# Large Language Models for Combinatorial Optimization: A Systematic Review

FRANCESCA DA ROS, Mathematics, Computer Science and Physics, University of Udine, Udine, Italy
MICHAEL SOPRANO, Mathematics, Computer Science and Physics, University of Udine, Udine, Italy
LUCA DI GASPERO, Polytechnic Department of Engineering and Architecture, University of Udine, Udine, Italy
KEVIN ROITERO, Mathematics, Computer Science, and Physics, University of Udine, Udine, Italy

This systematic review explores the application of Large Language Models (LLMs) in Combinatorial Optimization (CO). We report our findings using the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) guidelines. We conduct a literature search via Scopus and Google Scholar, examining over 2,000 publications. We assess publications against four inclusion and four exclusion criteria related to their language, research focus, publication year, and type. Eventually, we select 103 studies. We classify these studies into semantic categories and topics to provide a comprehensive overview of the field, including the tasks performed by LLMs, the architectures of LLMs, the existing datasets specifically designed for evaluating LLMs in CO, and the field of application. Finally, we identify future directions for leveraging LLMs in this field.

CCS Concepts: • **General and reference** → **Surveys and overviews**; • **Computing methodologies** → **Natural language processing**; • **Theory of computation** → **Discrete optimization;**

Additional Key Words and Phrases: Systematic review, large language models, combinatorial optimization

**ACM Reference Format:**
Francesca Da Ros, Michael Soprano, Luca Di Gaspero, and Kevin Roitero. 2026. Large Language Models for Combinatorial Optimization: A Systematic Review. *ACM Comput. Surv.* 58, 11, Article 272 (April 2026), 53 pages. https://doi.org/10.1145/3801961

## 1 Introduction

**Combinatorial Optimization Problems (COPs)** are a class of optimization problems characterized by discrete variable domains and finite search space. **Combinatorial Optimization (CO)** plays a crucial role in identifying promising solutions in many areas requiring complex decision-making

This research is partially supported by the PRIN 2022 Project—"MoT—The Measure of Truth: An Evaluation-Centered Machine-Human Hybrid Framework for Assessing Information Truthfulness"—Code No. 20227F2ZN3, CUP No. G53D23002800006 Funded by the European Union—Next Generation EU–PNRR M4 C2 I1.1., by the Strategic Plan of the University of Udine–Interdepartment Project on Artificial Intelligence (2020–2025), and by the Strategic Plan of the University of Udine (2022–2025).
Authors' Contact Information: Francesca Da Ros, Mathematics, Computer Science and Physics, University of Udine, Udine, Udine, Italy; e-mail: francesca.daros@uniud.it; Michael Soprano (corresponding author), Mathematics, Computer Science and Physics, University of Udine, Udine, Udine, Italy; e-mail: michael.soprano@uniud.it; Luca Di Gaspero, Polytechnic Department of Engineering and Architecture, University of Udine, Udine, Udine, Italy; e-mail: luca.digaspero@uniud.it; Kevin Roitero, Mathematics, Computer Science, and Physics, University of Udine, Udine, Italy; e-mail: kevin.roitero@uniud.it.


ACM 0360-0300/2026/04-ART272
https://doi.org/10.1145/3801961

capabilities, such as industrial [220] and employee scheduling [25, 102], facility location [27, 64], and timetabling [198, 255]. Traditionally, such problems are modeled with techniques like **Linear Programming (LP)**, **Integer Linear Programming (ILP)**, **Mixed Integer Linear Programming (MILP)**, and **Constraint Programming (CP)**, further solved through commercial solvers such as IBM ILOG CPLEX [88] or Gurobi [70] and through heuristic and **Metaheuristic (MH)** algorithms [195].

While many successful CO applications have been developed, the design and engineering of optimization tasks remain primarily human-driven. Users must convert the problem into an optimization model by defining a set of variables, constraints, and one or more objective functions, then coding and running a software solver or algorithm to find solutions. These activities are not trivial and require a certain extent of expertise.

Inspired by the recent advancements in the usage of **Large Language Models (LLMs)** to perform a wide array of complex tasks, there is growing interest in integrating LLMs into CO to mitigate the human-intensive aspects of optimization [53, 84, 145, 237]. The abilities of LLMs to process, interpret, and generate human language make them particularly suited for tackling activities within CO, including the translation of natural language descriptions to formalisms such as mathematical models [74, 89] and code generation [111, 213].

The rapid advancement in **Artificial Intelligence (AI)** and in particular in **Natural Language Processing (NLP)** has led to a rapid increase in the capabilities and applications of LLMs, resulting in a proliferation of scholarly works and models being developed. While highlighting the increasing activity in the field, this multitude of studies has created a complex body of knowledge, that is, challenging to navigate. Looking specifically at the application of LLMs to CO, the academic literature is limited and fragmented, with existing works characterized by diverse methodologies, areas of applications, and findings.

Therefore, this systematic review aims to consolidate the current state-of-the-art in LLMs applied to CO. We identify, screen, analyze, and systematically organize the literature to clarify the topic and identify strategic directions for ongoing and future research efforts. The process is reported following the **Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA)** guidelines. Through this examination, we seek to understand the capabilities of LLMs in addressing complex optimization tasks and to explore the evolving trends and directions in this field. By systematically synthesizing and analyzing existing research, this review aims to provide a structured understanding of how LLMs are employed in CO and offer insights that can inform future research in the field.

The remainder of this review is structured as follows. In Section 2, we discuss the aims and motivations that drove our work. We then explore the relationships and differences with related work in Section 3. In Section 4, we present the background necessary to understand the interconnections between LLMs and CO. We provide a detailed account of the methodology we followed in Section 5. In Section 6, we classify and discuss the studies included in our review. Next, we outline future research directions in Section 7 and discuss the limitations of our approach in Section 8. Finally, we draw some conclusions and propose future work in Section 9.

## 2 Aims and Motivations

The main objective of this systematic review is to critically evaluate the current state of LLMs application in CO. To this end, we aim to answer the following questions: *(i)* How are LLMs currently being applied to COPs? *(ii)* Which tasks of the optimization process are aided with LLMs? *(iii)* Which LLM architectures and training paradigms are most effective for CO? *(iv)* Which application fields are employing LLMs within CO? *(v)* What are the main trends in the application of LLMs within CO? *(vi)* What are possible research directions in the application of LLMs to CO?

Two primary motivations drive this systematic review. Firstly, traditional optimization approaches primarily rely on human expertise (e.g., application domain knowledge and coding capabilities), thus limiting their applicability, scalability, and efficiency. Leveraging LLMs offers a promising direction, as their success in tasks like entity recognition, domain knowledge, and coding suggests their potential for CO. For instance, LLMs could be used for translating high-level descriptions of solution algorithms to executable code [125].

Secondly, the rapid evolution of NLP (in particular of LLMs) and their application in optimization techniques (specifically CO), highlights the urgency of a comprehensive systematic review. Existing literature and reviews are scattered, lacking a cohesive framework that provides researchers and practitioners with clear guidance in the domain of LLMs applied to CO.

While exploring the reverse relationship between LLMs and CO (i.e., applying CO techniques to enhance LLMs) is also a valid research direction, we deliberately chose not to include it in this survey. This is primarily because such applications essentially use well-established CO methods in a new, albeit challenging, domain. Instead, we focus on the novel and emerging contributions of LLMs to the practice of CO, as this also aligns more closely with the primary research interests of some of the authors, particularly in exploring the potential for LLMs to enhance CO techniques. For the same reason, our review emphasizes CO rather than other paradigms, such as continuous optimization. This decision is also motivated by the inherent differences between the two paradigms: continuous optimization typically relies on mathematical models or black-box ones, whereas CO encompasses a broader and more diverse set of problem domains (e.g., scheduling, assignment, routing, and permutation-based problems) characterized by intricate substructures and complex constraints. This diversity makes the application of LLMs to CO potentially more impactful, given the richness and complexity of the CO landscape.

## 3 Related Work

A limited number of surveys have addressed the intersection of LLM and CO along with related topics. It is important to remark that none of these works are systematic reviews. In this section, we overview their main characteristics and highlight the differences with respect to our work.

Fan et al. [53] presented a review of AI applications in **Operations Research (OR)**, with a focus on three specific aspects of the optimization process: conversion of data into modeling parameters, model formulations, and model optimization. Although they extensively covered various AI techniques, such as **Reinforcement Learning (RL)** and **Neural Networks (NNs)**, the usage of LLMs was investigated only in the context of model formulation [53, Section 4]. Huang et al. [84] explored the topic by including not only CO, but also continuous optimization and the use of optimization for LLMs. However, when discussing LLMs for optimization, they examined a set of 20 papers and focused on a narrow range of optimization aspects, specifically algorithm generation and the usage of LLMs as search operators. Lai et al. [109] reviewed the literature on LLMs concerning their mathematical capabilities and briefly discussed their applications in CO along with math word problems, geometry problems, and theorem proving. Liu et al. [128] described the application of LLMs to algorithm design in various fields, like optimization, **Machine Learning (ML)**, mathematical reasoning, and scientific discovery. Wu et al. [237] provided a review on the intersection of **Evolutionary Computation (EC)** and LLMs. As done by Huang et al. [84], they also covered aspects like continuous optimization and the use of optimization for LLMs enhancement. However, their focus on EC represented only a partial view of the CO landscape and, more generally, of optimization techniques. Similarly, Yu and Liu [247] and Cai et al. [23] described the evolution of **Evolutionary Algorithms (EAs)** to solve optimization problems from heuristic approaches to LLMs.

Our systematic review differ from the aforementioned surveys in the following ways: *(i)* We systematically reviewed the topic using a rigorous methodology to identify, screen, include, and

analyze relevant literature works. We reported the process following the PRISMA 2020 guidelines. *(ii)* We limited our focus to the usage of LLMs within CO, thus excluding optimization areas outside discrete combinatorial optimization, as well as the use of optimization within LLMs, nor, more generally, the use of LLMs for mathematical problem-solving. *(iii)* We addressed the entire optimization process, not focusing solely on specific parts. *(iv)* We considered a broader set of optimization techniques and algorithms, not just evolutionary techniques. *(v)* We described datasets, frameworks, tools, and metrics that could enhance applications of LLMs within CO.

## 4 Background

### 4.1 Large Language Models

LLMs have fundamentally transformed the landscape of NLP and related fields. At the core of this transformation is the Transformer architecture introduced by Vaswani et al. [221], which revolutionized the NLP field with the introduction of an improved self-attention mechanism built on top of the one proposed by Bahdanau et al. [12]. This mechanism allows models to weigh the importance of different words in a sentence, irrespective of their distance, leading to more contextually aware representations.

The Transformer architecture operates with an arrangement of two specified modules, an encoder and a decoder, designed to transform the input data into more abstract and general-purpose representations. Each encoder and decoder in the architecture is built from layers that perform specific functions. First, positional encoding is introduced to inject information about the order of input words into the model, which is crucial for processing sequences where the arrangement of elements carries meaning, as is common in NLP. Following positional encoding, the embedding layer converts the input tokens (i.e., individual pieces of text and typically words or sub-words) into vectors of continuous numbers. This transformation turns linguistic information into a mathematical form that neural networks can process. Once the input has been encoded and embedded, it passes through the Transformer's core mechanisms, i.e., the attention layers. The encoder relies on a self-attention mechanism to independently assess and emphasize different parts of the input data. This allows the model to understand each input segment in relation to the rest of the input, thus enhancing the context awareness of the system. In parallel, the decoder also employs a self-attention layer but focuses on generating the next output token in the sequence conditioned on the tokens generated so far. This ensures that each generated element is contextually aligned with the previous (i.e., already generated) text. Additionally, cross-attention layers in the decoder access the encoder's output to guide the generation process. These layers allow the decoder to reference the full context provided by the encoder, ensuring that each output token is a contextually appropriate continuation or response to the specific provided input sequence. This comprehensive mechanism of encoding, self-attention, and cross-attention within the Transformer allows for highly effective processing and generation of text, making it a robust model for various complex language understanding and generation tasks.

Building on the innovative self-attention mechanism of the Transformer, subsequent models have been developed with specialized architectures tailored for different tasks. `ELMo` [170] defined the usage on contextual embeddings. Encoder-only models, like `BERT` [44], specialize in tasks that require a deep understanding of language context, making them ideal for applications like sentiment analysis and named entity recognition. On the other hand, decoder-only models, such as `GPT-3` [21] and `GPT-4` [22, 161], excel in generating coherent and contextually appropriate text, powering applications in creative writing and dialogue systems.

After encoder and decoder-only models, the introduction of instruction-based models marks a significant evolution in LLMs. These models are trained to follow user-provided instructions [34, 162], making them versatile tools across various tasks without needing task-specific fine-tuning. Instruction-based training involves exposing the model to various tasks during training, along

with corresponding instructions, thereby enabling the model to generalize from instructions at inference time. This approach has started the development of emerging abilities in LLMs, such as zero-shot learning capabilities [104], complex reasoning strategies [15, 231], and the discovery of latent abilities that emerge as models sled [230].

Following the introduction of instruction-based models in LLMs, several state-of-the-art models have epitomized this approach, showcasing remarkable capabilities in handling a wide array of tasks directly based on user instructions. Some notable models in this field include PaLM [33], which represents a significant advancement in scaling up transformer-based architectures designed to perform well across diverse linguistic tasks. Its successor, PaLM-2 [66], builds upon this foundation with improved training techniques and larger model capacities, further enhancing its ability to understand and generate nuanced text based on instructions. PaLM-E [49] further extends the PaLM series by emphasizing efficiency in energy usage and processing speed, making it a more sustainable option for deploying sophisticated NLP tasks at scale.

Another architecture is LLaMA [211] and its successors, LLaMa 2 [62] and LLaMa 3 [132], which focus on achieving high cross-task effectiveness with relatively smaller model sizes, facilitating easier deployment and lower operational costs without compromising on capability. These architectures have demonstrated significant prowess in tasks requiring deep contextual understanding. Mistral 7B [90] and Mixtral 8x7B [91] introduce a unique approach to model training called **Mixture of Experts** (**MoE**) that involves dynamic adjustments of model parameters based on task complexity, which enhances the model's adaptability and performance across different NLP tasks. Gemini 1.5 [205] is another innovative model that integrates dual mechanisms of understanding and generation to improve interaction dynamics in conversational AI applications. Bard [141] focuses on incorporating broad, encyclopedic knowledge and the ability to update its understanding in real-time, making it exceptionally useful for applications that require up-to-date information. Finally, Claude [11] distinguishes itself by its ethical training framework, prioritizing safety and fairness, setting a new standard in responsible AI development.

These models exemplify the most advanced and state-of-the-art instruction-based learning, where each has been tailored to excel in standard benchmarks and improve specific aspects such as versatility, efficiency, and ethical considerations.

