Enhancing Decision Making Through the Integration of Large Language Models and Operations Research Optimization

Segev Wasserkrug [1], Leonard Boussioux [2,3,4], Dick den Hertog[5], Farzaneh Mirzazadeh[6], Ş. İlker Birbil [5], Jannis Kurtz [5], Donato Maragno [5]
[1] IBM Research - Israel, Haifa, Israel
[2] University of Washington, Michael G. Foster School of Business, Seattle, WA, USA
[3] University of Washington, Paul G. Allen School of Computer Science & Engineering, Seattle, WA, USA
[4] Laboratory for Innovation Science at Harvard (LISH), Cambridge, MA, USA
[5] University of Amsterdam, Amsterdam Business School, Business Analytics Section, The Netherlands
[6] MIT-IBM Watson AI Lab, IBM Research, Cambridge, MA, USA
segevw@il.ibm.com, leobix@mit.edu, d.denhertog@uva.nl, farzaneh@ibm.com, s.i.birbil@uva.nl, j.kurtz@uva.nl, d.maragno@uva.nl

Abstract

Many critical business and societal decisions in areas such as supply chain and healthcare involve numerous potential actions, complex constraints, and goals that can be modeled as objective functions. Mathematical optimization, a core area in Operations Research (OR), provides robust, mathematically grounded methodologies to address such decisions and has shown tremendous benefits in many applications. However, its application requires the creation of accurate and efficient optimization models, necessitating rare expertise and considerable time, creating a barrier to widespread adoption in decision-making. Thus, it is a long-standing goal to make these capabilities widely accessible.

The advent of Large Language Models (LLMs) has made advanced Artificial Intelligence (AI) capabilities widely accessible through natural language. LLMs can accelerate expert work in creating formal models like computer programs, and emerging research indicates they can also speed up the development of optimization models by OR experts. We, therefore, propose integrating and advancing LLM and optimization modeling to empower organizational decision-makers to model and solve such complex problems without requiring deep expertise in optimization.

In this work, we present our vision for democratizing optimization modeling for organizational decision-making by such a combination of LLMs and optimization modeling. We identify a set of fundamental requirements for the vision's implementation and describe the state of the art through a literature survey and some experimentation. We show that a) LLMs already provide substantial novel capabilities relevant to realizing this vision, but that b) major research challenges remain to be addressed. We also propose possible research directions to overcome these gaps. We would like this work to serve as a call to action to bring together the LLM and OR optimization modeling communities to pursue this vision, thereby enabling much more widespread improved decision-making and increasing by orders of magnitude the benefits AI and OR can bring to enterprises and society.

1 Introduction

Many critical real-world decision-making problems in areas like supply chain and healthcare involve numerous possible actions, complex constraints, and objectives that can be modeled with a mathematical function. Mathematical optimization, a core discipline within Operations Research (OR), is well-suited to solving such problems and has delivered significant financial and societal benefits in real-world applications, such as more efficient food delivery by the United Nations (Peters et al. 2022), millions in savings for e-commerce sites (Deng et al. 2023), and improved radiotherapy treatment for cancer patients (Zarepisheh et al. 2022). However, applying mathematical optimization requires significant expertise and time, and most decision-makers lack the necessary skills or access to experts. Thus, it has long been an important goal to make optimization widely accessible to decision-makers.

Large Language Models (LLMs), like ChatGPT (OpenAI 2023a), have democratized AI, making advanced capabilities accessible through natural language and simplifying tasks such as document and code generation. This technology can enhance the effectiveness and productivity of knowledge workers, from accelerating programming with tools like GitHub Copilot (Peng et al. 2023) to enabling business consultants to deliver higher-quality results more quickly (Dell'Acqua et al. 2023).

With the new capabilities provided by LLMs, we believe the time is right to bring together the AI and OR communities to integrate these technologies towards the goal of creating a *Decision Optimization CoPilot* (*DOCP*): an AI tool that interacts with decision-makers in natural language, leveraging the decision-makers' knowledge and feedback to generate and solve problem-specific optimization models.

In this article, we outline the DOCP vision and the core requirements for its implementation, describe the current state of the art, and suggest research directions to address existing gaps. We conclude with a call to action for the AI and OR communities to come together to realize this vision, thereby significantly enhancing the impact of AI and OR on complex, critical decision-making.

2 Theory and Practice of Optimization Modeling

Mathematical optimization requires formally modeling a decision-making problem as:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{g}(\mathbf{x}) \leq 0, \\ & \mathbf{h}(\mathbf{x}) = 0, \end{array} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the vector of decision variables (representing the possible decisions), $f(\cdot)$ is the objective function or goal to optimize, $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ are a set of vector-valued constraint functions that define the set of feasible solutions.

Once a decision problem is modeled as an optimization problem, various algorithms can be used to find a solution, depending on the type of optimization model. Efficient algorithms exist for Linear Optimization (LO), suitable when all functions in (1) are linear. *Nonlinear Optimization* (*NLO*) extends this to cases where the objective or constraint functions are nonlinear. *Mixed Integer Linear Optimization* (*MILO*) addresses problems where the objective and constraint functions are linear, but some or all decision variables are constrained to be integers. MILO is often required for optimization applications in critical domains such as manufacturing and supply chain management. However, most real-world MILO problems are NP-hard, making it computationally challenging to find exact solutions for large-scale instances. Thus, creating the most efficient model is a crucial skill, as NLO, LO and MILO formulations may exist for the same decision problem. Additionally, advanced techniques like *decomposition* (breaking down complex problems into simpler subproblems) and *cut generation* (iteratively adding constraints to simplify the solution search) have been developed to handle MILO problems more efficiently (Wolsey and Nemhauser 1988). These techniques often require problem specific implementation, requiring significant expertise.

