www.editada.org

*International Journal of Combinatorial Optimization Problems and Informatics, 16(3), May-Aug 2025, 334-344. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2025.v16i3.746*

# A Current Revision of Prompt Engineering in Business Operations

Juan C. Aldana-Bernal[1,2] and Ana X. Halabi-Echeverry[2*]
[1] National Colombian University, Bogotá D.C
[2] NextPort vCoE INC
jcaldanab@unal.edu.co; axhalabi@nextport.org
*Corresponding Author: axhalabi@nextport.org

**Abstract.** Prompt engineering in the context of operations emerges as a key discipline to optimise artificial intelligence (AI) models for specific operational tasks. Considering the importance of this field, our group offers a current review of designed prompts using language models in business operations. At this stage, the study focuses on confirming the advancement of prompts in response to their precise formulation and applicability in real-world scenarios and different engineering approaches. We use the gold mining problem to evaluate prompt techniques such as Few-shot, Chain-of-thought, and Tree-of-thoughts (ToT) in LLMs. The results show the importance of adapting the prompts to the type of technique and the characteristics of the problem at hand. Our research also offers theoretical and practical foundations for their integration with AI models, highlighting the importance of prompt engineering to enhance automation and decision-making in business environments.

**Keywords:** Artificial Intelligence Models, Prompts, Operations.

Article Info
Received January 28, 2025
Accepted March 17, 2025

## 1. Introduction

Large Language Models (LLMs) advance artificial intelligence (AI) in several accessible ways. The widespread adoption of AI has led to their popularization, allowing individuals and organisations alike to utilize them in daily activities. However, LLMs' capabilities are not fully exploited. LLMs are trained on massive datasets and their responses are generated based on an average of the available data. LLMs' may exhibit inaccuracies when querying elements in specific situations and conditions due to the need for users to provide a context in which they can offer a valid response (ChatGPT 4, OpenAI & Sabit, E., 2024). Prompt Engineering has emerged to receive the instructions sorted by LLMs generating consistent responses to user requirements and its design helps to maximize the benefits of LLMs through precise instructions. Organisations may turn to substantial improvements in their operations by utilizing well-designed prompts. Furthermore, Prompt engineering can streamline, enhance, and reduce the costs of many processes currently in place.

Our research uses the gold-hunting problem to offer an application in an operational domain such as routing allocation, and the evaluation for optimal alternatives in a project. We seek to identify how Prompt Engineering also known as prompting can aid in properly solving this problem. The document briefly reviews how prompting has been achieved, then defines the main elements of prompts, followed by a description of the fundamental components of the prompts and an identification of the particularities of complex prompts. Finally, the case study and the results generated using different LLMs are presented.

## 2. Previous developments

Alan Turing's 1950 proposal stipulated that a key requirement for AI was to satisfy the imitation game. The term *artificial intelligence* was later introduced by John McCarthy, who solidified the idea that machines could think. A notable advancement in AI occurred with Apple's release of Siri in 2011, an application for mobile devices that uses natural language processing to respond to basic user inquiries. A milestone in how AI operates was achieved by Google's AI team in their 2017 work: *Attention is All You Need*, introduced the transformer architecture. This model possesses memory capabilities and processes sequences of tasks in parallel (Abeliuk & Gutiérrez, 2021). Building upon these advancements, artificial neural networks (deep learning, DL)

and natural language processing (NLP) further evolved. In 2022, OpenAI launched ChatGPT-3.5 (Generative Pre-trained Transformer), considered the true birth of AI due to its ability to surpass the Turing test. Since then, various large language models (LLMs) have been developed. These models are characterized by their high capacity to generate language, resembling human speech, and predicting subsequent words based on prior sequences (Bommasani et al., 2021; McCoy, Yao, Friedman, Hardy, & Griffiths, 2023). LLMs have facilitated numerous applications across diverse sectors. Generative language models fall under the broader category of generative AI, which encompasses machine learning algorithms capable of learning from diverse input types such as text, images, and audio (natural language) and generate new content in formats including text, images, video, and audio. Users engage with these models by inputting prompts, which enables LLMs to provide consistent responses based on knowledge acquired in its training process (Cao, Li, Liu, Yan, Dai, Yu, et al., 2023).

## 3. Prompts, challenges and characteristics

Prompt engineering or prompting is a form of interaction between humans and AI (Oppenlaender, 2022). It consists of a set of techniques and methods for designing, writing, and optimizing the set of instructions or prompts. Prompt engineering involves a bidirectional interaction between humans and AI, where in each iteration, the LLM's results can be refined (Knoth et al., 2023). The main benefit obtained from prompting is to achieve optimized responses with the least user iteration effort (IBM, 2024). Prompting research has significant opportunities for education, health, science, engineering, and business (Busch, Rochlitzer, Sola, and Leopold, 2023; Velásquez-Henao et al, 2023). Knoth et al., (2023) consider its importance on educating new generations on the necessary skills and meaningful use of AI.