### 4.2 Combinatorial Optimization

COPs are a class of optimization problems defined by discrete decision variables and the objective of finding one or more optimal solutions within a finite search space of solutions [97]. Many of these problems are classified as NP-hard, meaning that, based on current knowledge, they require exponential time to be optimally solved. Besides the prominent **Boolean Satisfiability Problem (SAT)**, prototypical COPs include the **Permutation Flowshop Scheduling Problem (PFSP)** [165], the **Knapsack Problem (KP)**, the **Graph Coloring Problem (GCP)**, and the **Traveling Salesperson Problem (TSP)** [174]. CO is also widely applied to tackle real-world problems across various domains. Examples include employee scheduling [101, 248], machine scheduling [107, 151], educational timetabling [26], and automotive production [233], to name a few.

Figure 1 outlines the steps practitioners and researchers undertake to address an optimization problem [14, 213]. In the following, we detail the process specifically for the case of CO.

The first step regards the description in **Natural Language (NL)** of the decision-making process or real-world issue to be tackled. Its specifications must be outlined to identify the decisions to be made, the rules that must be followed, and the goals to be achieved.

After framing the decision-making process, the practitioner must create a formal representation to enable a software solver or a custom algorithm to tackle it. A given NL description can correspond to multiple representations, called *models* or *problem formulations*. Models are articulated in terms

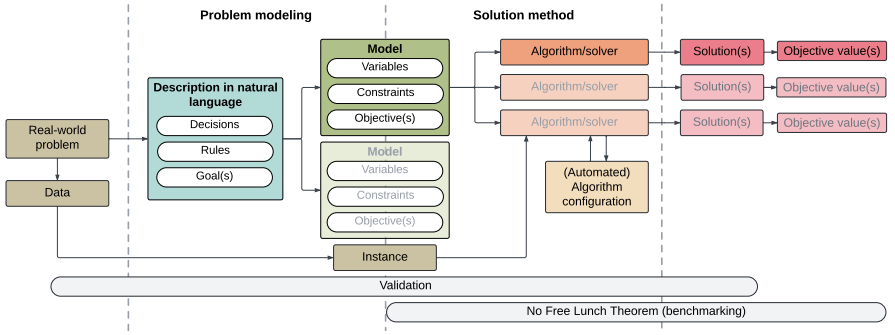


Fig. 1. Overview of the steps for addressing an optimization problem.

of *(i) variables* (or decision variables), which represent the decisions to be made; *(ii) constraints*, which are restrictions on the possible values of the variables and capture problem substructures and/or business rules; and *(iii)* at least one *objective* (or fitness) *function*, which assesses the quality of a solution. When the problem tackles more than one objective function, it is the case of multi-objective optimization. The development of an effective problem formulation is known as *modeling*. How we define variables, objectives, and constraints determines the model type. E.g., if variables assume integer values and constraints are linear, the model is an ILP. Other model types, such as MILP and LP, handle both discrete and continuous variables, while structured constraints can be represented using CP.

While a problem formulation provides an abstract formal description of a decision process, an *instance* (or problem instance) represents a specific case of the problem defined by concrete data. A *solution* to a problem instance is an assignment of values to all the decision variables. As mentioned before, the solution quality is evaluated through an objective function, which provides one or more objective values, also referred to as scores or costs (typically numerical values). The search space of an instance encompasses all possible assignments of the variables; the subset of solutions that satisfy all constraints is called the set of *feasible* solutions. A solution which violates at least one constraint is called *infeasible*. When no feasible solution exists for an instance, resulting in a logically inconsistent instantiation of the problem model, the model instance is also called infeasible.

The search space is explored using *solution methods*, which are algorithms or software solvers. Solution methods for COPs can be classified based on the completeness of their search (i.e., complete or incomplete methods) and how solutions are constructed (i.e., perturbative or constructive methods). Note that models and solution methods are strictly coupled together.

Complete methods exhaustively explore the search space, ensuring that *(i)* If a solution exists, the complete method will eventually find it; *(ii)* When the search terminates, the best solution found is guaranteed to be optimal (for this reason, complete methods are also called exact methods). Examples of these methods include LP, MILP, and CP [186]. For such methods, numerous widely accepted software tools exist, including IBM ILOG CPLEX and IBM CP Optimizer [88], Gurobi Optimizer [70], OR-Tools [169], Gecode [60], and the solvers included in the MiniZinc distribution [156] together with interfaces and APIs available in common programming languages.

However, given that most COPs are NP-hard [59, 106], exhaustive exploration can result in an exponential increase in runtime w.r.t. the size of the problem instance. Additionally, in many real-world applications, it is not necessary to certify the optimality of a solution. Instead, obtaining a good, feasible solution in a reasonable amount of time is sufficient. Incomplete methods address such necessities by exploring the search space non-exhaustively, often using stochastic approaches. When these methods are tailored to the problem, they are called heuristics; when they employ

problem-independent strategies, they are referred to as MHs [144]. Examples of MHs include **Tabu Search (TS)** [65], **Simulated Annealing (SA)** [55, 100], **Greedy Randomized Adaptive Search Procedure (GRASP)** [108], **Large Neighborhood Search (LNS)** [191] and its adaptive version [232], **Genetic Algorithm (GA)** [76], **Biased Random-key Genetic Algorithm (BRKGA)** [133], and **Ant Colony Optimization (ACO)** [48]. Differently from complete methods, widely accepted tools have not been available for incomplete methods, where researchers still rely on customized coding solutions [202–204] and on a sprout of possible frameworks and libraries [135, 168].

Constructive methods build solutions from scratch and iteratively set all variables according to specific policies. Examples include CP and ACO. Conversely, perturbative methods start from an existing complete solution and generate new solutions by modifying some variables. Examples include methods based on the concept of neighborhood, such as TS and SA.

Algorithmic choices in solvers and algorithms can be addressed through *automated algorithm configuration*, as advocated by the **Programming by Optimization (PbO)** manifesto [77]. Such a paradigm calls for a shift of perspective in software development, where the design of components is approached through optimization. This involves a systematic exploration of design alternatives to select a configuration for the different components via automated tools (e.g., irace [138]).

The process undergoes various types of *validation*. First, a practitioner must ensure that the models adhere as closely as possible to the real-world scenario, even though some assumptions are necessary to generalize the concepts (i.e., validation of specifications). Secondly, model and technical validation should be applied to check the correctness of the code, ensure it adheres to the model, and evaluate its performance efficiency. Furthermore, to efficiently solve a problem, evaluating and comparing different algorithms, solvers, and models is essential. This comparison is necessary because of the *No Free Lunch Theorem* [234], which states that no single algorithm is best suited for all instances of an optimization problem. Therefore, rigorous tests are conducted to determine the most effective approach for the specific problem, ensuring that the chosen solution method is robust and efficient (in other fields, this type of validation is addressed as a *benchmarking* process).

## 5 Methodology

### 5.1 Prisma

The PRISMA guidelines are a well-known framework for reporting systematic reviews. The current version was proposed in 2020 [163] and builds upon the PRISMA 2009 formulation [150]. This methodology evolved from the earlier **Quality of Reporting of Meta-analyses (QUOROM)** guidelines [149], which were firstly introduced in 1999. This evolution over time has broadened the scope of the guidelines. While QUOROM primarily focused on improving reports of meta-analyses in clinical trials, PRISMA addresses systematic reviews that evaluate the effects of health interventions more broadly. These guidelines are sufficiently general to apply to reports of systematic reviews that evaluate other types of interventions [121, 235], systematic reviews with objectives beyond evaluating interventions [75, 185], and systematic reviews not related with the medical field [155, 194, 216]. PRISMA can be used to report systematic reviews that involve result synthesis, such as meta-analyses and other statistical methods. It is also helpful for reviews identifying only a single eligible study and for mixed-methods approaches.

At its core, PRISMA is composed of four main elements: the statement [163], the explanation and elaboration document [164], the checklist, and the flow diagram.[1] The statement introduces the purpose of the methodology. The checklist consists of 27 items across seven sections, providing

[1] https://www.prisma-statement.org/prisma-2020-checklist and https://www.prisma-statement.org/prisma-2020-flow-diagram (accessed on 2026-03-08).

guidelines for writing a systematic review report. It should be used alongside the explanation and elaboration document, which offers additional reporting guidance for each item. The diagram shows the flow of information through the review phases.

In this work, we adopt, use, rely on, and refer to the PRISMA 2020 guidelines [163, 164].

### 5.2 Terminology

Most of the terminology used within PRISMA resources [163, 164] derives from its original field of application—systematic reviews of health-related interventions—which may be confusing for researchers from other disciplines. We particularly refer to the following definitions [163]: *(i) Study*: an experiment including a defined group of participants and one or more interventions and outcomes. *(ii) Report*: a paper providing information about a particular study. Multiple reports may refer to the same study. *(iii) Record*: the title or abstract (or both) of a report indexed in a database or website. *(iv) Outcome*: a measurement event for participants in a study. *(v) Result*: the combination of a point estimate and precision measurement for an outcome.

Since our systematic review covers a broader scope than health-related interventions, where we expect to find only individual eligible studies rather than multiple studies referring to a given experimental design, we will join the definitions of *study* and *report*, using *study* to refer to papers in general. This approach will also apply to the definitions of *result* and *outcome*. We consider the term *record* to encompass titles and abstracts of papers. For example, the PRISMA flow diagram explicitly distinguishes between records screened and reports sought for retrieval. We will adjust this to refer only to records as we define them.

### 5.3 Literature Collection

*5.3.1 Process.* Our systematic review includes studies up to the end of 2024. The literature collection process, conducted according to PRISMA, involves three main activities: identification, screening, and inclusion (Figure 2). To provide a comprehensive overview of our application of the PRISMA guidelines, we have included the checklists in Appendix A.

While the inclusion activity simply involves reporting the number of studies included in the systematic review, the task of identifying and screening records is more complex. Figure 3 reports a **Business Process Model and Notation (BPMN)** diagram [159] describing how we conducted these activities. Initially, all authors worked together to define the keywords for searching records, establish eligibility criteria for study inclusion/exclusion, and select the databases for record identification. Subsequently, one author sent queries to the selected databases to retrieve records and performed data cleaning to remove duplicates and records without authors. Then, the author screened the remaining records by reviewing titles, abstracts, and publication years in order to apply an initial filter. The resulting list was then augmented through citation tracking [36]. After this phase, three authors independently read the full text of one-third of the studies each. They then decided whether to include each study based on the eligibility criteria. Subsequently, we collectively cross-checked the studies read by the other authors. The author not initially involved in full-text reading performed the final screening and resolved conflicts.

*5.3.2 Identification.* On January 7, 2025, we searched two popular databases to identify records: Scopus and Google Scholar. Scopus is an extensive, multidisciplinary database of peer-reviewed literature, while Google Scholar allows for broader searches, including pre-prints and other types of studies.

To identify studies addressing the usage of LLMs within CO, we used the following set of keywords: “large language models”, “generative artificial intelligence”, “GPT”, “optimization”, “combinatorial optimization”, “mathematical formulations”, “metaheuristics”, “constraint programming”,

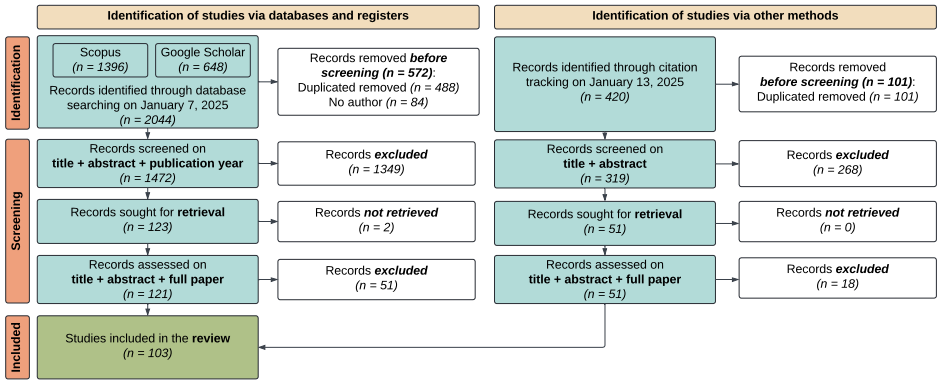


Fig. 2. Literature identification, screening, and inclusion activities conducted following the PRISMA guidelines.

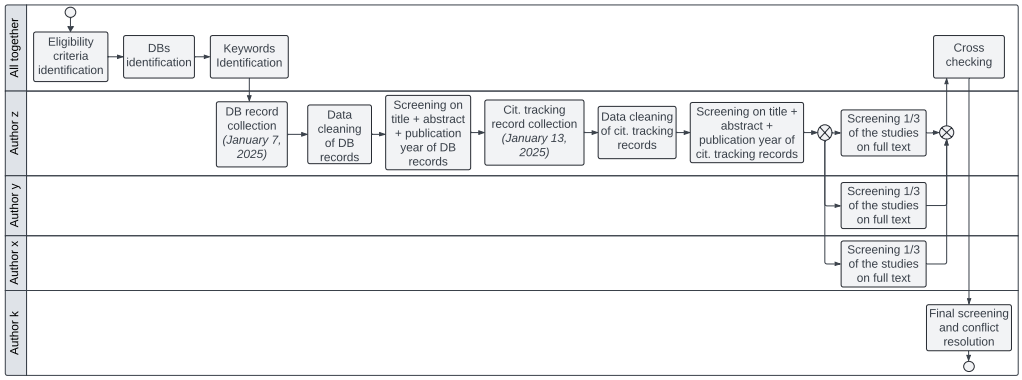


Fig. 3. BPMN diagram describing the identification and selection activities of records in the literature collection process.

"integer programming", "integer linear programming", and "NL4Opt", "Ner4Opt" . These keywords were combined into 14 search queries, reported in Table 1 along with the number of matches found for each query. Besides the self-explanatory queries (i.e., those including the relevant LLM and CO terms like "combinatorial optimization"), we also include the keywords "NL4Opt" and "Ner4Opt" in our Google Scholar search. These terms relate to a recent competition focused on extracting the meaning and formulation of an optimization problem from its textual description. Note that Google Scholar only provides an estimated number of matches and that we used the advanced query functionality on Scopus to restrict the search to study titles, abstracts, and keywords using the `TITLE-ABS-KEY(...)` construct. Using these queries, we retrieved approximately 648 records from Google Scholar and 1,396 from Scopus, for a total of 2,044 records. This list required further processing because both databases contain duplicates. We found 269 duplicate records from Google Scholar, 196 from Scopus, and 23 shared records, totaling to 488 duplicates. Another case to consider is a small subset of records that lack an author string and should, therefore, be removed from the list of 2,044 records. Specifically, we identified 4 records without an author string on Google Scholar and 84 records on Scopus. When combining the two lists, the number of distinct records without an author string is 84, as all four records found on Google Scholar are also present on Scopus. The final number of distinct records from Google Scholar and Scopus is 1, 472.

