused setting)	**6.0±0.0**	8.1±0.3	5.7±1.0	3.8±0.7
GA (Exploitation-focused setting)	7.4±1.5	9.9±1.4	7.6±1.4	6.6±1.7
GA (Balanced setting)	**6.0±0.0**	8.4±0.5	6.2±1.2	4.2±1.2
Deterministic Methods[2]				
Out-In Degree	10.0±0.0	12.3±0.5	10.3±0.6	5.7±0.9
Eigenvector	15.0±0.0	13.8±0.7	13.1±1.7	11.1±1.4
Walk-based (Exponential)	15.0±0.0	12.0±0.0	8.0±0.0	11.0±0.0
Walk-based (Resolvent)	9.0±0.0	12.0±0.0	8.0±0.0	11.0±0.0
Visibility	25.6±2.2	8.8±0.7	6.0±1.3	**3.0±0.0**
LLM-driven Methods (Ours)				
Single-trial with knowledge	6.6±0.7	**8.0±0.0**	4.8±0.6	4.4±0.9
Single-trial without knowledge	11.9±3.4	8.1±0.3	5.8±1.1	5.9±1.4
5-trial with knowledge	6.1±0.3	**8.0±0.0**	4.0±0.4	**3.0±0.0**
5-trial without knowledge	7.5±0.9	**8.0±0.0**	4.9±0.5	4.0±1.0
20-trial with knowledge	**6.0±0.0**	**8.0±0.0**	**3.6±0.5**	**3.0±0.0**
20-trial without knowledge	6.4±0.7	**8.0±0.0**	4.1±0.3	3.4±0.7

5. Concluding remarks

Experimental results demonstrate that our proposed method, particularly when incorporating contextual domain knowledge, offer significant advantages in both convergence speed and solution quality compared to benchmarking stochastic and deterministic approaches. Across all experiments and backbone LLMs, the inclusion of domain knowledge consistently improved performance. While Claude-3.5-Sonnet is the most effective backbone LLM, open-source models such as Llama3-70B also shows strong performance. Overall, these findings illustrate the capability of our proposed method to effectively address DSM sequencing tasks, leveraging both semantic and mathematical reasoning of LLMs for real-world combinatorial optimization problems.

Despite the promising results, there are some limitations. First, the scope of the current experiments could be expanded by collecting more diverse DSM cases, including larger and more complex networks, to achieve more robust statistical results. Second, while we focused on DSM sequencing tasks, the effectiveness of our proposed method could be explored across a wider range of real-world combinatorial optimization tasks to assess its generalizability. Third, LLMs exhibit a black-box nature, making it challenging to fully understand their decision-making mechanisms. Therefore, we plan to delve into the intermediate changes in LLM outputs during optimization in future work, which may provide greater interpretability and insights into the LLMs' reasoning processes.

[1] Consistent with Figure 3, all three GA settings show optimization performance with number of unique sequences explored = 10,000.

[2] All deterministic methods can be considered single-trial approaches, as they produce consistent outcomes for identical inputs across repeated executions. However, certain nodes might share the same measures. For these nodes, a random order is applied, and the final results are derived over 10 runs.

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

Table 3. Ablation on the Backbone LLM

# of Trial	Backbone LLM	Knowledge	Activity-Based DSMs: Unmanned Aerial Vehicle	Activity-Based DSMs: Microfilm Cartridge	Parameter-Based DSMs: Heat Exchanger	Parameter-Based DSMs: Automobile Brake System
1	Mixtral-7x8B	with	12.5±2.2	19.5±5.0	16.1±2.2	12.1±3.1
		without	15.8±4.6	17.7±3.3	16.8±1.9	16.0±2.3
	Llama3-70B	with	7.6±0.8	8.3±0.5	7.9±2.0	6.1±1.1
		without	12.5±3.8	10.2±1.2	10.1±1.8	7.0±2.2
	GPT-4-Turbo	with	9.6±3.2	10.0±1.6	9.2±2.1	5.0±1.2
		without	13.9±4.3	11.0±1.5	12.9±3.1	7.5±2.7
	Claude-3.5-Sonnet	with	**6.6±0.7**	**8.0±0.0**	**4.8±0.6**	**4.4±0.9**
		without	11.9±3.4	8.1±0.3	5.8±1.1	5.9±1.4
5	Mixtral-7x8B	with	10.6±1.4	13.1±2.7	13.0±2.1	9.7±2.5
		without	12.3±2.6	15.0±2.2	15.4±2.3	13.8±1.7
	Llama3-70B	with	6.7±0.5	8.1±0.3	5.5±0.9	5.4±1.1
		without	8.4±1.7	9.3±0.5	8.7±1.2	5.4±1.4
	GPT-4-Turbo	with	6.8±0.6	8.4±0.5	6.6±1.3	4.6±0.9
		without	8.9±0.8	9.6±0.8	10.1±2.7	5.7±1.8
	Claude-3.5-Sonnet	with	**6.1±0.3**	**8.0±0.0**	**4.0±0.4**	**3.0±0.0**
		without	7.5±0.9	**8.0±0.0**	4.9±0.5	4.0±1.0
20	Mixtral-7x8B	with	10.1±1.2	11.2±2.3	10.7±1.2	8.8±2.0
		without	11.3±1.7	13.0±1.2	14.0±1.3	13.1±1.7
	Llama3-70B	with	6.7±0.5	**8.0±0.0**	4.7±0.5	5.2±1.2
		without	7.4±0.8	8.8±0.6	7.7±1.3	4.6±0.9
	GPT-4-Turbo	with	6.1±0.3	8.2±0.4	5.1±0.8	4.1±0.7
		without	7.3±0.6	9.1±0.3	7.7±1.8	5.0±1.5
	Claude-3.5-Sonnet	with	**6.0±0.0**	**8.0±0.0**	**3.6±0.5**	**3.0±0.0**
		without	6.4±0.7	**8.0±0.0**	4.1±0.3	3.4±0.7