Prompts are a set of instructions provided by LLMs inputs, they are customized, specified, refined, and enhanced by the responses the LLMs provide (Liu, Yuan, Fu, Jiang, Hayashi, and Neubig, 2023). However, creating effective prompts is not a simple task, especially for non-technical individuals or a complex task that needs to be resolved. Prompts rely on trial-and-error techniques (Dang, Benharrak, Lehmann, & Buschek, 2022; Lo, 2023). While the responses provided by the LLMs can be imprecise, vague, incorrect, or inappropriate for the required context (Jha, Jha, Lincoln, Bastian, Velasquez, and Neema, 2023), effective prompts can identify and meet principles of clarity and precision, contextual information, and control of verbosity (Lo, 2023). Designing effective prompts is a pressing need (Giray, 2023; White, Fu, Hays, Sandborn, Olea, Gilbert, et al., 2023). The vast majority of prompts created in recent years satisfy repetitive tasks, some examples are workflow templates, marketing, advertising, and translations (Velásquez-Henao, Franco-Cardona & Cadavid-Higuita, 2023). In conclusion, prompts feed LLMs to generate precise, reliable, replicable, and objectively correct responses (Eager, & Brunton, 2023; Giray, 2023; White, et al., 2023).

A high-quality prompt provides the structure for the dialogue with the LLM on what information is important to the user, and the form and content expected in the response (White, et al., 2023). The design of LLMs also constrain the user's desirable output. However, understanding some of the elements that are embedded in LLMs is important to develop more effective prompts, among which the following stand out:

**a. Semantics:** semantics homogenized the terminology and concepts prompt need, using classifications and areas of knowledge (Velásquez-Henao et al., 2023). LLMs are trained with a vast amount of textual data and use neural networks to identify complex and linguistic patterns. LLMs understand terms and expressions in different languages, relationships of expressions, synonyms, and antonyms. Prompts use closed or open responses and thus can be divided into four categories. The first category presents simple questions, the second provides additional context about the writer and the language model. The third category enriches the request with examples of the LLM, and the fourth divides the request into individual components resembling a sequence of steps that improve the response's accuracy (Heston, and Khun, 2023). However, creating an efficient and generalizable prompt can be difficult as it requires different modeling strategies and trial-and-error process (Oppenlaender et al., 2022).

**b. Context:** refers to the theoretical and logical framework in which the request is grounded. It should include a definition of the social, organizational, or personal environment, also the knowledge domain for the LLM to generate a response, the task scope and parameters to be solved, the target audience, the level of specialization in the response and the expected language (Eager, et al., 2023). A comprehensive context improves the LLM results (Wu, Terry, & Cai, 2021).

**c. Training and Feeding:** LLMs are trained on massive amounts of unlabeled data, allowing the model to learn representations, semantics, and syntax, with hundreds of millions of parameters. This enables them to generate text with a high level of relevance and coherence. Although generalization is sought, the LLMs are limited by geographic,

cultural, and temporal availability or a few years of human knowledge (OpenAI, 2024). Fine-tuning occurs when the user defines specific requirements for the model to perform a particular task. In this sense, it has been shown that it is advisable, especially for medium and high-complex models, to refine the LLM with specific data and samples in the context of the requirement (Shieh, 2023).

**d. Memory:** The transformer architecture and the self-attention mechanism allow prompts to work with natural language processing (NLP) through parallel processing of text sequences. Moreover, the self-attention mechanism enables them to assign different levels of importance to several words in the sequence of interaction with the user. Through the encoding mechanism, the user's requirements are captured, and through decoding, the output is generated. The mechanism keeps memory of the different points of interaction, so it can go back and adjust previous responses (OpenAI, 2023). This feature makes the model intelligent because of the memory and learning from the past. At times of interaction, the mechanism may overlook the initial requirements, and therefore, human intervention may remind the prompt of its purpose and context.

**e. Positivity:** The LLMs are designed to avoid information that could negatively impact humanity. Their structure is designed to generate a positive response to requests, even if it is not the most concrete, appropriate, or logical result, resulting in hallucinations, that is a lack of common sense and understanding of reality (Floridi and Chiriatti, 2020; Ji et al., 2023). For this reason, when designing prompts, it is important to provide the system with relevant feedback, rephrase the requirement with different expressions, or define a better context. When necessary for a less positive evaluation in a text or report, it is advisable to use the word "critical" because it will understand that it is required to evaluate positive or negative aspects of a particular task.