Table 1. Queries Used to Retrieve Records from Scopus and Google Scholar

| Scopus | | Google Scholar | |
|---|---|---|---|
| Query | Count | Query | Count |
| "large language models" and "optimization" | 944 | "large language models" and "combinatorial optimization" | 334 |
| "GPT" and "optimization" | 394 | "large language models" and "constraint programming" | 104 |
| "large language models" and "constraint programming" | 10 | "large language models" and "mathematical formulations" | 106 |
| "large language models" and "integer programming" | 14 | "large language models" and "metaheuristics" | 88 |
| "large language models" and "integer linear programming" | 15 | "NL4Opt" | 11 |
| "generative artificial intelligence" and "combinatorial optimization" | 0 | "Ner4Opt" | 5 |
| "large language models" and "combinatorial optimization" | 14 | | |
| "GPT" and "combinatorial optimization" | 5 | | |

*5.3.3 Screening.* We define eight eligibility criteria to screen the list of studies referred by the 1,472 records retrieved. Specifically, we define four inclusion (INCL) criteria and derive four exclusion (EXCL) criteria:

INCL1 The study is written in English.
INCL2 The study focuses primarily on the usage of LLMs in the field of CO.
INCL3 The study has been published in 2016 or later.
INCL4 The study is in the form of a research article, a case report, a technical note, a narrative review, a systematic review, a position paper, a pictorial essay, or a PhD Dissertation.

EXCL1 The study is written in languages other than English.
EXCL2 The study focuses primarily on optimization in the context of LLMs, solely on CO, solely on LLMs, or solely on optimization different from CO, such as Continuous Optimization.
EXCL3 The study has been published in 2015 or earlier.
EXCL4 The study is not in the form of a research article, a case report, a technical report, a narrative review, a systematic review, a position paper, a pictorial essay, or a PhD Dissertation.

In summary, we focus on how LLMs are used in optimization, specifically targeting studies that address the subfield of CO (INCL2); consequently, we exclude those taking the opposite perspective (EXCL2). Note that some of the exclusion criteria may appear redundant w.r.t. the inclusion ones; this is due to a rigorous application of the PRISMA guidelines in reporting on our activity of review.

The formulation of these criteria is based on the concept of "primary focus" on the use of LLMs in the field of CO. According to our criteria, a study focusing on the use of LLMs in COPs should either: *(i)* Describe the usage of LLMs within one of the tasks of the overall optimization process. *(ii)* Address specific aspects of a COP. *(iii)* Propose improvements or new approaches for a given optimization paradigm. *(iv)* Outline use cases, perspectives, and/or potential future applications. *(v)* Involve specific LLMs and, if applicable, provide metrics and/or reference datasets. We believe that this notion of "primary focus" allows us to consider not only research papers but also reviews and surveys, which are equally important as they may provide general directions and perspectives (INCL2 and INCL4). This also helps clarify the rationale behind EXCL2 and EXCL4; if any of the components described above is missing, we exclude the study from consideration. For example, let us hypothesize a study on how an LLM could enhance an evolutionary procedure for solving a COP; this study would align with our inclusion criteria. Conversely, consider a study proposing the use of a MH to generate prompts for LLMs; such a study would not meet our criteria and would be excluded. Concerning the criteria for the publication year (INCL3/EXCL3), we selected 2016 as the earliest publication year to ensure comprehensive coverage for our systematic review. This

choice aligns with key developments in LLMs, particularly following the "Attention Is All You Need" paper [221], which introduced the Transformer architecture in 2017 (Section 4.1)

Regarding the types of publications considered, we have identified a subset that aligns with our objectives (INCL4). We therefore exclude all other publication types not specified therein (EXCL4).

We screened the identified records (1,472) according to our eligibility criteria. We removed 68 records because they were published earlier than 2016 (INCL3/EXCL3). A total of 1227 records tackled solely LLMs, solely optimization, or a type of optimization different from CO. Additionally, we excluded 47 records that addressed optimization in the context of LLMs (INCL2/EXCL2). We also removed 7 records because they referred to studies of the wrong type (EXCL4). In total, we removed 1,349 records, resulting in a final count of 123 records.

On January 13, 2025, we performed citation tracking [36] on the 123 records that remained after the initial screening. Specifically, we traced all the citations received by the collected studies referred to in the records, identifying a total of 420 records. We then checked for duplicates in this set, identifying 101 duplicates. Thus, we ended up with 319 distinct records from citation tracking.

We manually addressed each record, evaluating them according to our criteria as we did previously. Among the 319 distinct records, we found that 251 tackled solely LLMs, solely optimization, or a type of optimization different from CO (EXCL2). Additionally, 11 records referred to studies that tackled optimization in the context of LLMs. We excluded 5 records because they referred to studies of the wrong type (EXCL4) and one study because it was not written in English (EXCL1). In total, we removed a total of 268 records, leaving us with 51 records. By examining these citations, we uncovered additional highly related or complementary sources w.r.t. the initial set of records, thereby broadening and enriching our literature review.

In summary, combining the records retrieved through databases (123) and those obtained through citation tracking (51), we had a total of 174 records.

*5.3.4 Inclusion.* We read the full text of each of the 174 studies referred to by the records obtained after the screening activity to decide which ones to include in our systematic review. This involved a more thorough evaluation of each study concerning our eligibility criteria. Below are a few examples:

- Amarasinghe et al. [8] proposed to automate the formulation of optimization problems from NL descriptions, using fine-tuned LLMs to generate modular code for complex real-world business optimization scenarios. In this study, LLMs are used to enhance the formulation of a COP, thereby satisfying all our inclusion criteria.
- Pluhacek et al. [171] used an LLM to identify and decompose six well-performing swarm algorithms for continuous optimization. Despite the involvement of LLMs, we exclude this study because it focuses exclusively on continuous optimization problems (EXCL2).
- Wang et al. [228] proposed a survey on the usage of **Deep Generative Models (DGMs)** for COPs. We exclude this study since DGMs are not LLMs (EXCL2).

Thus, the final number of studies included in this systematic review is 103 (i.e., 2 studies were not retrieved and 69 were not included as per inclusion/exclusion criteria, thus 174 = 103 + 2 + 69).

Table 2 summarizes the characteristics of the included studies. Please note that when feasible, the characteristics reported are those of the peer-reviewed version of the studies, i.e., if a study was previously published in a pre-print form and later in a peer-reviewed version, we consider the latter here. The majority (82 studies, 80%) were published in 2024. The remaining studies were published in 2023 (17 studies, 16%) and in 2022 (4 studies, 4%). Considering their identification approach, 70 (68%) were retrieved from databases; additionally, we included 33 (32%) studies referred to by records found through citation tracking. Regarding the publication venues, approximately 50 studies (49%) were disseminated through pre-print distribution services such as arXiv. The remaining studies were

Table 2. Overview of the 103 Studies Included in Our Review

| Year | Total | Identification | | Venue | | | Publication Type | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | DB | Citation Tracking | Journal | Conference | Pre-print | Research | Review | Report | Position |
| 2022 | 4 | 3 | 1 | 0 | 1 | 3 | 4 | 0 | 0 | 0 |
| 2023 | 17 | 16 | 1 | 3 | 4 | 10 | 15 | 0 | 1 | 1 |
| 2024 | 82 | 51 | 31 | 22 | 23 | 37 | 65 | 11 | 0 | 6 |
| **Total** | 103 | 70 | 33 | 25 | 28 | 50 | 84 | 11 | 1 | 7 |

peer-reviewed, with 25 (24%) published in journals and 28 (27%) in conference proceedings. Focusing on the type of each study, as specified in our inclusion criteria (INCL4), 84 of them (approximately 82%) are research studies addressing specific research questions and/or experimental settings. Additionally, we included 11 (10%) literature reviews, one (1%) technical report, and 7 (7%) position papers, as these provide perspectives, guidelines, or future directions.

## 6 Analysis

In this section, we classify and discuss the 103 studies. The analysis is conducted in terms of optimization process, LLMs, benchmark datasets, and application domain. Each selected study is described in Appendix B.

### 6.1 Optimization Process

In this section, we report the analysis of the included studies w.r.t. the optimization process, both considering the general task (e.g., problem modeling) and the related activities (e.g., domain knowledge). A detailed tabular overview is available in Appendix C.

This analysis accounts for 86 out of the 103 studies (83%). The omitted studies provide perspective and general direction on the topic (Section 6.5). In the counts related to the number of studies per step or task, a study addressing multiple tasks within the same step is counted only once for that step. For instance, Li et al. [116] addresses both entity recognition and model creation within the problem modeling step. Therefore, it is counted as one in the problem modeling step. If a study addresses multiple steps, it is counted as 1 in each step. For instance, Tsouros et al. [213] addresses activities both in the problem modeling and solution method steps, thus it is counted as one in each of them. Most of the studies (64 out of 103) focus on activities related to the solution method, representing 63% of the total. The problem modeling task follows closely, with 38 studies, accounting for 37%. A limited number of studies are related to benchmarking (7, 7%) and validation (9, 9%). In the following sections, we outline how LLMs are used within each step of the process.

The optimization process could benefit from the application of LLMs in several ways, as advocated by Wasserkrug et al. [229], who argues for their integration throughout the entire process. While such a holistic approach is possible, only 24 studies (23%) address multiple steps across the pipeline.

*6.1.1 Problem Modeling.* Problem modeling aims to translate problems, expressed in NL, into mathematical and computational models. This step is foundational, as the quality of the models influence the success of optimization.

A specialized understanding of the domain in which the COP is located is crucial for effective modeling and for ensuring realistic and applicable solutions. Traditionally, human experts are necessary in this phase as they bring deep knowledge of industry-specific rules, regulations, and so on. In such context, LLM can help by extracting implicit knowledge from the real-world data and

integrating it into solution processes [40, 131, 188, 254] and to generate solutions [37, 39, 95]. Li et al. [119] integrates the domain knowledge, with both entity recognition and the generation of solution code.

In the given problem context, it is crucial to identify the key entities and their interrelations to properly model variables, constraints, and objectives. LLMs can be particularly useful for recognizing and labeling these elements within a NL description of the problem [47, 225]. This capability can enhance the accessibility and usability of the models, enabling non-domain experts to solve significant problems across various industries. Such activity is frequently paired with the extraction of domain knowledge [6, 215], the creation of the model [4, 45, 74, 89, 116, 157, 158, 179], and toward code generation [87, 119]. Note that Obata et al. [158] also addresses solution generation. As mentioned, LLMs can produce optimization models [58, 181] and assist users in doing so, as evidenced, for instance, by the conversational agents developed by Abdullin et al. [1]. The generation of an optimization model is often accompanied by the production of the related code [82, 94, 111, 239]. Several studies have jointly addressed entity recognition, model creation, and code generation [2, 3, 71, 92, 96, 153, 213, 241, 249], with Michailidis et al. [146] also addressing solution generation.

A total of 38 studies focused on problem modeling, encompassing the identification of domain knowledge (11 studies), entity recognition (25 studies), and model creation (25 studies).

*6.1.2 Solution Methods.* Solution methods refer to strategies and algorithms used to find (near) optimal solutions to COPs. The choice of the solution method is linked to the problem model and is deemed critical, as it determines the efficiency, scalability, and accuracy of the results. Thus, optimization researchers' expertise is essential to select the right solving strategy and to design algorithms and their components. LLMs have been adopted in the design of solution methods, especially through code generation; there exists one case where LLMs have been used for algorithm selection [238]. Complete algorithms have been designed starting from NL description of solution methods for CP [8, 223] and MILP [7, 114, 118]. Liu et al. [127] and van Stein et al. [219] used LLMs to generate heuristic algorithms within an evolutionary framework, with the latter also including in the loop parameter tuning using an automated tool (SMAC [122]). On the contrary, a few studies implemented a LLM-based parameter tuning [63, 110, 143, 217]. Ye et al. [244] and Liu et al. [125] developed heuristics for a hyper-heuristic framework. Romera-Paredes et al. [184] introduced FunSearch, a method for discovering new heuristics for the online bin packing problem, achieving improvements over widely used baselines. Following, many other studies tackled the topic of generating code for heuristics [30, 127, 129, 200, 242, 243, 246, 251]. Mao et al. [142] integrated LLMs into mutation and crossover operators for GAs. Most of the generated code is written in Python (30, note that overall 35 studies use Python).

Generating solutions that satisfy the constraints and are diverse enough is complex and sometimes even unfeasible [222]. LLMs have been asked to directly produce solutions [19, 32, 52, 63, 69, 80, 81, 86, 93, 115, 124, 130, 148, 154, 182, 183, 187, 212, 227, 240, 245]. Such an approach is useful, for instance, in MHs, where complete solutions are required as a starting point or to be mutated during the evolutionary process, and in population-based optimization algorithms, where a large number of (possibly diverse) solutions are required at each iteration. Finally, code generation and solution generation have been tackled together by two works [20, 98].

A total of 64 studies focused on solution methods, encompassing code generation (36 studies), solution generation (28 studies), parameter tuning (4), and algorithm selection (1).

*6.1.3 Validation.* The validation step ensures that the model, the code, and the produced solutions align with the problem requirements and meet end-users' needs. This process often requires substantial human input, as domain experts are typically involved in an iterative feedback loop to critically review the models and solutions to verify that all specifications have been adequately

addressed. As LLMs have been proven to perform fairly well in such tasks [189], they have also been applied to optimization. Huang et al. [87] employed an LLM to validate solutions in an end-to-end framework (spanning from problem modeling to technical validation). Conversely, Hao Chen and Li [72] acted at the model level, employing a LLM agent to validate the (MI)LP models. Specifically, they integrated LLMs in diagnosing inconsistent (i.e., over-constrained) MILP optimization models, trying to isolate the minimal subset of linear constraints (including variable bounds) that makes the model instance infeasible. Validation has also been addressed by other studies, reported also in other steps of the optimization process [52, 71, 115, 153, 239, 249]. Note that we do not address bug checks in LLM-generated code as proper technical validation (e.g., during code generation for the solution method step) since we assume such checks are inherently part of that activity (if not state otherwise). A total of nine study focused on validation (five on solution validation and six on model validation).

*6.1.4 Models and Algorithm Types.* Since the optimization process can be clearly outlined in terms of tasks and steps, it is crucial to recognize that modeling choices, solution methods, and validation procedures are closely interconnected. This section presents an overview of the algorithms and solvers considered in the studies. Please refer to the sections related to the single steps.

Considering the underlying model and algorithm type, the majority of studies (19) focus on LP and MILP (18). Other exact methods involve CP (7) and ILP (2). (Meta-)Heuristic algorithms account for 20 studies; among them, 9 focus on heuristics, 3 focuses on hyper-heuristics, 3 on GAs, and 3 on EAs. Furthermore, 3 study explores the capabilities of LLM w.r.t. multi-objective CO [124]. One study deals with SAT—even though LLMs are used to generate heuristics for their solution [200].