Given the complexity of solving real-world MILO problems, the use of optimization engines like Gurobi (Gurobi Optimization, LLC 2024) and CPLEX (IBM 2024) has become common. These engines solve MILO problems using the latest algorithms and offer numerous parameters that optimization experts can set to achieve additional efficiency for the specific model at hand.

2.1 Optimization Modeling

Using optimization to solve a real-world business problem is a complex, multi-step process. It requires a skilled optimization practitioner to work with the decision-maker, following the process depicted in Figure 1.

In this process, the optimization practitioner must thoroughly understand the problem and the available data to create a formal optimization model (in the form of Equation (1)). The practitioner then improves the model, aiming for the most efficient formulation. Advanced techniques like decomposition, cut generation, and fine-tuning optimization engine parameters often need to be applied by the optimization practitioner to ensure the problem is solved efficiently. The final optimization problem and its solutions must then be validated against the real-world decision-making problem, with discrepancies leading to additional iterations. Throughout, the optimization modeler and the decision-maker must continuously interact, with the optimization modeler thinking in terms of mathematical formulations and the decision-maker communicating in terms of the problem domain. Consequently, this process often requires multiple iterations.

3 Our Vision: a Decision Optimization CoPilot

We envision a **Decision Optimization CoPilot (DOCP)**: A friendly AI system incorporating an LLM, accessible to any business decision-maker who needs to make a decision that can benefit from optimization. The DOCP will guide the decision-maker through the decision-making process by interacting in natural language and, behind the scenes, carrying out the steps depicted in Figure 1, ultimately creating an optimization model that provides effective solutions for the specific business task.

As a concrete example of such a decision problem, consider Example 3.1 (based on a real-world use case - see Boutin (2023)).

Example 3.1. A company wants to develop a methodology to identify optimal locations for its bike rental hubs. A hub is composed of a parking facility where users (commuters and tourists) can park their cars, and bicycles are available for rental. The methodology should optimize the maximum coverage of the needs of the users in the cities targeted to maximize the number of users and minimize the number of kilometers driven in the city.

To apply optimization, the decision-maker would start by providing the DOCP with a natural language description of the decision problem, as in the example. Through conversational interaction, the DOCP should first understand the precise objectives, constraints, and available data. Next, the DOCP should validate whether optimization is the appropriate approach and determine the type of optimization model needed (*e.g.*, LO, MILO) based on the problem's characteristics. If optimization is not suitable, the DOCP should inform the user and stop. If it is, the DOCP would then create the optimization model (possible models for the problem in Example 3.1 appear in Wasserkrug et al. (2024)[1]), run the appropriate algorithm (typically through an optimization engine), and provide actionable recommendations to the user.

The DOCP should be capable of receiving user feedback at any stage and incorporating it to course-correct or update the models and processes accordingly. Additionally, the DOCP must communicate in a language that is easily understood by the business decision-maker.

3.1 Core Capabilities Required by a DOCP

Requirement #1 - Translating a Business Level problem definition to an Optimization Model: A key part of the optimization modeling process (Figure 1) is developing a detailed and precise *mental model* of the decision-making problem at a sufficient level of detail and precision so that

[1]This is an older version of the current article that contains more details about models, experiments, etc.

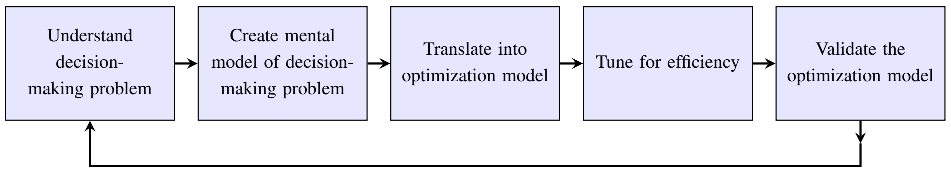

Understand decision-making problem
Create mental model of decision-making problem
Translate into optimization model
Tune for efficiency
Validate the optimization model

Figure 1: Optimization Modeling Process

a formal optimization model can be created. Therefore, The DOCP must guide the decision-maker through this process, starting from a high-level business problem definition and refining it to the detail needed for optimization. Additionally, the DOCP must understand the available data and determine the appropriate type of optimization model.

Requirement #2 - Enabling the Decision Maker to Verify Optimization Model Correctness: As LLMs can produce erroneous output, the optimization model generated by the DOCP could contain errors. While some generated model errors could manifest as runtime errors, there are often more subtle semantic errors that would, therefore, be harder to detect. Such errors could include the generated objective not corresponding to the real-world decision objective or a missing or erroneous constraint. Using such erroneous formulations could result in the model being infeasible or generate solutions that, while optimal for the generated optimization model, are not good solutions for the actual decision-making problem. As decision-makers who do not have optimization expertise cannot be expected to validate the optimization model itself, a DOCP must both a) take whatever steps are possible to reduce the number of errors in the model and b) provide sufficient tools appropriate for the decision-maker to both indirectly validate the model and provide feedback in a way that will enable the DOCP to correct the optimization model.