In having greater clarity on how LLMs have been constructed, some authors suggest the structure that prompts should have, especially if working on tasks that are not basic and correspond to complex processes (Eager et al., 2023; Velásquez et al., 2023). Among the main elements of a complex prompt are:

- **Gold**: Indicates the objective the user wants to achieve, usually corresponds to a verb.
- **Output**: Defines the product, process, or action to be generated with specific requirements.
- **Context**: Explains the framework, scope, and parameters in which a request is made.
- **Characteristics and conditions**: Provides the attributes and conditions for generating an output. This element specifies the scope and clarifies the output, allows for the role of the prompt and the inquirer, as well as the desired response format.
- **Alignment**: Matches the LLM's output with the user's intent.
- **Constraints and limitations**: These are identified limitations for processing the response that the LLM considers.
- **Examples**: Adding examples, samples, or demonstrations essentially helps to identify and process the information, format of the output, and suitable responses—especially important for complex models.
- **Evaluate the response**: This step involves checking whether the criteria are met, including output accuracy, relevance, restrictions, appropriateness, or if hallucinations occurred. The CLEAR framework (Lo, 2023) can be used here to ensure the response is concise, logical, explicit, adaptive, and reflective. If the response does not meet the criteria, the process continues to an iteration step.
- **Iterate**: This step involves developing strategies for interacting with the LLM until getting an appropriate response and minimizing hallucinations (ChatGPT 4, 2024). Various strategies are suggested, such as a)rephrasing questions with different words, b)challenging answers by presenting counterarguments, c)providing additional information and sources, supplying examples, requesting arguments for the answers, asking for bullet points and staged responses, d)requesting data sources, e)asking for critical perspectives to foster alternative views, f)asking the LLM to reformulate, split questions, or breaking down a question into steps for complex models (Henrickson & Meroño Peñuela, 2023). Additionally, domain-specific models must be trained with sector or knowledge-specific data and present valuable responses.

## 4. Advanced Prompting Techniques

According to what is presented, prompts can respond to queries ranging from simple to highly technical solutions. They are grounded in iterative refinement, combining various prompt engineering techniques to train effectively on diverse input data, learning to adapt and mitigate biases and confusion, and generating more accurate results (IBM, 2024). Based on their complexity, these techniques are presented:

**a. Zero-shot prompting** This technique involves providing the prompt with no specific training task. The expected response is solely based on training and in a general context. The technique is suitable for general knowledge responses, text generation on various topics, translations, basic logical reasoning, and simple mathematical operations. An application is for simple general knowledge questions, where there is no possibility of ambiguity.
**b. Few-shot prompting** This technique allows the prompt to provide precise contextual information on a few examples and demonstrations to help the LLM understand what the user requires and how a response can be generated. It is important in this technique to provide at least some labeled data. For example, to classify animals, objects, or sentiments, a few samples can be used for satisfactory results. This technique has proven to be more effective for accurate responses than zero-shot (Pornprasit, C, & Tantithamthavorn, C, 2024).

**c. Chain of thought prompting (CoT)** Considering an advanced prompting technique offered by Wei et al. (2022), to provide a model with a step-by-step to generate a chain of logical thought, enables a better understanding of the task and generates the expected results. It can be used when the model needs to perform complex tasks such as of logical, arithmetic, and symbolic reasoning. A significant difference can be made by adding a phrase like "think step by step" to the prompt, allowing the LLM to break down the response into logical steps. It is useful when sequential steps are expected, such as in a production line process.

**d. Self-consistency** This complex technique involves presenting few examples and ways of thinking, to allow the LLM to identify alternatives and the one that generates a consistent response (Want, et al, 2022). This technique has been shown to significantly improve the accuracy of tasks involving arithmetic reasoning, common sense, and linguistics (Wang, et al, 2023). A typical case for this problem is retail with multiple items and inventory inflows and outflows due to purchases or sales. "Expert" alternatives are suggested, and the option with the highest score is selected. It is used when it is necessary to evaluate alternatives in a marketing plan with different strategies.

**e. Tree of Thought (ToT)** is considered a complex technique being proposed by Yao et al., (2023) and Long (2023), who, through activities and games demonstrate how tree search can improve the ability to solve complex problems. It is based on CoT and works with decision trees where thoughts correspond to language sequences that are partial steps toward solving a problem. In this situation, the prompt self-assesses its progress through the intermediate steps performed with an explicit reasoning process. This ability advances the intermediate thought steps supported by search algorithms such as breadth-first search (BFS) or depth-first search (DFS) and combines a systematic exploration in forward and backward search. This technique is favored for solving the gold-hunting problem and other alternatives of complex models that usually use graphs in their representation, such as the traveling salesman, or the CPM/PERT in a project.

Figure **1** shows the different techniques and their problem-solving capabilities. Section (a) can be associated with the zero-shot and few-shot techniques, where no context is provided or context and demonstrations are provided only at the beginning, but there are no intermediate steps. For case (b) CoT, additional intermediate steps are identified, which are sequential. In case (c) Self-consistency shows to be more robust than CoT alternatives. It has the disadvantage of being sequential and it is not easy to go back to ensure the desired outcome. Finally, (d) ToT combines non-sequential trees, with evaluation steps that could generate different alternatives at each step or even go back to identify better alternatives.