One reason for the prominence of LP in these studies is the **Natural Language for Optimization Modeling (NL4Opt)** competition. The primary objective of this competition was to assess whether models could generalize to unseen problem domains, with an emphasis on two sub-tasks: named entity recognition for LP components and LP model generation. For more detailed information on the competition, readers are forwarded to the report by Ramamonjison et al. [180], while a description of the related dataset is provided in Section 6.3.

*6.1.5 Benchmarking.* In CO, benchmarking involves evaluating and comparing the performance of algorithms and techniques. The objective is to determine how effective a solution method is and, potentially, understand the reasons behind its performance. Conventional comparison methods involve gathering numerical data from algorithm runs and reporting them in the form of tables and boxplots, alongside considerations of computational times and statistical tests (such as the Friedman test) to account for stochasticity. For decades, researchers have emphasized the need for additional tools to better understand the behavior of optimization algorithms [18]. One such tool is related to **Search Trajectory Networks (STNs)**, a graph-based tool for the visualization of MH behavior [160]. While these visualizations are very informative, they require prior knowledge to be produced (e.g., parameters for search space partitioning) and interpreted (e.g., which algorithm is superior). To bridge this gap, Chacón Sartori et al. [28] enriched STNs with LLM-generated explanations and suggestions.

The visual investigation of solutions is enabled by Da et al. [37], who also allows solution validation. The visual capabilities of LLM are also explored by Elhenawy et al. [52], who directly employ them to solve problems, presenting instances in visual form.

An increasingly relevant topic in CO is explainability of results [203] as a way for enhancing confidence and trust in the solutions. Interesting questions include identifying which solution components most significantly influence the final result and understanding the characteristics of high-quality solutions, among others. Providing informative answers to these questions requires in-depth knowledge of the application domain, the solution method, and the specific instance being addressed. To reduce this human-intensive task, Kikuta et al. [99] proposed a post hoc

LLM-based explanation framework aimed at clarifying the decision-making process of **Vehicle Routing Problems (VRPs)**. Similarly, also other study included an explainability level [80, 183].

A total of 7 studies focused on benchmarking (3 on visual analysis and 4 on explainability).

### 6.2 Large Language Models

The LLMs available today exhibit different capabilities: some are designed for processing instruction-like prompts, others specialize in generating programming code, and so on. Understanding the role of each LLM in the reviewed studies is valuable for developing future approaches to solving COPs, complementing the findings in Section 6.1.

The 70 LLMs used in the 103 studies are based on 22 different architectures. Despite the diverse capabilities of the architectures employed, most LLMs have been used in CO to enhance problem modeling and solution methods. In the following, we provide a description of how LLMs are used within the included studies, considering their architectures, general tasks, and related activities. Note that the numbers for activities may not sum to those at the main task level, as a given LLM can be used for multiple purposes. For a tabular overview of the LLMs, we refer the interested reader to Appendix D.

*6.2.1 LLM Architectures.* 13 out of the 22 architectures focus on generative approaches. Introduced in 2022, `GPT-3.5` [21] is particularly effective for tasks requiring fluent text generation. `GLM` [51] emerged as a balanced approach to generative tasks, leveraging both bidirectional and autoregressive capabilities. During 2023, several architectures were released: `LLaMa 2` [62], optimized for efficiency and suitable for tasks involving scalability and quick inference times; `PaLM 2` [66], which performs strongly in scenarios requiring comprehension and contextual text generation; `Mistral` [90], designed for efficient inference and competitive performance in text generation, making it suitable for resource-constrained deployments; and `Qwen` [175], focusing on general-purpose text generation with multilingual support and fine-tuned reasoning capabilities. In the same year, `GPT-4` [161] appeared as a particularly strong model for fluent text generation, alongside `LLaMa 3` [132], which is optimized for instruction-following and structured text understanding. `DeepSeek` [41] also adopts generative approaches with a particular focus on the Chinese language, while `Claude 3.5` [10] excels in dialogue-based applications and knowledge-intensive tasks. `Qwen2` [176] extends its predecessor with multilingual capabilities and refined reasoning. `Cohere`'s `Command-R+` [5] is optimized for **Retrieval-Augmented Generation (RAG)**, improving factual accuracy and knowledge recall. Finally, `Yi` [209] is tailored for diverse text generation tasks, integrating efficient training strategies for improved performance. A total of 62 studies rely on these architectures, namely 24 on `GPT-3.5` (2022), 2 on `GLM` (2022), 9 on `LLaMa 2` (2023), 3 on `PaLM 2` (2023), 2 on `Mistral` (2023), 3 on `Qwen` (2023), 43 on `GPT-4` (2023), 7 on `LLaMa 3` (2024), 6 on `DeepSeek` (2024), 7 on `Claude 3.5` (2024), 1 on `Qwen2` (2024), 1 on `Cohere` (2024), and 1 on `Yi` (2024).

A total of 5 architectures are designed to handle textual input. `T5` [178] (2019) treats all text-based tasks as text-to-text problems, offering significant flexibility across a range of natural language processing applications. `BART` [113] (2019) is particularly effective at transforming textual input into structured outputs, making it suitable for complex textual tasks. `UnixCoder` [68] (2022) is an architecture specialized in programming-related tasks, such as code generation, code summarization, and code translation. Additionally, `LLaMa 2` [62] (2023) and its successor, `LLaMa 3` [132] (2024) provide LLMs optimized for instruction-following (e.g., `LLaMa 3-70B-Instruct`) and are fine-tuned for tasks requiring structured text understanding. A total of 9 studies rely on these architectures: 3 on `T5` (2019), 3 on `BART` (2019), 1 on `UnixCoder` (2022), 1 on `LLaMa 2` (2023), and 1 on `LLaMa 3` (2024).

Additionally, 4 architectures focus on multimodal data. All these architectures (or their multimodal models) were introduced in 2024. `Gemini` emphasizes both multimodal capabilities and

generative approaches, making it ideal for tasks that require integrating and generating text, images, and possibly other forms of data. `GPT-4-Vision-Preview` extends the `GPT-4` [161] architecture by incorporating visual input processing, allowing it to handle tasks involving image understanding and text-image reasoning. `Mixtral` [91] leverages an ensemble of models designed to handle multimodal tasks, such as text-to-image and image-to-text conversions. `OpenCoder` [83] is optimized for code-related tasks and supports multimodal inputs, particularly focusing on enhancing software development workflows with a combination of textual and structural data processing. A total of 9 studies rely on these architectures: 5 on `Gemini`, 2 on `Mixtral`, 1 on `GPT-4`, and 1 on `OpenCoder`.

At the time of writing, several LLMs analyzed in this study have been deprecated or replaced by newer versions. OpenAI's `Text-Davinci-003` and `Text-Davinci-Edit-001`, based on `GPT-3.5`, have been phased out in favor of `GPT-4-Turbo` and `GPT-4o`. Google rebranded `Bard` as `Gemini`, transitioning from `PaLM 2` to the `Gemini 1.x` and `2.x` series. Meta's `LLaMa 2-13B` has been largely replaced by `LLaMa 3`, while OpenAI's `GPT-4` evolved into `GPT-4o`, incorporating multimodal capabilities. Similarly, `Mixtral-8x7B-Instruct-v0.1`, `Qwen`, and `DeepSeek` have advanced to `Mixtral`, `Qwen2`, and `DeepSeek-V2`, respectively. Anthropic's `Claude 3.5` is also expected to be succeeded by newer iterations. Given the rapid evolution of LLMs, readers should consider the latest available models when interpreting our findings.

Given the varying performance of LLMs across architectures, access to their source code remains a key issue to be analyzed. Most high-performing models, including OpenAI's `GPT-3.5` and `GPT-4`, Google's `PaLM 2` and `Gemini`, as well as Anthropic's `Claude 3.5` and Cohere's `Command-R+`, are closed source and require paid access. `Mixtral`, while open-weight, is commercialized via paid APIs, and `DeepSeek V2` adopts a more restrictive licensing model. In contrast, Meta's `LLaMa 2` and `LLaMa 3` are open source under specific licenses, while Mistral AI offers both open-weight and commercial models. `Qwen` and `Qwen2` provide open-source variants alongside proprietary versions. Fully open-source models include `BART`, `T5`, and certain versions of `LLaMa`, `Mistral`, `StarCoder`, and `OpenCoder` (the latter two specialized for code generation). Given the evolving nature of access policies and licensing conditions, these constraints may influence model selection in research.

We count separately both specific implementations designed for conversational purposes and versions of the same model updated to a specific timestamp, such as `GPT-4-0613` (referring to a version of `GPT-4` released or fine-tuned on June 13, 2023). This approach aims to enhance clarity and improve reproducibility. However, there is no guarantee of consistent responses even when using different versions of the same models [56, 103], especially since chat completions are not deterministic by default.[2]

When using LLMs in a research setting, several key aspects must be addressed to ensure both reliable and reproducible results. Documenting the exact prompts used is crucial, as LLMs are highly sensitive to prompt variations, which can lead to significantly different outcomes [9, 16]. Additionally, detailed reporting on the model version, configuration, and any fine-tuning is essential for accuracy and replicability. Notably, a limited number of studies (7) that conducted experiments in this area did not provide any details about the LLM employed [45, 131, 146, 154, 158, 181, 217]. Understanding the training data used in an LLM can help identify potential data leakage, biases, or knowledge gaps that might affect results [13]. Moreover, documenting any additional steps, such as data preprocessing or post-processing of model outputs, is critical for ensuring that findings can be replicated across different studies and datasets. By addressing these factors, research involving LLMs remains transparent, reproducible, and robust.

Finally, when evaluating LLMs in the context of COPs, no universally shared metrics emerge. Accuracy is the most commonly used metric, but its interpretation varies by task. Researchers

[2] https://developers.openai.com/api/docs/guides/advanced-usage/ (accessed on 2026-03-08).

sometimes define a problem-specific "score", while other metrics remain domain-specific. Indeed, many studies evaluate LLMs based on the objective values of the solutions they produce. As discussed in Section 4.2, this evaluation is problem-specific, as each problem has distinct objective functions. For instance, in TSP, the goal is to minimize travel distance, whereas in PFSP, it is typically to minimize tardiness or completion time. Since the optimal solution may not always be available, evaluations often compare the obtained solutions to the best-known ones. A common metric for this comparison is the optimality gap (or relative deviation), calculated as: $\text{gap} = 100\times(Z_{\text{llm}} - Z_{\text{best}})/Z_{\text{best}}$ where $Z_{\text{llm}}$ represents the objective value of the LLM solution, and $Z_{\text{best}}$ denotes the best-known objective value.

*6.2.2 Problem Modeling.* A total of 40 LLMs out of 70 (57%) have been used for problem modeling. Among them, 19 LLMs have focused on enhancing *model creation*. A common approach involves creating models from unstructured NL. Some of these approaches address optimization problems more broadly, utilizing LLMs such as `GPT-3.5-Turbo` (an optimized variant of `GPT-3.5` for faster performance and lower cost [239]), `GPT-3.5-turbo-0613` (a June 2023 release of `GPT-3.5-Turbo`), `GPT-4-0613` (a June 2023 release of `GPT-4`), and `GPT-4` itself [94], as well as `LLaMa 2-7b`, a version of `LLaMa 2` with 7 billion parameters [4], and `T5-Base`, the base model built on the `T5` architecture [74]. Additional optimized or specialized variants, including `GPT-4o`, an improved multimodal version of `GPT-4` [2], `Qwen1.5-14B`, `DeepSeek-V2` [241], `Mistral-7B`, `DeepSeek-Math-7B`, `LLaMa 3-8B`, and `Qwen2.5-7B` [82], have also been used for broad optimization tasks, leveraging improved reasoning capabilities and efficiency. Moreover, instruction-tuned models such as `Code Llama-Instruct` and `Zephyr-7b-beta`, a 7B-parameter model designed for instruction-following, have been used for structured task execution [153]. Furthermore, both `GPT-4o` and `Claude 3.5 Sonnet` have been employed in a zero-shot fashion to generate problem formulations directly from user input, thus reducing the need for extensive prompts or examples [71]. Other approaches focus on specific formulations. For instance, LLMs have been used to automate the generation of LP models with `ChatGPT-3.5` [116], `GPT-4` [1], and the `BART` architecture, including both `BART-Base` [58, 179] and `BART-Large` variants [58, 89]. Furthermore, `ChatGPT-3.5` has also been employed in MILP problems [8], as have `Bard`, a conversational model built on the `PaLM-2` architecture [116], and `LLaMa 3-70B` [96]. Instruction-tuned `Code Llama-Instruct` and `Zephyr-7b-beta` have likewise been applied to **Quadratic Programming (QP)** tasks by translating user directives into valid constraints and objective functions.

Concerning *entity recognition*, 22 LLMs have been used to identify and extract specific elements from model representations. Specifically, `GPT-4` and `Text-Davinci-003`, based on `GPT-3.5` and optimized for a wide range of tasks, have been used to translate user input into constraints that the underlying CP model can process [111]. This approach has also been applied to LP and MILP problems using `GPT-3.5` [3, 249], `GPT-4` [249], `ChatGPT-4` [6], `LLaMa 3-70B` [2, 96, 153], `ChatGPT-4o` [2, 71, 92], and `Claude 3.5 Sonnet` [71]. Meanwhile, `CodeLlama-Instruct` and `Zephyr-7b-beta` have also been used for QP problems. Additionally, `ChatGPT-3.5` and `Bard` have been used exclusively for MILP [116], while `T5-Base` has been employed for LP [74]. Furthermore, several LLMs have been applied to general optimization problems, including `GPT-3.5-Turbo-0163`, `GPT-4`, `LLaMa 2-7b`, `LLaMa 2-13b`, and `LLaMa 3-70B` [4, 37], as well as `GPT-4o-mini` [119], `BART-Base`, `BART-Large` [89, 179], and `Gemini 1.0 Pro` [87].

Additionally, there are 12 applications of LLMs for extracting *domain-specific knowledge*. For instance, `GPT-4` and `GPT-4-0163` have been used to extract general domain knowledge in household financial planning [40, 96]. `GPT-4o`, `Claude 3.5 Opus`, `Command-R+`, and `Mixtral-8x2B` have been applied to a social network problem [188], and `GPT-4o` has also been used in robotics [148]. `LLaMa2-7B`, `LLaMa2-13B`, `GPT-3.5`, and `GPT-4` have been employed to model knowledge for the

VRP problem [37]. Moreover, a single LLM, ChatGPT-4, has been used to model domain knowledge in the form of knowledge graphs [215].