Requirement #3 - Creating Efficient Optimization Models: Often, solving an optimization model of a real-world problem is NP-hard, and, as a result, many correct optimization models can still not provide useful solutions within a reasonable time frame. In such cases, advanced modeling techniques such as finding equivalent, more efficient models and techniques such as decomposition must often be introduced. Moreover, when using off-the-shelf engines such as CPLEX (see Section 2) to solve such models, engine parameters must be tuned in a model-specific manner. A DOCP needs be able to apply such techniques and engine parameter tuning on behalf of the decision-maker.

4 State-of-the-Art with Regards to DOCP Requirements

In this section, we describe the state-of-the-art capabilities relevant to a DOCP, based both on a survey of related work and experiments with ChatGPT (OpenAI 2023a).

4.1 Related Work

There is An increasing amount of work leveraging LLMs to generate optimization models so as to make optimization more widely accessible. The work of Amarasinghe et al. (2023) states as a goal the ability to "leverage LLMs to support non-expert users to carry out business optimizations without having to consult experts". Focusing on production scheduling, this work constructed an open-source dataset consisting of relatively precise natural language problem descriptions and corresponding model formulations and used it to fine-tune a code-generating LLM to generate optimization models. This approach reported good performance as measured by running the generated models, ensuring that they run correctly, and then comparing both execution time and obtained solution to the "ground truth" models. However, it does not address model validation by non-experts nor explicitly how to generate efficient models. Li, Zhang, and Mak-Hau (2023) proposed a framework for automatic formulation from unstructured text descriptions of MILO problems. It utilized a constraint classification scheme and constraint templates to guide the LLMs in creating MILO models from natural language descriptions. This work relied on OR experts to evaluate the quality of the generated models.

The NL4Opt NeurIPS competition (Ramamonjison et al. 2022b) garnered significant attention in the realm of extracting linear optimization (LO) models from natural language. The competition was inspired by the pilot study of Ramamonjison et al. (2022a), where the authors introduced an automated formulation method named OptGen, designed to handle LO problem descriptions with diverse constraints. OptGen and the NL4Opt competition employ a two-step mapping strategy: In the first step, all problem entities of the optimization problem (e.g., objective function, decision variables, etc.) are detected, and in the second step, the problem description, together with these labeled problem entities, is used to generate the final mathematical representation of the problem. Evaluation is conducted by comparing the predicted and ground truth models using a predefined canonical form.

In their survey paper, Fan et al. (2024) assessed LLMs' performance on textbook problems and real-world scenarios using the NL4OPT dataset and evaluation metrics. LLMs demonstrated impressive proficiency in modeling textbook-level problems; however, in the context of real-world problems, the generated models exhibited errors such as incorrect constraints, extra constraints, and missing constraints. Nonetheless, pointing out errors and prompting corrections proved effective, making LLM-generated formulations more accurate.

More recent works have gone beyond the two-step approach of NL4Opt to address the modeling of optimization problems from natural language by creating LLM based *agentic* systems (Li 2024) - systems consisting of multiple LLMs and additional tools such as optimization engines. Tsouros et al. (2023) proposed a modular framework for problem modeling via LLMs composed of four subtasks, each implemented using prompting (Khot et al. 2022). These subtasks include extracting problem entities, establishing relationships, formalizing the optimization problem, and translating it into a constraint modeling language, like CPMpy (Guns et al. 2024). Following the modeling steps, the code is compiled, executed, and automatically debugged if necessary. The system can also engage in an interactive refinement process with the user. OptiMUS (Ahmaditeshnizi, Gao, and Udell 2024) is another agentic system that includes an "Evaluator" a "Programmer" and a "Formulator" as LLM agents, together with calls to an optimization solver. Xiao et al. (2024) is an agentic system that introduces a "Terminology Interpreter" LLM agent - for incorporating domain or problem-specific terminology in areas such as supply chain, and a "Conductor" - an LLM that coordinates the workflow between the multiple agents in the system.

To summarize, while a lot of work exists in generating models from natural language descriptions, there are still many errors in the models generated by these works. In addition, all of these works require quite a precise natural language description of the problem rather than a high-level description, such as in Example 3.1, and therefore, are not suitable for organizational decision-makers. Zhang et al. (2024) begins to address the organizational decision-maker through a framework called OptLLM, which adds the capability to carry out a natural language dialogue with the user to refine the optimization problem definition. However, in this work, the dialogue capability was only demonstrated on a single example, with the more rigorous testing carried out on precise natural language descriptions.

In addition to the limitations of existing works in addressing the organizational decision makers, only Amarasinghe et al. (2023) addressed the need to generate efficient models, and this also in a limited, implicit fashion (by having execution time as one of the measurement metrics). Finally, none of the above works address the need of a decision-maker to verify the correctness of a generated model or the need to update the model as a result of business user feedback.

4.2 Experimentation Using ChatGPT

We conducted experiments specifically targeting the three requirements described in Section 3.1. We did this using prompting with ChatGPT 4 (OpenAI 2023a).