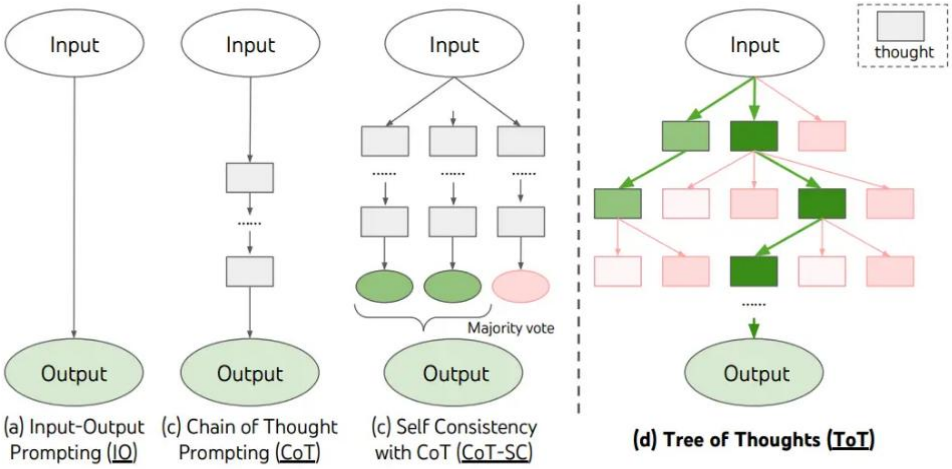


**Figure 1.** Solution Techniques with LLMs. Source: Yao et al., 2024**.**

### 4.1. The Gold-hunting problem

The selected model was the gold-hunting problem, which presents a fortune hunter joining the gold rush in 19th-century California in the American Wild West. This fortune hunter wanted to transport his gold through stagecoaches across this dangerous territory to the East, seeking the lowest cost route, determined by the lowest cost policy for each segment, as shown in Figure 2. The objective function is to minimize the travel cost between Node A in stage 1 and Node J in stage 5. The Gold-hunting problem is illustrated in Figure 2. It is structured into four different stages, each representing a sequential decision point. The objective is to optimize the overall cost by making the most favorable choice at every stage. Traditionally, dynamic programming problems can be solved by working either forward or backward through the stages. However, considering the iterative nature of the objective function, a backward induction approach proves to be more efficient. This method necessitates a dynamic programming table to store intermediate optimal solutions. The algorithm recursively determines the optimal path by iteratively updating the table, ensuring that the accumulated cost of the chosen path is minimized.

### 4.2. Iteration with LLMs

Despite assuming that the three selected LLMs, this is ChatGPT, Gemini, and Copilot (Microsoft) have different structures, databases, and training, we observe that Gemini may display more up-to-date information due to its integration with Google search. Secondly, ChatGPT focuses on capturing and summarizing text (IBM, 2024), with similar requirements that can better be defined to a particular LLMs. While identifying the prompting technique to the best support the of the solution of the gold-hunting problem, the Tree of Thought, techniques such as Few-shot prompting and CoT were tested to get the type of responses the LLMs can generate in each case.

In general, approximately four iterations were made with the LLMs before obtaining a satisfactory and complete response.

- The first iteration is identified as data loading. In this step, the prompt is defined with all relevant information about the problem, the objective, the type of problem, and the basic characteristics and constraints of the context. Since the graph is identified as an essential element, it is uploaded as an attachment. This model is tested on the three LLMs under the premise that it is a Few-shot technique, and knowing the problem's characteristics, a satisfactory response was not expected.
- The second iteration is identified as adjusting the input data and the first approximation to a response. Given the difficulties the LLMs had in interpreting the input data, the data feeding format and the way they were attached were changed, and their understanding was verified. In this case, the first approximate response to the request was obtained.
- The third iteration is identified as context specification. Given the responses obtained, the prompt is strengthened with examples and the premise of step-by-step development from back to front. Examples with breadth-first search (BFS) and depth-first search (DFS) for binary trees were specified, showing the desired

thought process. Here, CoT and ToT techniques were used. Satisfactory but incomplete responses were obtained in this phase.

- The fourth iteration is identified as obtaining expected results. Given the difficulties found in the third iteration models, questions were generated for the LLMs to analyse their outputs more deeply, generate introspection, and find potential optimization alternatives.

$$F_n^*(S_n, X_n) = C_s X_n + f_{n+1}^*(X_n) \quad (1)$$

- $X_n$: is the immediate destination in stage n.
- $S_n$: is the state variable of resources allocated up to stage n.
- $C_s X_n$: is immediate cost of stage n
- $f_{n+1}^*(X_n)$: is minimum future cost (stages n+1 onwards)