*6.2.3 Solution Methods.* LLMs has been employed in 60 out of 70 (85%) works for what concerns the solution method task. Specifically, 23 LLMs have been used for *solution generation*. One approach involves creating candidate solutions for well-known COPs, such as a specific class of the VRP, by leveraging both GPT-based models like GPT-4 [3], GPT-4-Turbo, and GPT-4o [98], and LLaMa-based or T5-based models, including LLaMa 2 and T5-Base [32]. Another approach uses LLMs to combine problem descriptions and previously generated solutions within a meta-prompt processed by ChatGPT-3.5-Turbo, PaLM 2-L, PaLM 2-L-IT, and text-bison [240]. The multimodal capabilities of GPT-4-Vision-Preview and ChatGPT-4o have also been employed to generate solutions through visual prompts [52, 86]. Moreover, ChatGPT-4 has been adopted for finance-related solutions [183] or automated sequence planning in robot-based assembly [245]. Other LLMs used for domain-specific problems include GPT-4 for travel planning [39], program scheduling [95], and traffic simulation [37]. LLaMa 2-7b and LLaMa 2-13b have also been applied to traffic simulation. Claude 3.5 Sonnet has been used in molecular biology [182], and ChatGPT-4-Turbo together with ChatGPT-4o-mini has been used to coordinate computation in a graph reasoning scenario [80]. GPT-3.5-Turbo has found utility in industry-related problems [239], while GPT-4-Turbo, GPT-4o, Gemini 1.5 (Pro and Flash), and Gemma 2 27B have been employed in planning tasks [19, 148]. Furthermore, in the context of GAs, GPT-3.5-Turbo-0613 has been used to select parent solutions from the current population and perform crossover and mutation [130], as well as to perform mutation and crossover only [142].

Additionally, 35 LLMs leverage approaches for *code generation*. GPT-3.5, GPT-4, GPT-4o, and Qwen (LoRA Fine-Tuned), which is a version of the Qwen family fine-tuned using LoRA [79], have been used to generate code specifically for LP, **Mixed Integer Programming (MIP)**, and MILP approaches to COPs [3, 94, 118, 249], whereas CodeLlama-Instruct and Zephyr-7b-beta have been applied to both MILP and QP [153]. Various methods for automating the generation of heuristic algorithms rely on a range of LLMs, each optimized for different aspects of code generation. For example, CodeLlama, a code-oriented variant of LLaMa 2 [251], and StarCoder, trained on extensive code-related datasets, are fine-tuned for specific programming tasks, while DeepSeek-LLM-7B-Base, GPT-3.5-Turbo, and GPT-4-Turbo are optimized for speed and efficiency [96, 124, 125, 127, 129, 184, 242, 244, 246]. GPT-3.5-Turbo-0613 has likewise been employed for generating crossover and mutation implementations in Python [142] within EA contexts. Multimodal models, such as GPT-4o, and advanced reasoning models like Claude 3.5 Opus and Claude 3.5 Haiku, have also been utilized in heuristic generation tasks, and DeepSeek-Coder along with its updated DeepSeek-Coder-V2 focus on high-performance code synthesis. Similarly, GLM-3-Turbo, OpenCoder-8B-Instruct, and Yi-34b-Chat provide structured, optimized code generation [129, 243]. For large-scale problem-solving, models such as Gemini 1.0 Pro and Codey, both built on PaLM 2, have shown strong performance in complex coding scenarios, while the latest iterations of foundation models (e.g., LLaMa 3-70B, LLaMa3-70B-Instruct, LLaMa 3.1-8B, GPT-3.5-Turbo, GPT-4o-Mini, and Qwen-Turbo) have been refined for specialized programming applications [129, 243, 251]. Other approaches harness a self-reflection mechanism to directly generate executable Python code from natural language, exemplified by GPT-4 and Gemini 1.0 Pro [87]. A comprehensive code-generating framework for business optimization has also been developed using CodeT5-finetuned_CodeRL [112], a model that employs reinforcement learning on top of CodeT5 [8]. Additionally, Text-Davinci-Edit-001 has been employed to generate MiniZinc-specific representations [7], while Text-Davinci-003 and GPT-4 have produced Gurobi-based code for supply chain optimization [114]. GPT-3.5-Turbo, Mistral-7B, DeepSeek-Math-7B, LLaMA 3-8B, and Qwen2.5-7B have been broadly used for industry-related problems [82, 239], and

a specialized version called GPT-4o-2024-08-06 has been introduced to optimize code for SAT solvers [200]. Moreover, GPT-4o and Claude 3.5 Sonnet have addressed supply chain and robot logistics tasks [71], and ChatGPT-4o-mini has been applied to multi-agent solution design [119].

Moving to *parameter tuning*, we identified 5 applications of LLMs. ChatGPT-4o has been adopted to model user preferences by adjusting an optimizer's weights [63], whereas GPT-3.5, GPT-4, Gemini, and Le Chat (the chatbot interface for Mistral models) have been used to set MHs parameters [143].

A single application aimed at enhancing *algorithm selection*: UnixCoder, a code representation model designed for programming tasks, has been used to extract features linked to the underlying optimization algorithms [238].

*6.2.4 Validation.* We found 10 applications of LLMs out of 70 (14%), in the context of validation for optimization models. In particular, 7 LLMs have been involved in *solution validation*. GPT-4 and Gemini 1.0 Pro have been used to validate solutions generated for the VRP [87], while GPT-3.5, LLaMa 2-7b, LLaMa 2-13b, and ChatGPT-4o have been applied to broader validation tasks [37, 52]. Additionally, Qwen (LoRA Fine-Tuned) has been leveraged for solution validation [249].

As for *model validation*, 9 LLMs have played a role. GPT-4 has been employed to diagnose infeasible MILP models through interactive sessions [72], while GPT-3.5-Turbo has been used to validate models built from natural language [239]. Similarly, GPT-3.5, GPT-4, and Qwen (LoRA Fine-Tuned) have also been applied in this context [72, 249], with GPT-4o and Claude 3.5 Sonnet reported for model validation tasks [71]. In addition, code-focused models have been utilized to ensure correctness: CodeLlama-Instruct and Zephyr-7b-beta have been adopted to verify that the generated code remains consistent with the original model formulations [153].

*6.2.5 Benchmarking.* As for benchmarking, 9 out of 70 (14%) LLMs have been used to enhance it. Specifically, 8 LLMs use *visual analysis*, as they are integrated, for instance, with a tool designed to visualize the behavior of various algorithms applied to specific instances of a COP [28]. The models used by Chacón Sartori et al. [28] include Mixtral-8x7B-Instruct-v0.1, which is optimized for instructional and guided tasks, GPT-4-Turbo, and Tulu-v2-dpo-7b, a fine-tuned version of LLaMa 2 that was trained on a mix of publicly available, synthetic, and human-generated datasets using a parametrization of the RLHF algorithm known as Direct Preference Optimization [177]. Furthermore, GPT-3.5, GPT-4, LLaMa 2-7b, and LLaMa 2-13b have been employed to visualize solutions for a domain-specific problem [37], while ChatGPT-4o has been used to interpret visual inputs in lieu of purely numerical data [52].

We also identified 5 applications aimed at improving *explainability*. ChatGPT-4 has been employed to describe the decision-making process for the VRP [99], while ChatGPT-4-Turbo, ChatGPT-4o, ChatGPT-4o-mini, and Claude 3.5 Sonnet have been used, for instances, of both the TSP and mTSP [37, 63, 80, 182].

*6.2.6 Platforms for Supporting LLMs-Based Approaches.* While reviewing the studies considered for inclusion, we came across two platforms used by Chacón Sartori et al. [28] to support the design of their approach. Although these platforms are general-purpose and facilitate the use of LLMs broadly rather than specifically in the context of CO, we believe it is beneficial to describe them briefly.

Chatbot Arena [31] is an open platform for evaluating LLMs based on human preferences. It offers access to over twenty LLMs, including both proprietary and open-source options, and provides a leaderboard to compare results. Chat2Vis [139], on the other hand, focuses on generating visualizations directly from natural language text. It uses various LLMs and shows that a set of

proposed prompts offers a reliable approach to rendering visualizations from natural language queries, even when they are highly misspecified or underspecified.

### 6.3 Benchmark Datasets

The methodologies proposed in the studies have been evaluated using well-known CO instances, related problem suits (e.g., MIPLIB and ASLIB),[3] or datasets specifically developed for the case of LLMs. Considering this latter point, 29 studies assess the efficacy and efficiency of leveraging LLM within CO with specific benchmark datasets developed for this purpose. We now describe these datasets and report a tabular overview in Appendix E.

The most frequently used benchmark dataset is the LPWP, also referred to as the NL4Opt dataset, which has been employed in 16 studies [1, 3, 4, 47, 58, 74, 82, 89, 92, 116, 146, 157, 179, 225, 239, 249]. Initially introduced by Ramamonjison et al. [179] and later expanded by Li et al. [116], this dataset has been used in the NL4Opt competition. The original dataset includes 4,216 NL problem declarations derived from 1,101 LP problems across six different domains. The extension enhances the dataset by introducing new problem descriptions and constraint types, such as logic constraints and binary variables. The second most used benchmark dataset is the ComplexOR dataset [239], which has been employed in three studies [3, 92, 239] and has been introduced to complement the NL4Opt dataset. The dataset consists of NL descriptions of 37 problems across different domains, sourced from academic studies and real-world scenarios, encompassing 25 LP formulations and 12 MILP formulations. As ComplexOR also the NLP4LP dataset [2, 3] has been used in three studies [2, 3, 92]. It encompasses NL descriptions of 67 problems across various domains and including both LP (54) and MILP (13) formulations. This same dataset has been enriched to up to more than 200 optimization problems by AhmadiTeshnizi et al. [2]. Huang et al. [82] proposed OR-Instruct, a pipeline for creating synthetic data for optimization data. To test the capabilities of such a pipeline, they created IndustryOR, which has been used in two studies [82, 92]. As the name suggests, it focuses on industrial problems (a total of 100 real-world problems from eight industries) covering LP, ILP, MILP, and non-linear programming. Huang et al. [85] introduced Mamo, a dataset for LP modeling. It includes 652 easy and 211 complex LP problems, each paired with its corresponding optimal solution, sourced from various academic materials. Mamo has been used in two studies [82, 92]. Luo et al. [137] introduced GraphInstruct, a dataset comprising 21 reasoning problems on the topic of graphs and networks, including COPs. GraphInstruct has been used in two studies [80, 119] . Similar to this dataset, there is Talk Like A Graph [54], LLM4DyG [253], GraphViz [29], NLGraph [224], and GNN-AutoGL [119] (note that these datasets have been used only by one study [119]).

The remaining datasets have been used exclusively in one out of the retrieved studies—many times this being the study proposing the dataset on the first place. Amarasinghe et al. [8] introduced the AI-copilot-data dataset, which includes 100 NL descriptions of problems within the production scheduling domain. Huang et al. [87] introduced the homonym dataset,[4] which comprises 80 NL descriptions of routing problems, including variants for single-robot and multi-robot routing. Almonacid [7] introduced a homonym dataset, which comprises 10 NL instructions for the creation of MiniZinc models involving discrete variables and arrays of discrete variables, thus ILP problems. Hao Chen and Li [72] introduced the OptiChat dataset, which comprises 63 infeasible (i.e., inconsistent) MILP model formulations across various domains. This dataset was derived from feasible models expressed through the Python library Pyomo [73] and sourced from a collection of libraries and textbooks. The formulations were created by modifying one or more model parameters (e.g., minimum inventory, demand, and maximum capacity) or adding constraints (e.g., maximum

[3] https://miplib.zib.de/ and https://www.coseal.net/aslib/ (accessed on 2026-03-08).
[4] We use the term "homonym" when the authors did not provide a specific name for the dataset.

cost and minimum demand for a particular product) so to make the instances infeasible (i.e., their set of feasible solutions is empty). Lawless et al. [111] introduced two datasets, Safeguard and Code Generation. They both are strictly connected to the framework the authors proposed. The Safeguard dataset is a binary classification dataset designed to evaluate whether the system at hand (i.e., a system that integrates CP and LLMs to schedule meetings in a company) has sufficient data sources (e.g., information related to the meeting attendees) to handle given NL constraints. The Code Generation dataset contains a collection of NL constraints that can be translated into Python code using the data structures available in the system. An example of such constraints is "The team has a no-meeting policy on {WEEKDAY}". This dataset aims to test the capability of generating executable Python code that satisfies the specified constraints. While employing NL4Opt and other existing dataset from the ML community, Michailidis et al. [146] also introduced a homonym CP-based benchmark. The dataset was built using 18 CP problems from a university-level CP modelling course. Yang et al. [241] introduced OptiBench and ReSocratic-29k. OptiBench includes 816 real-world optimization problems spanning multiple domains, focusing on linear and mixed-integer programming and introducing a graph-based evaluation method to assess model correctness. ReSocratic-29k consists of 29,000 optimization problem demonstrations, generated through a reverse synthesis approach that first constructs step-by-step formulations before back-translating them into NL questions. These datasets provide a targeted benchmark for assessing and improving LLMs in formulating and solving optimization tasks. Borazjanizadeh et al. [20] introduced Search-Bench, a dataset encompassing five problem categories and 1,107 instances, primarily focused on puzzles and general combinatorics. Each problem type corresponds to a well-known COP, but the constraints have been slightly modified to reduce the likelihood that LLMs encountered identical problems during their training phase. Mostajabdaveh et al. [153] presented a new dataset to complement the existing NL4Opt and ComplexOR benchmarks, aiming to provide less structured input and more complex optimization scenarios. This benchmark includes problems related to LP, MILP, and QP. Unlike NL4Opt, which expects a formal model as output, and ComplexOR, which requires Python code, this dataset outputs solutions in Zimple code. Similarly, Zhang et al. [249] enriched the NL4Opt dataset with English and Chinese problem descriptions. Ju et al. [96] introduced a dataset of 173,700 training and 21,800 test samples related to travel planning.

Finally, a different dataset is ORQUA [152]. It is designed to evaluate the extent of CO knowledge in LLMs. Given a problem description, the dataset assesses the model's ability to understand and identify, for example, the appropriate type of mathematical modeling the problem corresponds to.

Similar to the platforms discussed in Section 6.2.6, we also identified general-purpose datasets in the reviewed studies. These datasets are valuable for developing LLMs-based approaches to optimization, particularly by providing math-related problems expressed in unstructured natural language which are useful especially for problem modeling (e.g., handling addition, multiplication, and data structures like sets). These resources can support researchers and practitioners in designing new applications of LLMs in CO. We briefly outline their characteristics and report some examples in Table 3. Notably, all of these resources were used by Ahmed and Choudhury [4], while GSM8K was also employed by Yang et al. [240].

GSM8K [35] consists of 8,500 linguistically diverse grade-school human-generated math word problems. Unlike simpler arithmetic datasets like AddSub [78] (395 problems) and SingleOp [105] (562 problems), GSM8K requires multi-step reasoning. Its emphasis on stepwise numerical reasoning aligns with CO, where problems often require sequential computations and recursive decision-making.