Requirement #1: From Business Level Problem Definition to Optimization Model In these experiments, we acted as organizational decision-makers, trying to use ChatGPT to generate an accurate optimization model for the problem in Example 3.1 through natural language conversation. We provided context to ChatGPT by creating personalized GPTs - customized versions of ChatGPT with tailored natural language instructions (see OpenAI (2023b)). We tested two GPTs with slightly different instructions, guiding ChatGPT to converse with the business user to understand the problem, identify the required optimization model,

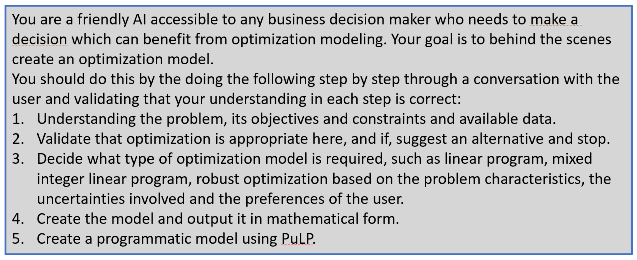
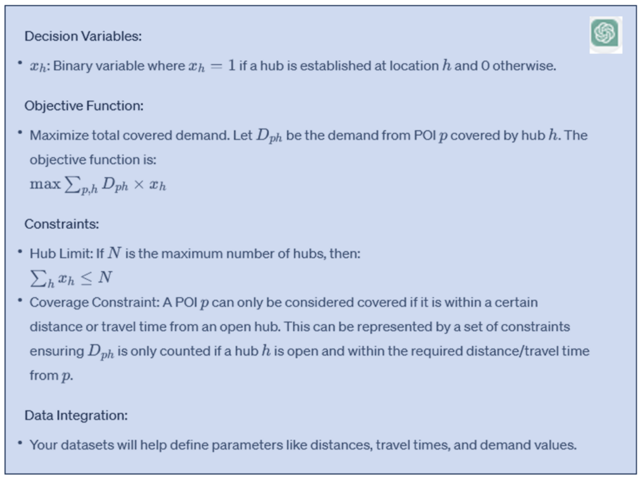

You are a friendly AI accessible to any business decision maker who needs to make a decision which can benefit from optimization modeling. Your goal is to behind the scenes create an optimization model.
You should do this by the doing the following step by step through a conversation with the user and validating that your understanding in each step is correct:
1. Understanding the problem, its objectives and constraints and available data.
2. Validate that optimization is appropriate here, and if, suggest an alternative and stop.
3. Decide what type of optimization model is required, such as linear program, mixed integer linear program, robust optimization based on the problem characteristics, the uncertainties involved and the preferences of the user.
4. Create the model and output it in mathematical form.
5. Create a programmatic model using PuLP.

Figure 2: The context provided to one of the custom GPTs

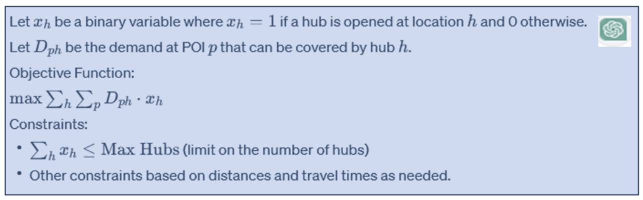

Decision Variables:
- x_h: Binary variable where $x_h = 1$ if a hub is established at location h and 0 otherwise.

Objective Function:
- Maximize total covered demand. Let D_{ph} be the demand from POI p covered by hub h. The objective function is:
$\max \sum_{p,h} D_{ph} \times x_h$

Constraints:
- Hub Limit: If N is the maximum number of hubs, then:
$\sum_h x_h \leq N$
- Coverage Constraint: A POI p can only be considered covered if it is within a certain distance or travel time from an open hub. This can be represented by a set of constraints ensuring D_{ph} is only counted if a hub h is open and within the required distance/travel time from p.

Data Integration:
- Your datasets will help define parameters like distances, travel times, and demand values.

(a) Model Generated by 1st GPT


Let x_h be a binary variable where $x_h = 1$ if a hub is opened at location h and 0 otherwise.
Let D_{ph} be the demand at POI p that can be covered by hub h.
Objective Function:
$\max \sum_h \sum_p D_{ph} \cdot x_h$
Constraints:
- $\sum_h x_h \leq \text{Max Hubs}$ (limit on the number of hubs)
- Other constraints based on distances and travel times as needed.

(b) Model Generated by 2nd GPT

Figure 3: Optimization models generated by ChatGPT (Requirement #1)

and create both a mathematical formulation and executable code for solving the model using an optimization engine (the prompt for one of the GPT's appear in Figure 2 and the full conversations appear in Wasserkrug et al. (2024)).

Both conversations started with a brief description of the problem as appearing in Example 3.1. In each case, ChatGPT correctly asked for further clarification and ultimately generated a mathematical formulation of an optimization model. Moreover, ChatGPT selected an appropriate model type for this problem - MILO.