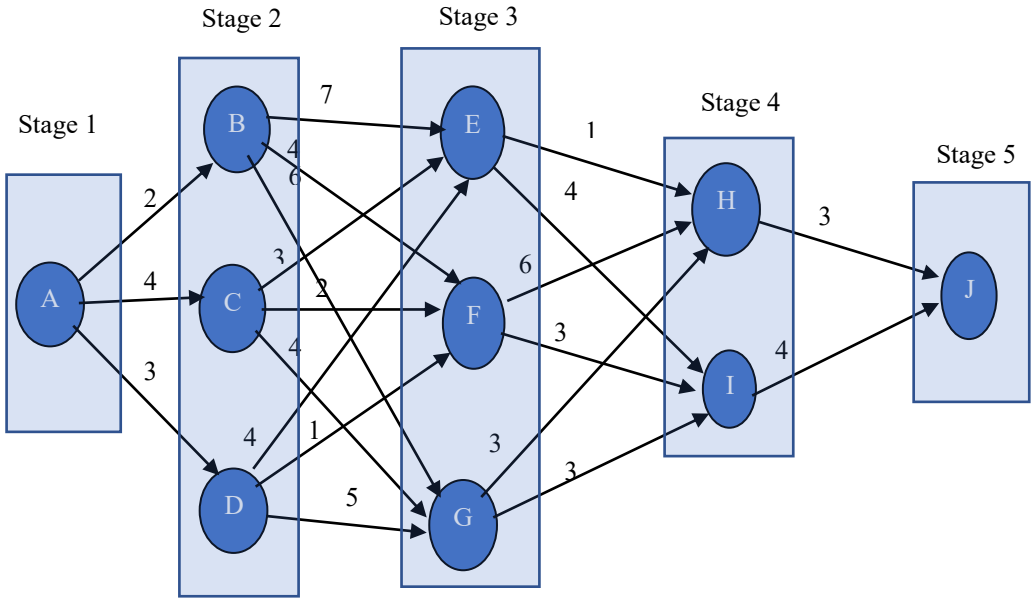


**Figure 2.** The Gold-hunting Problem - Transition Diagram

## 5. Results

### 5.1. First Iteration: Data Loading

All three LLMs were provided with the same file containing the graph of the model and a prompt with very similar characteristics, as shown in Figure 3. A Zero-shot prompting technique is used to identify answers they could provide. In this first iteration, the LLMs exhibited two main issues in their responses. They either could not read the graph properly or misinterpreted it as they failed to read it completely or accurately. This suggests that the initial step in a prompt should include verifying that files are loaded correctly, fully understood, and read in their entirety. It is also essential to ask the LLM about the types of files compatible with and assess that the concepts and relationships in the file align consistently with the provided content.

The second key aspect of the response is its positive approach, where the LLM acknowledges the incomplete information but still attempts to provide a response based on its partial understanding. These responses focused on specifying how the model's characteristics should be presented, including the types of files and prompt details required. Secondly, the response aimed to outline a methodology and a process to follow for addressing the solution (see Figure 4).

The graph in the attached file presents a network with the routes and costs for transportation that needs to go from node A in stage 1 to node J in stage 5. It must pass through all stages but only one node in each of them, with the purpose of identifying the minimum cost route. The cost matrix shows the costs and feasible paths from one node to another. Stage 1 only has node A, stage 2 has nodes B, C, and D. Stage 3 has nodes E, F, and G, and stage 4 has nodes H and I, and finally, stage 5 only has node J. What is the least cost route from A to J

**Figure 3**. First Iteration Prompt: Data Loading

Proposed Solution: Dijkstra's Algorithm

Dijkstra's algorithm is ideal for this type of problem as it allows finding the shortest path (in this case, the lowest total insurance policy cost) between two nodes in a weighted graph.

Steps to follow:

Initialization:

1. Assign each node a provisional value of infinity (except for node A, which starts with 0).
2. Create a set S to store visited nodes.

Node Selection:

3. Select the node with the smallest provisional value that is not yet in S. In this case, start with node A.

Neighbor Exploration:

4. Evaluate the neighboring nodes of the selected node and update their provisional values if a shorter route is found.

Add to Set S:

5. Add the selected node to set S.

Repetition:

6. Repeat steps 3, 4, and 5 until all nodes have been visited or until the destination node (J) has been added to S.

**Figure 4.** First Iteration Data Load Response

### 6.2. Second Iteration: Format Adjustment

At this stage, the graph format was changed to a table containing a cost matrix, with no modifications to the prompt. Based on the suggestions received from the LLMs, the format was switched to a cost matrix, which was then directly pasted into the instructions panel, or the matrix data, as described in this panel. The LLM's ability to load and understand the relationships in the file was evaluated. In another case, the matrix was not read correctly, but follow-up questions resolved the issue. Since the first response provides a solution sequence, the aim is to promote the Few-shot prompting technique.

The responses from this new iteration were more comprehensive but generally presented in a well-defined and adequate theoretical conceptualization. Although it explains that it is evaluating the response, it does not present it (see Figure 5). Only one of the LLMs provided a quantitative response in this iteration, but it was incorrect. This reflects that generally, the LLMs have complete theoretical information from different fields of knowledge, but for complex problems like the one mentioned, they need examples and demonstrations to generate a quantitative response. Hence, CoT and ToT models are necessary here.