MultiArith [78] consists of around 600 multi-step arithmetic problems that require sequential operations (e.g., addition, subtraction, and multiplication). It was partially generated from existing math problems and structured for ML applications. Its focus on structured numerical reasoning

Table 3. Examples of Mathematical Reasoning Tasks from Various Benchmark Datasets

| Dataset | Task | Example |
|---|---|---|
| GSM8K [35] | Arithmetic Word Problem | *Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?* |
| MultiArith [78] | Multi-Step Arithmetic | *John has three apples. He buys five more apples and then eats two. How many apples does he have left?* |
| AquA [123] | Probability | *From a pack of 52 cards, two cards are drawn together at random. What is the probability of both the cards being kings?* |
| BIG-Bench [196] | Logical Reasoning | *If a snail climbs a 10-meter pole at a rate of 2 meters per day but slides back 1 meter each night, how many days will it take to reach the top?* |
| BIG-Bench Hard [201] | Combinatorial Reasoning | *A school has six different clubs. Each student must join exactly two clubs. How many unique pairs of clubs can be formed?* |

makes it relevant to CO techniques such as branch-and-bound and constraint satisfaction, which rely on constructive decision-making.

AquA [123] presents 100,000 human-generated and machine-generated algebraic word problems in a multiple-choice format along with rationales explaining each answer. It requires the computation of correct answers and their selection from distractors. Many COPs involve symbolic reasoning and equation solving, making AquA useful for assessing a model's ability to handle algebraic structures and optimization constraints.

BIG-Bench [196] is a large-scale benchmark evaluating language models across more than 200 tasks, covering mathematics, reasoning, linguistics, commonsense understanding, and code generation. The dataset was created through a combination of human-designed tasks and algorithmically generated problem sets, allowing for a broad evaluation of logical reasoning and heuristic problem-solving. This makes it valuable for studying how models approach optimization-based decision-making. A more challenging subset, BIG-Bench Hard [201], focuses on 23 particularly difficult tasks from BIG-Bench that remain unsolved or challenging for state-of-the-art models. Many of these challenges relate to CO, where finding optimal solutions in complex search spaces is a key difficulty.

### 6.4 Application Domains

CO has been applied to a wide range of problems across various domains, and over the years, these problems have been formalized into well-known prototypical models, such as the TSP, the VRP, and the **Capacited Vehicle Routing Problem (CVRP)** in the field of routing. Among the 103 studies, 64 explicitly address problems within specific domains or problem formulations—which we overview in this section. In this analysis, literature reviews that provide information on the use of LLM without offering actual implementations are excluded, such as those by Fan et al. [53], Wu et al. [237], and Zhao et al. [254]. Exceptions apply to reviews like the one by Saka et al. [187] that also verified the application of LLMs in CO within the construction domain.

The majority of these studies (26) focus on routing problems, particularly within the contexts of the TSP [30, 52, 81, 98, 124, 125, 127, 129, 131, 143, 199, 219, 240, 242, 244, 251], VRP [32, 37, 63, 86, 87, 98, 99, 129, 236, 244], orienteering [244] , and traveling [39, 96, 115] . While the first three COPs are well-known in the CO field, the latter refers to the modeling of the traveling activity (i.e., visiting a new city) as a COP. It cannot be considered as a TSP variant in all cases, as it is not given that the goal is to find a Hamiltonian path. Two other domains that have garnered significant attention are scheduling/planning (14 studies) and networks and graphs (10). In the first domain, studies have focused on variants of well-known benchmark problems, like the PFSP [8, 125] and planning [19, 45, 71, 93, 148, 158, 166, 227, 245], as well as specific scheduling problems, like server [243] and meeting/conference [95, 111] scheduling. Also for what concerns the Network and Graph domains, studies focused on general network design [80, 119, 134, 215, 246], but also on classical problems like coloring [143], social networks Sartori et al. [188], critical node identification [142], and path finding [20, 98].

The remaining studies focused on packing (9), combinatorics (4), engineering (3), finance (3), bioinformatics (3), supply chain (1), and strings (1). For a breakdown of specific COPs, we refer the interested reader to Appendix F.

A limited number of studies (9) explore multiple problem domains and formulations to demonstrate the capabilities of LLMs across various contexts [20, 30, 125, 143, 184, 219, 242, 244, 251]. Additionally, Khan and Hamad [98] tackled a diverse set of problems, which, however, can all be reduced to graph-based ones, while studies like the one by Lawless et al. [110] inherently address multiple problem domains as they are based on library of problems (initially not thought for applications in the field of LLMs), like the MIPLIB dataset.

### 6.5 Positions from Non-experimental Literature

Among the 103 selected studies, 11 are literature reviews, 7 are position papers, and 1 is a technical report.

Fan et al. [53], Wu et al. [237], Huang et al. [84], Lai et al. [109], Cai et al. [23], and Liu et al. [128] presented literature reviews on the topic of LLM and optimization or on strictly related fields, such as algorithm design (Section 3). Saka et al. [187], Zhao et al. [254], Wu et al. [236], Pallagani et al. [166], Sui et al. [199], and Long et al. [134] presented literature reviews that deal with CO in specific application domains (i.e., engineering, planning, VRP, planning and scheduling, TSP, and network optimization, respectively) and in some cases involve (preliminary) studies and experiments with LLM.

Wasserkrug et al. [229] put forward a position paper advocating for the usage of an LLM within optimization considering all the optimization steps. Similarly, Tsouros et al. [213] proposed a LLM-aided optimization pipeline in the context of CP modeling. Freuder [57] highlighted the potential of LLMs in facilitating the discussion between optimization and domain experts. Srivastava and Pallagani [197], Yu and Liu [247], and Ustyugov [217] presented insights on the usage of LLMs within EAs, planning-like tasks, and automated MILP configuration.

The only technical report identified is by Ramamonjison et al. [179]. The report includes insights and comparisons related to the NL4Opt challenge (Section 6.1 and 6.3).

## 7 Future Research Directions

While various aspects of the optimization process have been addressed in the existing literature (Section 6), certain areas still call for further exploration. We have identified several future research directions:

- *Metaheuristics*: The adoption of LLMs in MH frameworks has been limited and scattered. Future research could investigate how LLMs can be used to dynamically adjust MH strategies, optimizing parameters (as anticipated in some studies [63, 110, 143]) or switching strategies based on the current state of the search. This could enhance the flexibility and effectiveness of MHs. Future studies could explore how LLMs might expand local search neighborhoods by suggesting structures or transition operators.
- *Algorithm Selection*: Utilizing LLMs to explore the search space of algorithm selection might be winning (as anticipated by Wu et al. [238]) and LLMs could help pinpoint scenarios where current solvers are less effective. Additionally, LLMs could be used to generate instances specifically designed to test the strengths and weaknesses of these solvers, leading to improved performance insights.
- *Synthetic Instance Generation*: LLMs could be leveraged to create synthetic instances that replicate the complexities of real-world problems or that are specifically crafted to challenge existing algorithms. This approach has been anticipated in the IndustryOR dataset [82].

- *LLMs behavior w.r.t. NLP problem description*: It remains unclear to what extent LLMs adjust their responses depending on how COPs are described. Understanding the sensitivity of these models to different textual and non-textual formulations of the same problem represents a promising research direction.
- *Evaluation Protocol*: Evaluating the performance of LLMs on COPs remains challenging due to several factors: problems can have multiple representations, various encoding methods, and often lack known optimal solutions. While some efforts exist, a promising research direction involves developing standardized evaluation protocols and identifying suitable metrics.
- *Agent-based System and COPs*: Recent studies have explored how LLMs can function as autonomous agents [226]. While some of the studies already use an ensemble of LLMs, a promising research direction would be to investigate thoroughly how these autonomous agents can be effectively utilized within CO.

Additional more general considerations for future research include ethical and bias considerations. As LLMs are integrated into optimization frameworks, research should also focus on ensuring that these technologies are used responsibly. This includes considering the environmental impact of the resources required by LLMs and developing more sustainable approaches to large-scale optimization. Future research should examine the potential biases in LLM-generated optimization solutions. More generally, strategies for ensuring fairness as well as incorporating ethical considerations into the optimization process could be explored.

## 8 Limitations

While our systematic review offers valuable insights into the use of LLMs in CO, it has some limitations. First, research on LLMs and their application to optimization is advancing rapidly, with new studies emerging continuously. Consequently, while this systematic review attempts to be as comprehensive as possible, it inevitably cannot cover all existing studies. Also, systematic reviews depend on data gathered from other studies, meaning the quality and bias level of the evidence in a systematic review are directly tied to those of the data sources [50]. Therefore, we acknowledge that following PRISMA guidelines might have led to the inclusion of too many studies that report positive outcomes or successful applications of LLM over those that do not, potentially overstating the effectiveness or applicability of LLM in this field due to inherent selection bias.

Then, our review focuses solely on how LLMs can be applied to CO. An equally significant aspect not covered in this study is how CO techniques can be employed to enhance LLMs, which is dual to our main focus. By not addressing this aspect, which can be pursued in future work, our review may miss relationships and interdependencies between the two research fields. Examples of such works include Pan et al. [167], who experiment with using MHs to engineer LLM prompts. More specifically, they use—among other methods—**Hill Climbing (HC)** and SA, to discover and learn new efficient prompts. Another example is the work by Singla et al. [193], who model the trade-off between response quality and inference cost of LLM as a bi-objective combinatorial optimization problem. Additionally, this study focuses on CO, disregarding other types of optimizations, such as Continuous Optimization. Examples of such works are those by Pluhacek et al. [171, 172].

We intentionally included a significant number of pre-prints alongside peer-reviewed publications (Section 5). Such a decision is driven by the fast-paced nature of research in the field of LLMs, where discoveries and advancements are rapidly shared through pre-prints before formal publication (see, for example, Devlin et al. [44]). Pre-prints allow for timely access to the latest research, innovations, and discussions and are widely used within the NLP community. We take no position on the content of pre-prints found on platforms like arXiv; instead, we include these documents to maximize the recall of our systematic review.

Eventually, our systematic review includes studies based on closed-source LLMs, like ChatGPT. The proprietary nature of these systems often restricts access to their full methodologies and inner workings, which creates significant challenges for understanding and replicating the reported findings. Despite these downsides, we include works dealing both with open-source and closed-source LLMs to maximize the recall of our systematic review.

## 9 Conclusions and Future Work

In this systematic review, we examined the application of LLMs to CO. Our study summarizes the literature leveraging the PRISMA 2020 guidelines. To our knowledge, this is the first attempt to comprehensively study the application of LLMs to CO and COP. Out of over 2,000 collected publications, we included 103 studies in our analysis. These studies were classified based on the task LLMs performed within the optimization process, the implementation details of the LLMs, the most commonly used datasets, and the application domains where LLMs have been employed in CO thus far. Additionally, we highlighted the caveats associated with using LLMs in the field of optimization, outlined future research directions, and exposed the limitations of the present review.

The research on LLMs is rapidly evolving, thus continuously identifying areas for further research is crucial. We plan to periodically update our systematic review, as advocated by the PRISMA guidelines [163]. This process ensures that our review remains relevant in this fast-moving field. In particular, updates will allow us to reassess studies that were initially included as pre-prints once they appear in peer-reviewed venues (thus solving one of the limitations of our work, Section 8), as well as to incorporate new contributions published after our search cutoff at the end of 2024, such as Michailidis et al. [147] and Singirikonda et al. [192]. Future research will also account for the aspects neglected by this article, such as the integration of LLMs into optimization paradigms other than CO and for the dual aspect of optimization used for enhancing LLMs.

With much of the state-of-the-art work being conducted relying on proprietary models such as ChatGPT, future studies will also focus on developing methods for assessing and reporting on such models to improve the transparency and replicability of the presented studies. This could involve creating frameworks for sharing results that do not compromise proprietary data but still provide sufficient detail for academic reproducibility/replicability (as also advocated by many researchers in the optimization field [202, 203]).

As LLMs are deployed in critical areas, their ethical implications, particularly in sensitive optimization tasks, require closer examination especially under the ethical point of view. Finally, addressing biases in LLM outputs remains a significant concern and is essential for ensuring fair and responsible applications of LLMs in CO.

## Data Availability

The data supporting the findings of this study are available within the paper and its appendices.

## References

[1] Yelaman Abdullin, Diego Molla, Bahadorreza Ofoghi, John Yearwood, and Qingyang Li. 2023. Synthetic dialogue dataset generation using LLM agents. In *Proceedings of the 3rd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*. ACL, Singapore, 181–191. Retrieved from https://aclanthology.org/2023.gem-1.16

[2] Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan Talaei, and Madeleine Udell. 2024. OptiMUS-0.3: Using large language models to model and solve optimization problems at scale. arXiv:2407.19633. Retrieved from https://arxiv.org/abs/2407.19633 (2024).

[3] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. 2024. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In *Proceedings of the 41st International Conference on Machine Learning (ICML '24)*. JMLR.org, Vienna, Austria, 1234–1245. DOI: 10.5555/3692070.3692094

[4] Tasnim Ahmed and Salimur Choudhury. 2024. LM4OPT: Unveiling the potential of large language models in formulating mathematical optimization problems. *INFOR: Information Systems and Operational Research* 62, 4 (2024), 559–572. DOI: 10.1080/03155986.2024.2388452

[5] Cohere AI. 2024. Command R+: A Large Language Model for Enterprise AI. Retrieved March 9, 2026 from https://docs.cohere.com/v2/docs/command-r-plus

[6] Mohammad Alipour-Vaezi and Kwok-Leung Tsui. 2024. Data-driven portfolio management for motion pictures industry: A new data-driven optimization methodology using a large language model as the expert. *Computers & Industrial Engineering* 197 (2024), 110574. DOI: 10.1016/j.cie.2024.110574

[7] Boris Almonacid. 2023. Towards an automatic optimisation model generator assisted with generative pre-trained transformer. arXiv:2305.05811. Retrieved from https://arxiv.org/abs/2305.05811 (2023).

[8] Pivithuru Thejan Amarasinghe, Su Nguyen, Yuan Sun, and Damminda Alahakoon. 2023. AI-copilot for business optimisation: A framework and a case study in production scheduling. arXiv:2309.13218. Retrieved from https://arxiv.org/abs/2309.13218 (2023).

[9] Sotiris Anagnostidis and Jannis Bulian. 2024. How susceptible are LLMs to influence in prompts? arXiv:2408.11865. Retrieved from https://arxiv.org/abs/2408.11865 (2024).

[10] Anthropic. 2024. Claude 3.5 Sonnet Model Card Addendum. Retrieved March 9, 2026 from https://www.anthropic.com/news/claude-3-5-sonnet

[11] Anthropic Team. 2024. Introducing the next generation of Claude. Retrieved June 18, 2024 from https://www.anthropic.com/news/claude-3-family

[12] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2016. Neural machine translation by jointly learning to align and translate. arXiv:1409.0473. Retrieved from https://arxiv.org/abs/1409.0473 (2016).

[13] Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the ACL (Volume 1: Long Papers)*. ACL, St. Julian's, Malta, 67–93. Retrieved from https://aclanthology.org/2024.eacl-long.5

[14] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. 2021. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research* 290, 2 (2021), 405–421. DOI: 10.1016/j.ejor.2020.07.063

[15] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 16 (3 2024), 17682–17690. DOI: 10.1609/aaai.v38i16.29720

[16] Roberto Bifulco, Federico Errica, Davide Sanvito, and Giuseppe Siracusano. 2024. What did I do wrong? Quantifying LLMs' sensitivity and consistency to prompt engineering. arXiv:2406.12334. Retrieved from https://arxiv.org/abs/2406.12334 (2024).