However, in both cases, the optimization problem generated by ChatGPT (see Figure 3) was incomplete and unsuitable for the business problem. The model incorrectly assigned a demand to a combination of a point of interest and a potential hub location (parameter $D_{p,h}$) rather than a more granular demand between a hub, a point of interest, and a junction (a point of entry of commuters into the city), as required in this specific problem. This error occurred despite the user explicitly stating that the available data was the demand from a junction to a point of interest. As an example of incompleteness, the model lacked constraints indicating whether a hub should cover a specific demand. Although ChatGPT asked for clarification to enhance the constraints, it also requested user feedback on the mathematical formulation despite being informed that it was interacting with a business decision-maker. (Figure 2). Finally, it did not directly generate the code but waited for feedback on the mathematical formulation.

In summary, the GPTs demonstrated impressive, relevant capabilities: they interacted with the user to understand the business problem and generated an optimization formulation. However, they did not sufficiently interact with the user to fully grasp the problem before creating a model, resulting in an unsuitable and incomplete formulation. Furthermore, after generating the mathematical formulation, they asked for validation of the model's correctness before proceeding – an unrealistic request from a business user.

Requirement #2: Verifying Optimization Model Correctness In these experiments, we tested ChatGPT's ability to generate correct models for optimization problems when given precise problem descriptions rather than high-level ones, such as in Example 3.1 (the full details of the experiments appear in Wasserkrug et al. (2024)). The optimization problems were optimizing the location of vaccination clinics and optimizing radiotherapy treatment for cancer patients (a problem similar to the one in (Zarepisheh et al. 2022)). Given ChatGPT's non-deterministic outputs, we ran ChatGPT five times on each problem. ChatGPT generally performed well, providing correct models four out of five times for both problems. However, for both problems, in one out of the five times, it generated an incorrect objective function (the correct and incorrect objective functions for both experiments appear in Figure 4). For the vaccination clinic problem, it missed the crucial term r_i which is required to adjust demand by assigning residents to clinics based on the area (see Figure 4(a) and 4(b) for the correct and incorrect objective function, respectively). For the radiotherapy treatment problem, one attempt incorrectly provided a nonlinear formulation of the problem by incorporating the BED nonlinear function (see Figure 4(d)), despite a linear formulation being explicitly requested. In the other four attempts, it correctly provided a linear objective function by calculating and summing the BED values for the input data (Figure 4(c)). Such relatively subtle mistakes, which are difficult even for novice optimization modelers to spot, highlight the need for higher-quality model generation, robust validation mechanisms for decision-makers, and the ability to incorporate user feedback to improve model accuracy.

Requirement #3: Creating Efficient Optimization models In these experiments, our goal was to test whether ChatGPT can create models that are more efficiently solvable.

We started by assessing ChatGPT's ability to simplify a

$$\text{Minimize} \quad \sum_{i=1}^{50}\sum_{j=1}^{10} r_i \times d_{ij} \times Y_{ij}$$

(a) Correct Objective Function
(Clinic Placement)

$$\text{Minimize} \quad \sum_{i=1}^{50}\sum_{j=1}^{10} d_{ij} \times Y_{ij}$$

(b) Incorrect Objective Function
(Clinic Placement)

$$\text{Maximize} \sum_{i=1}^{17}\sum_{p=0}^{15} \mathrm{BED}_i(p, 15-p) \times x_{i,p}$$

(c) Correct Objective Function
(Radiotherapy)

$$\textbf{Maximize} \textstyle\sum_{i=1}^{17} \mathrm{BED}_i(x_i, 15 - x_i)$$

(d) Incorrect Objective Function
(Radiotherapy)

Figure 4: Objective Functions Generated by ChatGPT (Requirement 2)

correct formulation of the bike hub optimization model by prompting it to restructure the problem into an equivalent form with fewer variables. In this experiment, ChatGPT succeeded in only one out of five attempts.

In the four instances where the formulations were incorrect, ChatGPT exhibited various errors, including introducing nonlinear constraints or objective functions and incorporating constraints that only included parameters without variables (for the full experiment and list of errors, see Wasserkrug et al. (2024)). Additionally, ChatGPT could not produce satisfactory reformulations without explicit instructions to maintain an equivalent form while minimizing variables. With more open-ended instructions, ChatGPT struggled, often oversimplifying the problem by omitting constraints or by incorrectly redefining sets.

In another experiment, we presented ChatGPT with a constraint involving absolute values and asked it if this constraint could be represented by a more efficient linear constraint. For convex $\leq$ constraints, ChatGPT's answers were always correct. However, for concave $\geq$ constraints, it gave incorrect answers. Issues also arose in cases involving the sum of maximum of linear functions, which often appear in production-inventory problems and which can be reformulated as linear constraints. Although ChatGPT found a linear reformulation, it neglected introducing binary variables for the $\geq$ constraint.

Overall, in a large number of these experiments ChatGPT failed to formulate efficient optimization models.

4.3 Summary of State of the Art

LLMs and the agentic paradigm provide important capabilities in realizing our DOCP vision. However, most previous works have focused on generating correct optimization models from detailed natural language descriptions (Requirement #2). Little work has addressed generating models from business-level problem descriptions or ensuring that models are efficiently solvable (Requirement #1 and #3). Our experiments also demonstrated significant gaps in these areas. Even for requirement #2, incorrect models are often generated, sometimes with subtle, hard to detect, errors. Therefore, Much work remains to achieve the DOCP vision.

5 Research Directions for Creating a DOCP

In creating a DOCP, LLM adaptation can play an important role. With the various methods available for task-specific adaptation, such as prompting (Brown et al. 2020; Sanh et al.