### 6.3. Third Iteration, Context Specification

Once the deficiencies in the LLMs' responses were identified, the prompt was strengthened in three aspects. First, it sought to explain that this model is stage-based and follows a step-by-step process if two of those had been already identified. Second, it suggested performing a backward evaluation as it is an iterative model. Third, providing an example of a BFS ToT (see Figure 6) and for one of the cases a DFS would be useful.

To solve this problem step by step:
Stage 1: Node A. Stage 2: Nodes B, C, and D. Stage 3: Nodes E, F, and G. Stage 4: Nodes H and I. Stage 5: Node J.
Process:

1. We start from node A in stage 1 with an accumulated cost of 0.
2. Analyze the minimum cost of moving to each node in stage 2.
3. Continue until reaching the final stage at node J, accumulating the minimum cost at each step and selecting the least cost route.

**Figure 5**. Second Iteration Response, Format Adjustment

*Yes, but how about if we perform the analysis step by step? For example, we could do it in parts. If we start from the end and move backwards, since it is an iterative function, we find the optimal route from each node to J. For instance, if we are at node E to reach J, we have two paths: the route through H, which costs E-H = 1 and H-J = 3, so the route cost is 4. The other is the route through I, which costs E-I = 4 and I-J = 4, so the route cost is 8. Therefore, the optimal cost for node E* = 4 because it is the least cost route. We could then propose a solution from A to J again.*

**Figure 6.** Iteration Three, Context Specification

The responses obtained in this iteration were very interesting because they approached the actual solution, presenting the step-by-step process followed and the analyses carried out. However, although they concluded that there are two routes with the same minimum cost, they only identified one as optimal, which raises two concerns.

The first refers to whether it should be specified that the model can have more than one answer despite being an optimisation problem, and the second concerns how the LLMs are processing to identify one route (see Figure 7). It is also likely that since a step-by-step methodology is defined, the LLM assumes it is a CoT technique rather than a ToT.

Furthermore, the Gemini language presents a much more robust theoretical structure, but not a concrete answer. To address this, the prompt had to be strengthened with a specification of a DFS tree. Despite this, the LLM has two outstanding characteristics. The first is that it autonomously generates a self-consistency technique, which allows the user to observe three alternatives of the responses to each request. The second is that due to its very extensive databases (IBM, 2024), it is theoretically quite powerful and usually generates a solution with Python code that the user can apply to their problems to obtain a general solution to similar problems.

### 6.4. Iteration four, generation of final solutions

To generate the expected responses from the responses in the previous iteration, the technique of re-questioning and retrospection was used to go back in the analyses. In these specific cases, it was asked why the other optimal solution reflected in the previous solution was not proposed as an alternative. Finally, it was also requested retrospectively to evaluate another ToT solution to determine if it was optimal. This technique allows us to retrace steps and analyse the alternatives again. The two adjustments to the prompts allowed the LLMs to evaluate effectively the requests. It is found that there are other optimal solutions to the problem, which are finally presented, and is noticed that there are three alternatives that provide optimal solutions to the problem posed. See Figure 8.