[17] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-objective optimization in Python. *IEEE Access* 8 (2020), 89497–89509. DOI: 10.1109/ACCESS.2020.2990567

[18] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys* 35, 3 (9 2003), 268–308. DOI: 10.1145/937503.937505

[19] Bernd Bohnet, Azade Nova, Aaron T. Parisi, Kevin Swersky, Katayoon Goshvadi, Hanjun Dai, Dale Schuurmans, Noah Fiedel, and Hanie Sedghi. 2024. Exploring and benchmarking the planning capabilities of large language models. arXiv:2406.13094. Retrieved from https://arxiv.org/abs/2406.13094 (2024).

[20] Nasim Borazjanizadeh, Roei Herzig, Trevor Darrell, Rogerio Feris, and Leonid Karlinsky. 2024. Navigating the labyrinth: Evaluating and enhancing LLMs' ability to reason about search problems. arXiv:2406.12172. Retrieved from https://arxiv.org/abs/2406.12172 (2024).

[21] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, Vol. 33. Curran Associates, Inc., Virtual Conference, 1877–1901. DOI: 10.5555/3495724.3495883

[22] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. arXiv:2303.12712. Retrieved from https://arxiv.org/abs/2303.12712 (2023).

[23] Jinyu Cai, Jinglue Xu, Jialong Li, Takuto Yamauchi, Hitoshi Iba, and Kenji Tei. 2024. Exploring the improvement of evolutionary computation via large language models. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Melbourne, VIC, Australia) *(GECCO '24 Companion)*. ACM, New York, NY, USA, 83–84. DOI: 10.1145/3638530.3664086

[24] Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. InternLM2 technical report. arXiv:2403.17297. Retrieved from https://arxiv.org/abs/2403.17297 (2024).

[25] Sara Ceschia, Luca Di Gaspero, Vincenzo Mazzaracchio, Giuseppe Policante, and Andrea Schaerf. 2023. Solving a real-world nurse rostering problem by Simulated Annealing. *Operations Research for Health Care* 36 (2023), 100379. DOI: 10.1016/j.orhc.2023.100379

[26] Sara Ceschia, Luca Di Gaspero, and Andrea Schaerf. 2023. Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research* 308, 1 (2023), 1–18. DOI: 10.1016/j.ejor.2022.07.011

[27] Sara Ceschia and Andrea Schaerf. 2024. Multi-neighborhood simulated annealing for the capacitated facility location problem with customer incompatibilities. *Computers & Industrial Engineering* 188 (2024), 109858. DOI: 10.1016/j.cie.2023.109858

[28] Camilo Chacón Sartori, Christian Blum, and Gabriela Ochoa. 2024. Large language models for the automated analysis of optimization algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24)*. ACM, New York, NY, USA, 160–168. DOI: 10.1145/3638529.3654086

[29] Nuo Chen, Yuhan Li, Jianheng Tang, and Jia Li. 2024. GraphWiz: An instruction-following language model for graph problems. arXiv:2402.16029. Retrieved from https://arxiv.org/abs/2402.16029 (2024).

[30] Zijie Chen, Zhanchao Zhou, Yu Lu, Renjun Xu, Lili Pan, and Zhenzhong Lan. 2024. UBER: Uncertainty-based evolution with large language models for automatic heuristic design. arXiv:2412.20694. Retrieved from https://arxiv.org/abs/2412.20694 (2024).

[31] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, et al. 2024. Chatbot arena: An open platform for evaluating LLMs by human preference. arXiv:2403.04132. Retrieved from https://arxiv.org/abs/2403.04132 (2024).

[32] Samuel J. K. Chin, Matthias Winkenbach, and Akash Srivastava. 2024. Learning to deliver: A foundation model for the Montreal Capacitated Vehicle Routing Problem. arXiv:2403.00026. Retrieved from https://arxiv.org/abs/2403.00026 (2024).

[33] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2024. PaLM: Scaling language modeling with pathways. *Journal of Machine Learning Research* 24, 1, Article 240 (3 2024), 113 pages. DOI: 10.5555/3648699.3648939

[34] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research* 25, 70 (2024), 1–53. Retrieved from http://jmlr.org/papers/v25/23-0870.html

[35] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. arXiv:2110.14168. Retrieved from https://arxiv.org/abs/2110.14168 (2021).

[36] Chris Cooper, Andrew Booth, Nicky Britten, and Ruth Garside. 2017. A comparison of results of empirical studies of supplementary search techniques and recommendations in review methodology handbooks: a methodological review. *Systematic Reviews* 6, 1 (28 Nov 2017), 234. DOI: 10.1186/s13643-017-0625-1

[37] Longchao Da, Kuanru Liou, Tiejin Chen, Xuesong Zhou, Xiangyong Luo, Yezhou Yang, and Hua Wei. 2024. Open-ti: Open traffic intelligence with augmented language model. *International Journal of Machine Learning and Cybernetics* 15, 10 (01 Oct 2024), 4761–4786. DOI: 10.1007/s13042-024-02190-8

[38] Parag Pravin Dakle, Serdar Kadıoğlu, Karthik Uppuluri, Regina Politi, Preethi Raghavan, SaiKrishna Rallabandi, and Ravisutha Srinivasamurthy. 2023. Ner4Opt: Named entity recognition for optimization modelling from natural language. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer Nature Switzerland, Cham, 299–319. DOI: 10.1007/978-3-031-33271-5_20

[39] Tomas De La Rosa, Sriram Gopalakrishnan, Alberto Pozanco, Zhen Zeng, and Daniel Borrajo. 2024. TRIP-PAL: Travel planning with guarantees by combining large language models and automated planners. arXiv:2406.10196. Retrieved from https://arxiv.org/abs/2406.10196 (2024).

[40] I. De Zarzà, J. De Curtò, Gemma Roig, and Carlos T. Calafate. 2024. Optimized financial planning: Integrating individual and cooperative budgeting models with LLM recommendations. *AI* 5, 1 (2024), 91–114. DOI: 10.3390/ai5010006

[41] DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, and Qiushi Du. 2024. DeepSeek LLM: Scaling open-source language models with longtermism. arXiv:2401.02954. Retrieved from https://arxiv.org/abs/2401.02954 (2024).

[42] DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, and Damai Dai. 2024. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. arXiv:2405.04434. Retrieved from https://arxiv.org/abs/2405.04434 (2024).

[43] DeepSeek-AI, Qihao Zhu, Daya Guo, Zhihong Shao, Dejian Yang, Peiyi Wang, Runxin Xu, Y. Wu, Yukun Li, and Huazuo Gao. 2024. DeepSeek-Coder-V2: Breaking the barrier of closed-source models in code intelligence. arXiv:2406.11931. Retrieved from https://arxiv.org/abs/2406.11931 (2024).

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the*

*ACL: Human Language Technologies, Volume 1 (Long and Short Papers)*. ACL, Minneapolis, Minnesota, 4171–4186. DOI: 10.18653/v1/N19-1423

[45] Neel Dhanaraj, Minseok Jeon, Jeon Ho Kang, Stefanos Nikolaidis, and Satyandra K. Gupta. 2024. Preference elicitation and incorporation for human-robot task scheduling. In *Proceedings of the 2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*. IEEE, Bari, Italy, 3103–3110. DOI: 10.1109/CASE59546.2024.10711695

[46] Steven Diamond and Stephen Boyd. 2016. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research* 17, 1 (1 2016), 2909–2913. DOI: 10.5555/2946645.3007036

[47] Xuan-Dung Doan. 2022. VTCC-NLP at NL4Opt competition subtask 1: An ensemble pre-trained language models for named entity recognition. arXiv:2212.07219. Retrieved from https://arxiv.org/abs/2212.07219 (2022).

[48] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. 2006. Ant colony optimization. *IEEE Computational Intelligence Magazine* 1, 4 (2006), 28–39. DOI: 10.1109/MCI.2006.329691

[49] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning (ICML'23)*. JMLR.org, Honolulu, Hawaii, USA, Article 340, 20 pages. DOI: 10.5555/3618408.3618748

[50] Aaron M. Drucker, Patrick Fleming, and An-Wen Chan. 2016. Research techniques made simple: Assessing risk of bias in systematic reviews. *Journal of Investigative Dermatology* 136, 11 (2016), e109–e114. DOI: 10.1016/j.jid.2016.08.021

[51] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. 2022. GLM: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the ACL (Volume 1: Long Papers)*. ACL, Dublin, Ireland, 320–335. DOI: 10.18653/v1/2022.acl-long.26

[52] Mohammed Elhenawy, Ahmad Abutahoun, Taqwa I. Alhadidi, Ahmed Jaber, Huthaifa I. Ashqar, Shadi Jaradat, Ahmed Abdelhay, Sebastien Glaser, and Andry Rakotonirainy. 2024. Visual reasoning and multi-agent approach in multimodal large language models (MLLMs): Solving TSP and mTSP combinatorial challenges. *Machine Learning and Knowledge Extraction* 6, 3 (2024), 1894–1920. DOI: 10.3390/make6030093

[53] Zhenan Fan, Bissan Ghaddar, Xinglu Wang, Linzi Xing, Yong Zhang, and Zirui Zhou. 2024. Artificial intelligence for operations research: Revolutionizing the operations research process. arXiv:2401.03244. Retrieved from https://arxiv.org/abs/2401.03244 (2024).

[54] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. arXiv:2310.04560. Retrieved from https://arxiv.org/abs/2310.04560 (2023).

[55] Alberto Franzin and Thomas Stützle. 2019. Revisiting simulated annealing: A component-based analysis. *Computers & Operations Research* 104 (2019), 191–206. DOI: 10.1016/j.cor.2018.12.015

[56] Yolanda Freire, Andrea Santamaría Laorden, Jaime Orejas Pérez, Margarita Gómez Sánchez, Víctor Díaz-Flores García, and Ana Suárez. 2024. ChatGPT performance in prosthodontics: Assessment of accuracy and repeatability in answer generation. *The Journal of Prosthetic Dentistry* 131, 4 (2024), 659.e1–659.e6. DOI: 10.1016/j.prosdent.2024.01.018

[57] Eugene C. Freuder. 2024. Conversational modeling for constraint satisfaction. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 20 (Mar. 2024), 22592–22597. DOI: 10.1609/aaai.v38i20.30268

[58] Neeraj Gangwar and Nickvash Kani. 2023. Highlighting named entities in input for auto-formulation of optimization problems. In *Intelligent Computer Mathematics*. Springer Nature Switzerland, Cham, 130–141. DOI: 10.1007/978-3-031-42753-4_9

[59] M. R. Garey, D. S. Johnson, and Ravi Sethi. 1976. The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research* 1, 2 (1976), 117–129. Retrieved from http://www.jstor.org/stable/3689278

[60] Gecode Team. 2006. Gecode: Generic Constraint Development Environment. Retrieved June 27, 2024 from http://www.gecode.org

[61] Meta GenAI. 2023. Camels in a changing climate: Enhancing LM adaptation with Tulu 2. arXiv:2311.10702. Retrieved from https://arxiv.org/abs/2311.10702 (2023).

[62] Meta GenAI. 2023. Llama 2: Open foundation and fine-tuned chat models. arXiv:2307.09288. Retrieved from https://arxiv.org/abs/2307.09288 (2023).

[63] Gianpaolo Ghiani, Gianluca Solazzo, and Gianluca Elia. 2024. Integrating large language models and optimization in semi-structured decision making: Methodology and a case study. *Algorithms* 17, 12 (2024), 15 pages. DOI: 10.3390/a17120582

[64] Ida Gjergji and Nysret Musliu. 2024. Large neighborhood search for the capacitated P-median problem. In *Metaheuristics*. Springer Nature Switzerland, Cham, 158–173. DOI: 10.1007/978-3-031-62922-8_11

[65] Fred Glover. 1997. *Tabu Search*. Springer, New York, NY. DOI: 10.1007/978-1-4615-6089-0

[66] Google. 2023. PaLM 2 technical report. arXiv:2305.10403. Retrieved from https://arxiv.org/abs/2305.10403 (2023).

[67] Tias Guns. 2019. Increasing modeling language convenience with a universal N-dimensional array: CPpy as a Python-embedded example. In *Proceedings of the 18th Workshop on Constraint Modelling and Reformulation at CP (ModRef 2019)*. ACP, Stamford, CT, USA, 8 pages. Retrieved March 9, 2026 from https://modref.github.io/ModRef2019.html

[68] Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, and Jian Yin. 2022. UniXcoder: Unified cross-modal pre-training for code representation. In *Proceedings of the 60th Annual Meeting of the ACL (Volume 1: Long Papers)*. ACL, Dublin, Ireland, 7212–7225. DOI: 10.18653/v1/2022.acl-long.499

[69] Pei-Fu Guo, Ying-Hsuan Chen, Yun-Da Tsai, and Shou-De Lin. 2024. Towards optimizing with large language models. arXiv:2310.05204. Retrieved from https://arxiv.org/abs/2310.05204 (2024).

[70] Gurobi Optimization, LLC. 2023. Gurobi Optimizer Reference Manual. Retrieved from https://www.gurobi.com

[71] Yilun Hao, Yang Zhang, and Chuchu Fan. 2024. Planning anything with rigor: General-purpose zero-shot planning with LLM-based formalized programming. arXiv:2410.12112. Retrieved from https://arxiv.org/abs/2410.12112 (2024).

[72] Gonzalo E. Constante-Flores Hao Chen and Can Li. 2024. Diagnosing infeasible optimization problems using large language models. *INFOR: Information Systems and Operational Research* 62, 4 (2024), 573–587. DOI: 10.1080/03155986.2024.2385189

[73] William E. Hart, Jean-Paul Watson, and David L. Woodruff. 2011. Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation* 3, 3 (01 Sep 2011), 219–260. DOI: 10.1007/s12532-011-0026-8

[74] JiangLong He, Mamatha N, Shiv Vignesh, Deepak Kumar, and Akshay Uppal. 2022. Linear programming word problems formulation using EnsembleCRF NER labeler and text generator with data augmentations. arXiv:2212.14657. Retrieved from https://arxiv.org/abs/2212.14657 (2022).

[75] Michael Heinrich, Luisa Hofmann, Hansjörg Baurecht, Peter M. Kreuzer, Helge Knüttel, Michael F. Leitzmann, and Corinna Seliger. 2022. Suicide risk and mortality among patients with cancer. *Nature Medicine* 28, 4 (01 Apr 2022), 852–859. DOI: 10.1038/s41591-022-01745-y

[76] John H. Holland. 1992. Genetic Algorithms. *Scientific American* 267, 1 (1992), 66–73. Retrieved June 27, 2024 from http://www.jstor.org/stable/24939139

[77] Holger H. Hoos. 2012. Programming by optimization. *Communications of the ACM* 55, 2 (feb 2012), 70–80. DOI: 10.1145/2076450.2076469

[78] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL, Doha, Qatar, 523–533. DOI: 10.3115/v1/D14-1058

[79] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. arXiv:2106.09685. Retrieved from https://arxiv.org/abs/2106.09685 (2021).