2021; Khot et al. 2022), supervised fine-tuning (Dai and Le 2015; Dodge et al. 2020; Wei et al. 2022a), and reinforcement learning with human feedback (RLHF) (Ziegler et al. 2019; Stiennon et al. 2020; Ouyang et al. 2022), an important research direction is determining how best to combine these techniques. For example: Should LLMs be fine-tuned with decision optimization knowledge? What can prompting techniques like *Chain of Thought* (Wei et al. 2022b), found effective in reasoning tasks, achieve? Which prompting techniques are best for each requirement? How can LLMs be adapted to generate correct and efficient optimization models, reduce variables, reformulate nonlinear models into linear ones, and incorporate problem-specific decomposition or cut generation?

A promising approach to augment an individual LLM's capability is through the agentic systems paradigm. Therefore, a key research direction is constructing an appropriate agentic system to implement a DOCP. Questions to address include which parts of such an agentic system should be driven by the LLM and which should be managed by external workflows like Graph of Thoughts (Besta et al. 2023) (a more structured reasoning flow that can be implemented, for example, through code). For instance, could leveraging existing methodological frameworks, like *CRISP-DM* (CRoss Industry Process for Data Mining (Wirth and Hipp 2000)), which guide analytics practitioner through the process described in Figure 1), inform a successful graph-of-thought DOCP implementation?

Regarding Requirement #2, since there is always a risk of generating an incorrect model, tools are needed to help business users validate solutions provided by a DOCP. Even today, business users often struggle to validate and understand solutions from expert-created optimization models, relying on tools like what-if analysis (which allows business users to modify a solution or propose an alternative one to see which constraints are violated and how the objective function value is affected). LLM capabilities could enhance these techniques (see, for example, Li et al. (2023)), enabling users to have natural language conversations for what-if anaylsis, better model understanding, and validation.

Another important research direction for Requirement #2 is how to update a generated model and the DOCP's capabilities based on user feedback. For instance, when carrying out what-if analysis, a user might discover that the wrong objective has been calculated or a business constraint was incorrectly flagged as violated. Users might also be able to provide solutions that satisfy the business constraints along with their objective function value. It is important to understand both how to correct a specific generated model and continuously improve the DOCP based on such feedback.

Regarding requirement #3, the parameter settings in an optimization engine, like which heuristics to use, can significantly impact performance. A substantial body of research, known as *Learning to Optimize* (Chen et al. 2022; Chi et al. 2022; Gupta et al. 2022; Khalil, Morris, and Lodi 2022) uses machine learning to find effective heuristic settings for a specific problem, reducing solution times. Currently, optimization experts tailor these techniques to specific problem instances. An interesting research direction would be to explore whether LLMs could be adapted to select and customize these techniques in a problem-specific way. Additionally, could the vast online knowledge of optimization theory and practice, including problem modeling and best practices (see, *e.g.*, Williams (2013)), be leveraged to fine-tune LLMs to apply techniques such as decomposition and cut generation, formulating more efficient optimization models, and setting engines' parameters?

5.1 Datasets and Benchmarks

Research toward a DOCP will require appropriate datasets and well-defined quality metrics. These datasets should include a set of optimization problems, each with a business-level description and a correct, efficient 'ground truth' optimization model. Journals (*e.g.*, IJAA (2024)) and books (Williams 2013) that describe real-world case studies and their models could serve as a source for such datasets. Existing datasets from works that combine LLMs and optimization, like Amarasinghe et al. (2023), could also be extended. Beyond natural language descriptions and 'ground truth' models, since the DOCP must converse with the business user to understand the problem and create a model, a component that simulates the business user's side of the conversation is needed. An interesting research direction would be exploring whether an LLM independent of the DOCP could emulate the business user, enabling DOCP development and testing.

Regarding evaluation, many existing works (see Section 4.1) measure the syntactical accuracy of the generated models against a 'ground truth' model. However, the goal of a DOCP is to generate high-quality solutions to the decision problem, and two syntactically different models could yield constraint-satisfying optimal solutions. Therefore, while the precise definition of these metrics and their computation is another important avenue for future work, we believe they should focus on whether the DOCP solutions satisfy the ground truth model's constraints, their quality in terms of objective value, and their execution time (potentially building on and extending metrics defined by Amarasinghe et al. (2023)).

6 Summary and Call to Action

In this work, we present our vision to combine LLMs and optimization modeling to democratize the use of optimization for decision-making, enabling any organizational decision-maker to make better decisions in complex settings. We outlined three core requirements and discussed the current state of the art, showing that the capabilities of LLMs and the agentic system paradigm have brought us closer than ever before to making optimization widely accessible. We have also demonstrated that significant gaps remain and proposed directions to start addressing these gaps. We would therefore like to conclude with a call to action to the AI and OR communities to come together to advance the research and practice at the intersection of LLMs and optimization to realize this vision, thereby resulting both in theoretical advancements in the fields of LLMs, agentic systems, OR, and their intersection and increasing by orders of magnitude the benefit that AI and OR can bring to enterprises and society.