In conclusion, our study highlights the potential of LLMs for combinatorial optimization of design structure matrix, showing that including domain knowledge can significantly enhance the performance of our methods. Future research can build on these findings to further refine LLM-driven methods and extend applications to more complex and diverse combinatorial optimization problems.

References

Alhijawi, Bushra, and Arafat Awajan. 2024. "Genetic algorithms: Theory, genetic operators, solutions, and applications." *Evolutionary Intelligence*, 1245–1256.

Amen, Rafael, Ingvar Rask, and Staffan Sunnersjö. 1999. "Matching Design Tasks to Knowledge-Based Software Tools: When Intuition Does Not Suffice." *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.* 1165–1174.

Anthropic. 2024. *Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku.* Accessed Oct 2024. https://www.anthropic.com/news/3-5-models-and-computer-use.

Black, Thomas Andrew, Charles Fine H., and Emanuel M. Sachs. 1990. "A method for systems design using precedence relationships: An application to automotive brake systems." *Working Paper of Sloan School of Management, Massachusetts Institute of Technology.*

Browning, Tyson R. 2001. "Applying the design structure matrix to system decomposition and integration problems: a review and new directions." *IEEE Transactions on Engineering management*, 292–306.

Browning, Tyson R. 1998. "Modeling and analyzing cost, schedule, and performance in complex system product development." *Ph D dissertation, Massachusetts Institute of Technology.*

Chang, Yupeng, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, . 2024. "Survey A. on Evaluation of Large Language Models." *ACM Transactions on Intelligent Systems and Technology*, 1–45.

Choi, Young Mi. 2011. "Effective scheduling of user input during the design process." *Proceedings of the 18th International Conference on Engineering Design (ICED 11).* 116–122.

Crofts, Jonathan J., and Desmond J. Higham. 2011. "Googling the brain: Discovering hierarchical and asymmetric network structures, with applications in neuroscience." *Internet Mathematics*, 233–254.

Dietzenbacher, Erik. 1992. "The measurement of interindustry linkages: Key sectors in the Netherlands." *Economic Modelling*, 419–437.