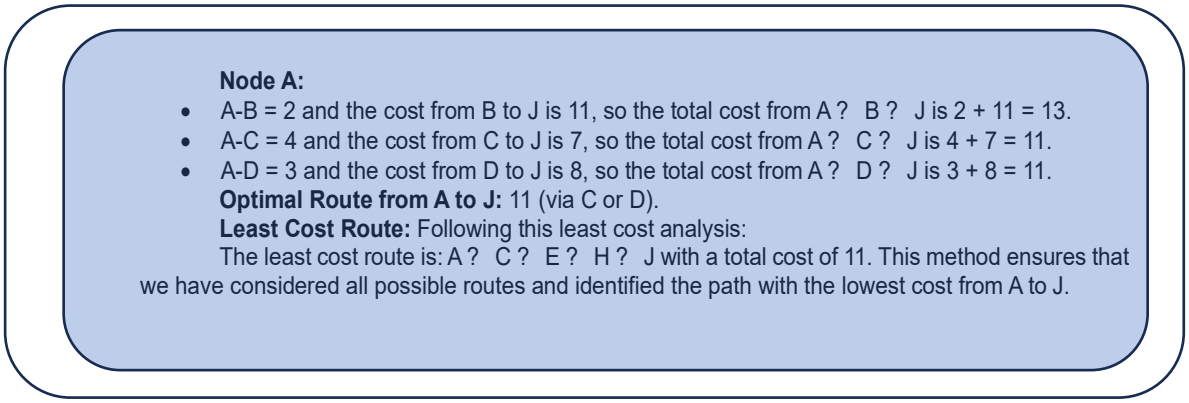


**Figure 7.** Response to Iteration 3, Context Specification

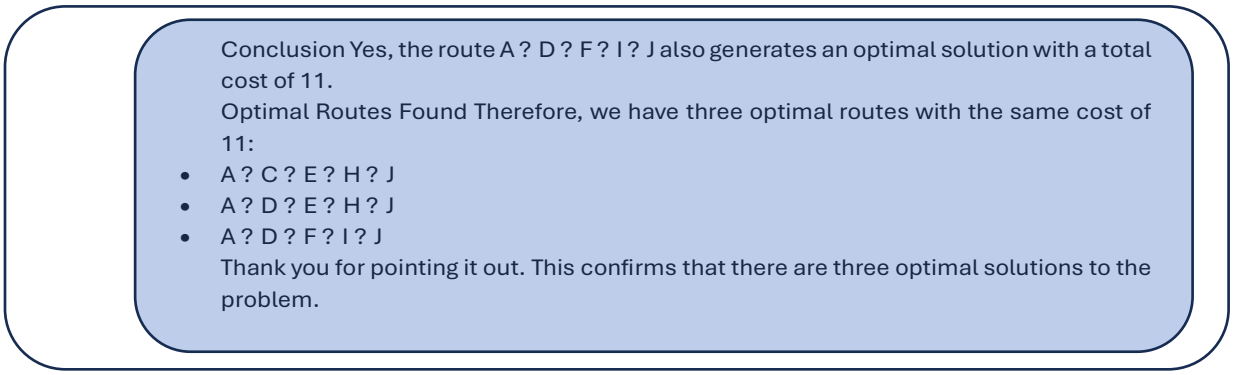


**Figure 8.** Output Iteration 4. Generation of final solutions

## 6. Conclusions and Recommendations

### 7.1. Conclusions

Prompts play a crucial role in enhancing LLMs as productivity tools and in creating effective solutions to organizational challenges. However, poorly designed prompts can lead to hallucinations and frustrations in addressing specific problems within organisations.

A thorough understanding of complex prompt techniques is essential for prompt engineers aiming to solve challenges efficiently, this also enables them to identify problem types suited to each technique. Each technique has unique requirements and a specific terminology that must be correctly applied for accurate interpretation.

A preliminary step when addressing complex problems, in particular, involves verifying the formats that LLMs load and the semantics used to avoid potential misunderstandings in data interpretation. Additionally, it is necessary to understand how these models are configured, including their semantic features and databases, as these factors can lead to differing responses to similar requests—a finding that is consistent with the work of Wang, Chen, Deng, X., et al. (2024). Therefore, certain LLMs may be better suited for text generation, others for quantitative analysis, or translation tasks, for example.

A key component in any complex prompt is the inclusion of examples or demonstrations, as these allow the LLM to learn or tailor the expected type of solution. Properly specifying a prompt can significantly improve the precision and clarity of the generated response, reducing costs in terms of time and machine processing. Iterations are important for interacting with LLMs and understanding the type of processing they perform on the information to produce accurate, reliable, and contextually appropriate answers.

The development of organizational operations usually involves the development of processes and decisions that are not obvious but rather complex. Thus, for example, the development of a production plan or a project plan has several complexities that, to be supported through an LLM, require detailed specifications and identification of task sequences that must be customisable. It other cases, where techniques such as CoT, Self-consistency, or ToT, are applied in prompting, those generate substantial increases in the productivity of organisations. More efficient and less costly processes better satisfy the customer, some examples are models such as the shortest route, scheduling, critical route or traveling salesman, that can usually be represented through graphs of high complexity.

Our study seeks to help in the search for solutions to complex problems in operations, which generate a high demand of resources, and need to continue improving substantially the increase for business productivity. Moreover, by designing prompts that are complete and suited to the required technique, significant efficiencies in development and timely results can be achieved. It is crucial to emphasize that any AI-generated output must be evaluated by human experts before decision-making, as human guidance tends to meet various criteria, including pedagogical ones, more consistently than LLMs alone (Bradford, Li, Gerard, & Linn, 2024).

### 7.2. Recommendations and Future Research

Further research is needed to identify the parameters, data, and characteristics of different LLMs, which would help determine which models are best suited to specific types of problems. This research would also enhance reliability concerning the types of files that LLMs can handle and their ability to generate results that meet the specified requirements.

As LLMs operate with diverse languages and semantic structures, future studies should aim to standardize the semantics used by these models, reducing the need for extensive verifications and training on the information they are expected to manage. Additionally, when working with complex techniques, it is essential to standardize language to establish reserved terms used exclusively for specific processes.

## References

Abeliuk, A., & Gutiérrez, C. (2021). Historia y evolución de la inteligencia artificial. *Revista Bits de Ciencia*, (21), 14–21.