Ethics Statement

In any optimization solution, there are dangers if the recommendations are followed blindly and poorly understood by the **decision-maker** whose responsibility is to make the decision. Introducing a DOCP does not and should not lessen the decision-maker's responsibility. When collaborating with an optimization modeler, there's the added benefit of a knowledgeable professional who can provide insights. This expert not only grasps the intricacies of the model but can also elucidate the underlying assumptions and its potential limitations for the business decision-maker. Therefore, moving to using a DOCP underscores the need for, and importance of, capabilities that allow the business decision-maker to understand the proposed solution and its limitations deeply. In high-stakes cases, the business decision-maker should still consult with an optimization modeler to review the optimization model created by the DOCP. Our proposed approach, which is to have the LLM generate an optimization model rather than directly produce a solution is, therefore, also beneficial from this perspective as it enables an optimization modeler to review and understand the assumptions and limitations of a mathematical optimization model. Such model generation allows for a more transparent and reviewable decision-making process, ensuring that the decision-maker is fully informed of all aspects of the solution.

In addition, in very high-risk cases, it may be better for the business decision-maker to forgo the direct use of a DOCP and instead engage with an optimization modeler, who may then leverage the DOCP to accelerate the creation of the optimization solution, ensuring a more informed and cautious approach to decision-making.

References

Ahmaditeshnizi, A.; Gao, W.; and Udell, M. 2024. OptiMUS: Scalable Optimization Modeling with (MI)LP Solvers and Large Language Models. In Salakhutdinov, R.; Kolter, Z.; Heller, K.; Weller, A.; Oliver, N.; Scarlett, J.; and Berkenkamp, F., eds., *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 577–596. PMLR.

Amarasinghe, P. T.; Nguyen, S.; Sun, Y.; and Alahakoon, D. 2023. AI-Copilot for Business Optimisation: A Framework and A Case Study in Production Scheduling. *arXiv preprint arXiv:2309.13218*.

Besta, M.; Blach, N.; Kubicek, A.; Gerstenberger, R.; Gianinazzi, L.; Gajda, J.; Lehmann, T.; Podstawski, M.; Niewiadomski, H.; Nyczyk, P.; et al. 2023. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*.

Boutin, B. 2023. Optimizing park and bike hub locations for sustainable urban mobility. *Master Thesis, University of Amsterdam*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Neural Information Processing Systems*.

Chen, T.; Chen, X.; Chen, W.; Wang, Z.; Heaton, H.; Liu, J.; and Yin, W. 2022. Learning to Optimize: A Primer and a Benchmark. *J. Mach. Learn. Res.*, 23(1).

Chi, C.; Aboussalah, A. M.; Khalil, E. B.; Wang, J.; and Sherkat-Masoumi, Z. 2022. A Deep Reinforcement Learning Framework For Column Generation. In *Neural Information Processing Systems (NeurIPS)*.

Dai, A. M.; and Le, Q. V. 2015. Semi-supervised Sequence Learning. In *Neural Information Processing Systems*.

Dell'Acqua, F.; McFowland, E.; Mollick, E. R.; Lifshitz-Assaf, H.; Kellogg, K.; Rajendran, S.; Krayer, L.; Candelon, F.; and Lakhani, K. R. 2023. Navigating the Jagged Technological Frontier: Field Experimental Evidence of the Effects of AI on Knowledge Worker Productivity and Quality. *Harvard Business School Technology & Operations Mgt. Unit Working Paper No. 24-013*. Available at SSRN: https://ssrn.com/abstract=4573321 or http://dx.doi.org/10.2139/ssrn.4573321.

Deng, Y.; Zhang, X.; Wang, T.; Wang, L.; Zhang, Y.; Wang, X.; Zhao, S.; Qi, Y.; Yang, G.; and Peng, X. 2023. Alibaba Realizes Millions in Cost Savings Through Integrated Demand Forecasting, Inventory Management, Price Optimization, and Product Recommendations. *INFORMS Journal on Applied Analytics*, 53(1): 32–46.

Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; and Smith, N. 2020. Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping. *arXiv preprint arXiv:2002.06305*.

Fan, Z.; Ghaddar, B.; Wang, X.; Xing, L.; Zhang, Y.; and Zhou, Z. 2024. Artificial Intelligence for Operations Research: Revolutionizing the Operations Research Process. *arXiv preprint arXiv:2401.03244*.

Guns, T.; et al. 2024. CPMpy: Constraint Programming in Python. https://github.com/CPMpy/cpmpy. Accessed: 2024-08-28.

Gupta, P.; Khalil, E. B.; Chetélat, D.; Gasse, M.; Bengio, Y.; Lodi, A.; and Kumar, M. P. 2022. Lookback for learning to branch. *Transactions of Machine Learning Research (TMLR)*.

Gurobi Optimization, LLC. 2024. Gurobi Optimizer Reference Manual.

IBM. 2024. IBM ILOG CPLEX Optimizer.

IJAA. 2024. INFORMS Journal on Applied Analytics. https://pubsonline.informs.org/loi/ijaa. Accessed: 2024-01-21.

Khalil, E. B.; Morris, C.; and Lodi, A. 2022. MIP-GNN: A Data-Driven Framework for Guiding Combinatorial Solvers. In *The AAAI Conference on Artificial Intelligence*.

Khot, T.; Trivedi, H.; Finlayson, M.; Fu, Y.; Richardson, K.; Clark, P.; and Sabharwal, A. 2022. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. *arXiv preprint arXiv:2210.02406*.