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

Eppinger, Steven D., and Tyson R. Browning. 2012. *Design structure matrix methods and applications.* MIT press.
Eppinger, Steven D., Daniel Whitney E., Robert Smith P., and David A. Gebala. 1994. "A model-based method for organizing tasks in product development." *Research in engineering design*, 1–13.
Estrada, Ernesto, and Desmond J. Higham. 2010. "Network properties revealed through matrix functions." *SIAM review*, 696–714.
Estrada, Ernesto, and Juan A. Rodriguez-Velazquez. 2005. "Subgraph centrality in complex networks." *Physical Review E-Statistical, Nonlinear, and Soft Matter Physics*, 056103.
Fortin, Felix-Antoine, Francois-Michel De Rainville, Marc-Andre Gardner, Marc Parizeau, and Christian Gagne. 2012. "DEAP: Evolutionary algorithms made easy." *Journal of Machine Learning Research*, 2171–2175.
Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, and Devendra Singh Chaplot. 2024. "Mixtral of experts." *arXiv preprint*, arXiv:2401.04088.
Jiang, Shuo, and Jianxi Luo. 2024. "AutoTRIZ: Artificial Ideation with TRIZ and Large Language Models." *International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE).* ASME.
Korte, Bernhard H., Jens Vygen, Korte B., and Vygen J.. 2011. *Combinatorial optimization. Springer.*
Liu, Shengcai, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024. "Large language models as evolutionary optimizers." *IEEE Congress on Evolutionary Computation. IEEE.* 1–8.
MacCormack, Alan, Carliss Baldwin, and John Rusnak. 2012. "Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis." *Research policy*, 1309–1324.
Makatura, Liane, Michael Foshey, Bohan Wang, Felix Hähnlein, Pingchuan Ma, Bolei Deng, Megan Tjandrasuwita, . 2024. "Large Language Models for Design and Manufacturing." *An MIT Exploration of Generative AI.*
Mei, Aoran, Guo-Niu Zhu, Huaxiang Zhang, and Zhongxue. Gan. 2024. "ReplanVLM: Replanning Robotic Tasks With Visual Language Models." *IEEE Robotics and Automation Letters*.
Meta. 2024. *Introducing Meta Llama 3: The most capable openly available LLM to date.* Apr. https://ai.meta.com/blog/meta-llama-3/.
Naseri, Gita, and Mattheos AG Koffas. 2020. "Application of combinatorial optimization strategies in synthetic biology." *Nature communications*, 2446.
Open AI. 2024. *GPT-4 Turbo.* Apr. https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo.
Pinedo, Michael L. 2012. *Scheduling*. New York: Springer.
Qian, Yanjun, Jun Lin, Thong Ngee Goh, and Min Xie. 2011. "A novel approach to DSM-based activity sequencing problem." *IEEE Transactions on Engineering Management*, 688–705.
Romera-Paredes, Bernardino, Mohammadamin Barekatain, Alexander Novikov, Matej Balog M. Pawan Kumar, Emilien Dupont, Francisco Ruiz JR, et al. 2024. "Mathematical discoveries from program search with large language models." *Nature*, 468–475.
Steward, Donald V. 1981. "The Design Structure System: Method A. for Managing the Design of Complex Systems." *IEEE Transactions on Engineering Management*, 71–74.
Ulrich, Karl T., and Steven D. Eppinger. 2016. *Product design and development.* McGraw-hill.
Wei, Jason, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, et al. 2022. "Emergent Abilities of Large Language Models." *Transactions on Machine Learning Research.*
Xidias, Elias, and Philip Azariadis. 2019. "Energy efficient motion design and task scheduling for an autonomous vehicle." *In Proceedings of the Design Society: International Conference on Engineering Design.* 2853–2862.
Yang, Chengrun, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc Le V., Denny Zhou, and Xinyun Chen. 2024. "Large Language Models as Optimizers." *The Twelfth International Conference on Learning Representations.*
Zhao, Wayne Xin, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, et al. 2023. "Survey A. of Large Language Models." *arXiv preprint*, arXiv:2303.18223.
Zhu, Qihao, and Jianxi Luo. 2023. "Generative transformers for design concept generation." *Journal of Computing and Information Science in Engineering*, 041003.

Appendix 1. Full prompt

Prompt for our proposed method:

You are an expert in the domain of combinational optimization.
Please assist me to find an optimal sequential order that minimizes feedback cycles in the dependency network described below. Your task is to propose a new order that differs from previous attempts and has fewer feedback cycles than any listed.

<Description of the entire Network< {network_description} </Description of the entire Network>
<Nodes with Descriptions> {node_list_with_description} </Nodes with Descriptions>
<Edges> {edge_list} </Edges>

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press

Below are some previous sequential orders arranged in descending order of feedback cycles (lower is better): {selected_historical_solutions}
Please suggest a new order that:

- Is different from all prior orders.
- Has fewer feedback cycles than any previous order.
- Covers all nodes exactly once.
- Starts with <order> and ends with </order>.
- You can use the descriptions of nodes and networks to support your suggestion.

Output Format:
<order> ... </order>

Please provide only the order and nothing else.

Appendix 2. Parameter settings for three variants of GA

In all three variants of the GA used for DSM sequencing tasks, we employed the following shared settings: the number of generations was set to 2,000, with a selection mechanism using tournament selection and mutation using shuffled indexes. The GA was implemented using the DEAP library with standard configurations for initial population generation. The different configurations for each GA setting are shown in the table below. The meaning of each parameter refers to (Fortin, et al. 2012).

	Population	Individual Mutation Prob.	Tournament Size	Crossover Prob.	Mutation Prob.
Exploration-focused	50	0.05	5	0.6	0.4
Exploitation-focused	10	0.01	20	0.9	0.1
Balanced	20	0.02	10	0.7	0.3

https://doi.org/10.1017/pds.2025.10234 Published online by Cambridge University Press