Bradford, A., Li, W., Gerard, L., & Linn, M. (2024). Comparing expert and ChatGPT-authored guidance prompts. *UC Berkeley*. https://doi.org/10.1145/3657604.3664669

Busch, K., Rochlitzer, A., Sola, D., & Leopold, H. (2023). Just tell me: Prompt engineering in business process management. In *International Conference on Business Process Modeling, Development and Support* (pp. 3–11). Springer.

Cao, Y., Li, S., Liu, Y., Yan, Z., Dai, Y., Yu, P. S., et al. (2023). A comprehensive survey of AI-generated content (AIGC): A history of generative AI from GAN to ChatGPT. *arXiv preprint*. https://arxiv.org/pdf/2303.04226

ChatGPT-4, OpenAI, & Sabit, E. (2024). *Prompt engineering for ChatGPT: A quick guide to techniques, tips, and best practices. Learn from the best: Let Genie (ChatGPT) teach you how to make wise wishes (prompts)*.

Eager, B., & Brunton, R. (2023). Prompting higher education towards AI-augmented teaching and learning practice. *Journal of University Teaching and Learning Practice*, *20*(5). https://doi.org/10.53761/1.20.5.02

Floridi, L., & Chiriatti, M. (2020). GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, *30*(4), 681–694. https://doi.org/10.1007/s11023-020-09548-1

Giray, L. (2023). Prompt engineering with ChatGPT: A guide for academic writers. *Annals of Biomedical Engineering*. https://doi.org/10.1007/s10439-023-03272-4

Henrickson, L., & Meroño Peñuela, A. (2023). Prompting meaning: A hermeneutic approach to optimising prompt engineering with ChatGPT. *AI & Society*. https://doi.org/10.1007/s00146-023-01752-8

Heston, T. F., & Khun, C. (2023). Prompt engineering in medical education. *International Medical Education*, *2*(3), 198–205.

IBM. (2024). ¿Qué es el prompting? Retrieved October 31, 2024, from https://www.ibm.com/es-es/topics/prompt-engineering

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., et al. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, *55*(12), 1–38.

Jha, S., Jha, S. K., Lincoln, P., Bastian, N. D., Velasquez, A., & Neema, S. (2023). Dehallucinating large language models using formal methods guided iterative prompting. In *IEEE International Conference on Assured Autonomy (ICAA)* (pp. 149–152). IEEE.

Knoth, N., Tolzin, A., Janson, A., & Leimeister, J. M. (2024). AI literacy and its implications for prompt engineering strategies. *Computers and Education: Artificial Intelligence*, *6*, 100225. https://doi.org/10.1016/j.caeai.2024.100225

Lo, L. D. (2023). The art and science of prompt engineering: A new literacy in the information age. *Internet Reference Services Quarterly*, *27*(4), 203–210. https://doi.org/10.1080/10875301.2023.2227621

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, *55*(9), 1–35.

Long, J. (2024). Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*. https://arxiv.org/abs/2305.08291

OpenAI. (2023). ChatGPT. Retrieved October 31, 2024, from https://chatgpt.com/c/6728108e-16d8-800e-bac2-b34d5d674af2

Oppenlaender, J. (2022). A taxonomy of prompt modifiers for text-to-image generation. *arXiv preprint*. https://arxiv.org/pdf/2204.13988

Pornprasit, C., & Tantithamthavorn, C. (2024). Fine-tuning and prompt engineering for large language models-based code review automation. *Information and Software Technology*, *175*, 107523. https://doi.org/10.1016/j.infsof.2024.107523

Shieh, J. (2023). Best practices for prompt engineering with OpenAI API. *OpenAI*. Retrieved October 3, 2023.

Velásquez-Henao, J. D., Franco-Cardona, C. J., & Cadavid-Higuita, L. (2023). Prompt engineering: A methodology for optimising interactions with AI-language models in the field of engineering. *Universidad Nacional de Colombia, Facultad de Minas*. https://research.ebsco.com/c/ernpho/viewer/pdf/tcu5ce2xzf

Wei, J., Wang, X., Zhou, D., et al. (2022). Chain-of-thought prompting elicits reasoning in large language models. *Neural Information Processing Systems*. https://arxiv.org/abs/2201.11903

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., et al. (2023). A prompt pattern catalog to enhance prompt engineering with ChatGPT. *arXiv preprint*. https://arxiv.org/pdf/2302.11382

Wu, T., Terry, M., & Cai, C. J. (2021). AI chains: Transparent and controllable human-AI interaction by chaining large language model prompts. *arXiv preprint*. https://arxiv.org/pdf/2110.01691

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., & Zhou, D. (2023). Self-consistency improves chain-of-thought reasoning in language models. *arXiv preprint arXiv:2203.11171*. https://arxiv.org/abs/2203.11171

Wang, L., Chen, X., Deng, X., et al. (2024). Prompt engineering in consistency and reliability with the evidence-based guideline for LLMs. *npj Digital Medicine*, *7*, 41. https://doi.org/10.1038/s41746-024-01029-4

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., & Narasimhan, K. (2024). Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*.