Li, B.; Mellou, K.; Zhang, B.; Pathuri, J.; and Menache, I. 2023. Large Language Models for Supply Chain Optimization. *arXiv preprint arXiv:2307.03875*.

Li, Q.; Zhang, L.; and Mak-Hau, V. 2023. Synthesizing Mixed-Integer Linear Programming Models from Natural Language Descriptions. *arXiv preprint arXiv:2311.15271*.

Li, X. 2024. A Survey on LLM-Based Agents: Common Workflows and Reusable LLM-Profiled Components. arXiv:2406.05804.

OpenAI. 2023a. ChatGPT: Language Model for Conversational Agents. Accessed: 1 January 2024.

OpenAI. 2023b. Introducing GPTs. Accessed: 1 January 2024.

Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Peng, S.; Kalliamvakou, E.; Cihon, P.; and Demirer, M. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. arXiv:2302.06590.

Peters, K.; Silva, S.; Wolter, T. S.; Anjos, L.; van Ettekoven, N.; Combette, E.; Melchiori, A.; Fleuren, H.; den Hertog, D.; and Ergun, Ö. 2022. UN World Food Programme: Toward Zero Hunger with Analytics. *Informs Journal on Applied Analytics*, 52(1): 8–26.

Ramamonjison, R.; Li, H.; Yu, T.; He, S.; Rengan, V.; Banitalebi-dehkordi, A.; Zhou, Z.; and Zhang, Y. 2022a. Augmenting Operations Research with Auto-Formulation of Optimization Models From Problem Descriptions. In Li, Y.; and Lazaridou, A., eds., *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 29–62. Abu Dhabi, UAE: Association for Computational Linguistics.

Ramamonjison, R.; Yu, T.; Li, R.; Li, H.; Carenini, G.; Ghaddar, B.; He, S.; Mostajabdaveh, M.; Banitalebi-Dehkordi, A.; Zhou, Z.; and Zhang, Y. 2022b. NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions. In Ciccone, M.; Stolovitzky, G.; and Albrecht, J., eds., *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, 189–203. PMLR.

Sanh, V.; Webson, A.; Raffel, C.; Bach, S. H.; Sutawika, L.; Alyafeai, Z.; Chaffin, A.; Stiegler, A.; Scao, T. L.; Raja, A.; et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Stiennon, N.; Ouyang, L.; Wu, J.; Ziegler, D.; Lowe, R.; Voss, C.; Radford, A.; Amodei, D.; and Christiano, P. F. 2020. Learning to Summarize from Human Feedback. In *Neural Information Processing Systems*.

Tsouros, D.; Verhaeghe, H.; Kadıoğlu, S.; and Guns, T. 2023. Holy Grail 2.0: From Natural Language to Constraint Models. *arXiv preprint arXiv:2308.01589*.

Wasserkrug, S.; Boussioux, L.; den Hertog, D.; Mirzazadeh, F.; Birbil, I.; Kurtz, J.; and Maragno, D. 2024. From Large Language Models and Optimization to Decision Optimization CoPilot: A Research Manifesto. arXiv:2402.16269.

Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022a. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; ichter, b.; Xia, F.; Chi, E.; Le, Q. V.; and Zhou, D. 2022b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems*, volume 35, 24824–24837. Curran Associates, Inc.

Williams, H. P. 2013. *Model Building in Mathematical Programming, 5th Edition*. Wiley. ISBN 978-1-118-44333-0.

Wirth, R.; and Hipp, J. 2000. CRISP-DM: Towards a standard process model for data mining. In *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, volume 1, 29–39. Manchester.

Wolsey, L.; and Nemhauser, G. 1988. *Integer and Combinatorial Optimization*. Wiley Series in Discrete Mathematics and Optimization. Wiley. ISBN 9780471828198.

Xiao, Z.; Zhang, D.; Wu, Y.; Xu, L.; Wang, Y. J.; Han, X.; Fu, X.; Zhong, T.; Zeng, J.; Song, M.; and Chen, G. 2024. Chain-of-Experts: When LLMs Meet Complex Operations Research Problems. In *The Twelfth International Conference on Learning Representations*.

Zarepisheh, M.; Hong, L.; Zhou, Y.; Huang, Q.; Yang, J.; Jhanwar, G.; Pham, H. D.; Dursun, P.; Zhang, P.; Hunt, M. A.; Mageras, G. S.; Yang, J. T.; Yamada, Y. J.; and Deasy, J. O. 2022. Automated and Clinically Optimal Treatment Planning for Cancer Radiotherapy. *INFORMS Journal on Applied Analytics*, 52(1): 69–89.

Zhang, J.; Wang, W.; Guo, S.; Wang, L.; Lin, F.; Yang, C.; and Yin, W. 2024. Solving General Natural-Language-Description Optimization Problems with Large Language Models. In Yang, Y.; Davani, A.; Sil, A.; and Kumar, A., eds., *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, 483–490. Mexico City, Mexico: Association for Computational Linguistics.

Ziegler, D. M.; Stiennon, N.; Wu, J.; Brown, T. B.; Radford, A.; Amodei, D.; Christiano, P. F.; and Irving, G. 2019. Fine-Tuning Language Models from Human Preferences. *CoRR*, abs/1909.08593.