[80] Yuwei Hu, Runlin Lei, Xinyi Huang, Zhewei Wei, and Yongchao Liu. 2024. Scalable and accurate graph reasoning with LLM-based multi-agents. arXiv:2410.05130. Retrieved from https://arxiv.org/abs/2410.05130 (2024).

[81] Beichen Huang, Xingyu Wu, Yu Zhou, Jibin Wu, Liang Feng, Ran Cheng, and Kay Chen Tan. 2024. Exploring the true potential: Evaluating the black-box optimization capability of large language models. arXiv:2404.06290. Retrieved from https://arxiv.org/abs/2404.06290 (2024).

[82] Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. 2024. ORLM: A customizable framework in training large models for automated optimization modeling. arXiv:2405.17743. Retrieved from https://arxiv.org/abs/2405.17743 (2024).

[83] Hao Huang et al. 2024. The open cookbook for top-tier code large language models. arXiv:2411.04905. Retrieved from https://arxiv.org/abs/2411.04905 (2024).

[84] Sen Huang, Kaixiang Yang, Sheng Qi, and Rui Wang. 2024. When large language model meets optimization. *Swarm and Evolutionary Computation* 90 (2024), 101663. DOI: 10.1016/j.swevo.2024.101663

[85] Xuhan Huang, Qingning Shen, Yan Hu, Anningzhe Gao, and Benyou Wang. 2025. LLMs for mathematical modeling: Towards bridging the gap between natural and mathematical languages. arXiv:2405.13144. Retrieved from https://arxiv.org/abs/2405.13144 (2025).

[86] Yuxiao Huang, Wenjie Zhang, Liang Feng, Xingyu Wu, and Kay Chen Tan. 2024. How multimodal integration boost the performance of LLM for optimization: Case study on capacitated vehicle routing problems. arXiv:2403.01757. Retrieved from https://arxiv.org/abs/2403.01757 (2024).

[87] Zhehui Huang, Guangyao Shi, and Gaurav S. Sukhatme. 2024. From words to routes: Applying large language models to vehicle routing. arXiv:2403.10795. Retrieved from https://arxiv.org/abs/2403.10795 (2024).

[88] IBM 2017. *IBM ILOG CPLEX Optimization Studio, Getting Started with Scheduling in CPLEX Studio*. IBM. Retrieved June 27, 2024 from https://www.ibm.com/docs/en/icos/20.1.0?topic=kit-getting-started-scheduling-in-cplex-studio

[89] Sanghwan Jang. 2022. Tag embedding and well-defined intermediate representation improve auto-formulation of problem description. arXiv:2212.03575. Retrieved from https://arxiv.org/abs/2212.03575 (2022).

[90] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7B. arXiv:2310.06825. Retrieved from https://arxiv.org/abs/2310.06825 (2023).

[91] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. arXiv:2401.04088. Retrieved from https://arxiv.org/abs/2401.04088 (2024).

[92] Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. 2025. LLMOPT: Learning to define and solve general optimization problems from scratch. In *Proceedings of the 13th International Conference on Learning Representations*. Conference Organizers, Singapore, 3160–3172. Retrieved from https://openreview.net/forum?id=9OMvtboTJg

[93] Shuo Jiang, Min Xie, and Jianxi Luo. 2024. Large language models for combinatorial optimization of design structure matrix. arXiv:2411.12571. Retrieved from https://arxiv.org/abs/2411.12571 (2024).

[94] Ming Jin, Bilgehan Sel, Fnu Hardeep, and Wotao Yin. 2024. Democratizing energy management with LLM-assisted optimization autoformalism. In *Proceedings of the 2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE Communications Society, Oslo, Norway, 258–263. DOI: 10.1109/SmartGridComm60555.2024.10738100

[95] Deddy Jobson and Yilin Li. 2024. Investigating the potential of using large language models for scheduling. In *Proceedings of the 1st ACM International Conference on AI-Powered Software* (Porto de Galinhas, Brazil) *(AIware 2024)*. ACM, New York, NY, USA, 170–171. DOI: 10.1145/3664646.3665084

[96] Da Ju, Song Jiang, Andrew Cohen, Aaron Foss, Sasha Mitts, Arman Zharmagambetov, Brandon Amos, Xian Li, Justine T. Kao, Maryam Fazel-Zarandi, et al. 2024. To the globe (TTG): Towards language-driven guaranteed travel planning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. ACL, Miami, Florida, USA, 240–249. DOI: 10.18653/v1/2024.emnlp-demo.25

[97] Maryam Karimi-Mamaghan, Mehrdad Mohammadi, Patrick Meyer, Amir Mohammad Karimi-Mamaghan, and El-Ghazali Talbi. 2022. Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research* 296, 2 (2022), 393–422. DOI: 10.1016/j.ejor.2021.04.032

[98] Muhammad Asif Khan and Layth Hamad. 2024. On the Capability of LLMs in Combinatorial Optimization. TechRxiv. DOI: 10.36227/techrxiv.173092026.60478567/v1

[99] Daisuke Kikuta, Hiroki Ikeuchi, Kengo Tajiri, and Yuusuke Nakano. 2024. RouteExplainer: An explanation framework for vehicle routing problem. In *Advances in Knowledge Discovery and Data Mining*. Springer Nature Singapore, Singapore, 30–42. DOI: 10.1007/978-981-97-2259-4_3

[100] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680. DOI: 10.1126/science.220.4598.671

[101] Lucas Kletzander and Nysret Musliu. 2020. Solving the general employee scheduling problem. *Computers & Operations Research* 113 (2020), 104794. DOI: 10.1016/j.cor.2019.104794

[102] Lucas Kletzander and Nysret Musliu. 2024. Hyper-heuristics for personnel scheduling domains. *Artificial Intelligence* 334 (2024), 104172. DOI: 10.1016/j.artint.2024.104172

[103] Krzysztof Kochanek, Henryk Skarzynski, and Wiktor W. Jedrzejczak. 2024. Accuracy and repeatability of chatgpt based on a set of multiple-choice questions on objective tests of hearing. *Cureus* 16, 5 (5 2024), 10 pages. DOI: 10.7759/cureus.59857

[104] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. arXiv:2205.11916. Retrieved from https://arxiv.org/abs/2205.11916 (2023).

[105] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2015. Parsing algebraic word problems into equations. *Transactions of the ACL* 3 (2015), 585–597. DOI: 10.1162/tacl_a_00153

[106] Wiesław Kubiak. 2021. On a conjecture for the university timetabling problem. *Discrete Applied Mathematics* 299 (2021), 26–49. DOI: 10.1016/j.dam.2021.04.010

[107] Marie-Louise Lackner, Christoph Mrkvicka, Nysret Musliu, Daniel Walkiewicz, and Felix Winter. 2023. Exact methods for the oven scheduling problem. *Constraints* 28, 2 (01 Jun 2023), 320–361. DOI: 10.1007/s10601-023-09347-2

[108] Manuel Laguna, Rafael Martí, Anna Martinez-Gavara, Sergio Perez-Peló, and Mauricio G. C. Resende. 2023. 20 years of greedy randomized adaptive search procedures with path relinking. arXiv:2312.12663. Retrieved from https://arxiv.org/abs/2312.12663 (2023).

[109] Han Lai, Bo Wang, Jiaqi Liu, Feijuan He, Chenxi Zhang, Haohan Liu, and Haoran Chen. 2024. Solving Mathematical Problems Using Large Language Models: A Survey. SSRN. DOI: 10.2139/ssrn.5002356

[110] Connor Lawless, Yingxi Li, Anders Wikum, Madeleine Udell, and Ellen Vitercik. 2024. LLMs for cold-start cutting plane separator configuration. arXiv:2412.12038. Retrieved from https://arxiv.org/abs/2412.12038 (2024).

[111] Connor Lawless, Jakob Schoeffer, Lindy Le, Kael Rowan, Shilad Sen, Cristina St. Hill, Jina Suh, and Bahareh Sarrafzadeh. 2024. "I Want It That Way": Enabling Interactive Decision Support Using Large Language Models and Constraint Programming. *ACM Transactions on Interactive Intelligent Systems* 14, 3 (Sept. 2024), 8432–8448. DOI: 10.1145/3685053

[112] Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. 2022. CodeRL: Mastering Code Generation Through Pretrained Models and Deep Reinforcement Learning. 15 pages. DOI: 10.5555/3600270.3601819

[113] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv:1910.13461. Retrieved from https://arxiv.org/abs/1910.13461 (2019).

[114] Beibin Li, Konstantina Mellou, Bo Zhang, Jeevan Pathuri, and Ishai Menache. 2023. Large language models for supply chain optimization. arXiv:2307.03875. Retrieved from https://arxiv.org/abs/2307.03875 (2023).

[115] Bohang Li, Kai Zhang, Yiping Sun, and Jianke Zou. 2024. Research on travel route planning optimization based on large language model. In *Proceedings of the 2024 6th International Conference on Data-driven Optimization of Complex Systems (DOCS)*. IEEE, Hangzhou, China, 352–357. DOI: 10.1109/DOCS63458.2024.10704489

[116] Qingyang Li, Lele Zhang, and Vicky Mak-Hau. 2023. Synthesizing mixed-integer linear programming models from natural language descriptions. arXiv:2311.15271. Retrieved from https://arxiv.org/abs/2311.15271 (2023).

[117] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, et al. 2023. StarCoder: May the source be with you! arXiv:2305.06161. Retrieved from https://arxiv.org/abs/2305.06161 (2023).

[118] Sirui Li, Janardhan Kulkarni, Ishai Menache, Cathy Wu, and Beibin Li. 2024. Towards foundation models for mixed integer linear programming. arXiv:2410.08288. Retrieved from https://arxiv.org/abs/2410.08288 (2024).

[119] Xin Li, Qizhi Chu, Yubin Chen, Yang Liu, Yaoqi Liu, Zekai Yu, Weize Chen, Chen Qian, Chuan Shi, and Cheng Yang. 2024. Facilitating large language model-based graph analysis via multi-agent collaboration. arXiv:2410.18032. Retrieved from https://arxiv.org/abs/2410.18032 (2024).

[120] Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. ACL, Barcelona, Spain, 74–81. Retrieved from https://aclanthology.org/W04-1013

[121] Hanli Lin, Liqun Zhang, Ruizhi Zheng, and Yishan Zheng. 2017. The prevalence, metabolic risk and effects of lifestyle intervention for metabolically healthy obesity: A systematic review and meta-analysis: A PRISMA-compliant article. *Medicine* 96, 47 (2017), 9 pages. DOI: 10.1097/MD.0000000000008838

[122] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. 2022. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research* 23, 54 (2022), 1–9. DOI: 10.5555/3586589.3586643

[123] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the ACL (Volume 1: Long Papers)*. ACL, Vancouver, Canada, 158–167. DOI: 10.18653/v1/P17-1015

[124] Fei Liu, Xi Lin, Zhenkun Wang, Shunyu Yao, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. 2024. Large language model for multi-objective evolutionary optimization. arXiv:2310.12541. Retrieved from https://arxiv.org/abs/2310.12541 (2024).

[125] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. arXiv:2401.02051. Retrieved from https://arxiv.org/abs/2401.02051 (2024).

[126] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*. JMLR.org, Vienna, Austria, 23 pages. DOI: 10.5555/3692070.3693374

[127] Fei Liu, Xialiang Tong, Mingxuan Yuan, and Qingfu Zhang. 2023. Algorithm evolution using large language model. arXiv:2311.15249. Retrieved from https://arxiv.org/abs/2311.15249 (2023).

[128] Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Zhe Zhao, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, et al. 2024. A systematic survey on large language models for algorithm design. arXiv:2410.14716. Retrieved from https://arxiv.org/abs/2410.14716 (2024).

[129] Fei Liu, Rui Zhang, Zhuoliang Xie, Rui Sun, Kai Li, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. LLM4AD: A platform for algorithm design with large language model. arXiv:2412.17287. Retrieved from https://arxiv.org/abs/2412.17287 (2024).

[130] Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024. Large language models as evolutionary optimizers. arXiv:2310.19046. Retrieved from https://arxiv.org/abs/2310.19046 (2024).

[131] Yang Liu, Fanyou Wu, Zhiyuan Liu, Kai Wang, Feiyue Wang, and Xiaobo Qu. 2023. Can language models be used for real-world urban-delivery route optimization? *The Innovation* 4, 6 (2023), 100520. DOI: 10.1016/j.xinn.2023.100520

[132] AI at Meta Llama Team. 2024. The llama 3 herd of models. arXiv:2407.21783. Retrieved from https://arxiv.org/abs/2407.21783 (2024).

[133] Mariana A. Londe, Luciana S. Pessoa, Carlos E. Andrade, and Mauricio G. C. Resende. 2024. Biased random-key genetic algorithms: A review. *European Journal of Operational Research* 318, 2 (2024), 22 pages. DOI: 10.1016/j.ejor.2024.03.030

[134] Sifan Long, Jingjing Tan, Bomin Mao, Fengxiao Tang, Yangfan Li, Ming Zhao, and Nei Kato. 2025. A survey on intelligent network operations and performance optimization based on large language models. *IEEE Communications Surveys & Tutorials* 27, 6 (2025), 3915–3949. DOI: 10.1109/COMST.2025.3526606

[135] Maria Amélia Lopes Silva, Sérgio Ricardo de Souza, Marcone Jamilson Freitas Souza, and Moacir Felizardo de França Filho. 2018. Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis. *Applied Soft Computing* 71 (2018), 433–459. DOI: 10.1016/j.asoc.2018.06.050

[136] Anton Lozhkov, Raymond Li, Loubna Ben Allal, Federico Cassano, Joel Lamy-Poirier, Nouamane Tazi, Ao Tang, Dmytro Pykhtar, Jiawei Liu, Yuxiang Wei, et al. 2024. StarCoder 2 and The Stack v2: The next generation. arXiv:2402.19173. Retrieved from https://arxiv.org/abs/2402.19173 (2024).

[137] Zihan Luo, Xiran Song, Hong Huang, Jianxun Lian, Chenhao Zhang, Jinqi Jiang, and Xing Xie. 2024. GraphInstruct: Empowering large language models with graph understanding and reasoning capability. arXiv:2403.04483. Retrieved from https://arxiv.org/abs/2403.04483 (2024).

[138] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Mauro Birattari, and Thomas Stützle. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives* 3 (2016), 43–58. DOI: 10.1016/j.orp.2016.09.002

[139] Paula Maddigan and Teo Susnjak. 2023. Chat2VIS: Generating data visualizations via natural language using ChatGPT, codex and GPT-3 large language models. *IEEE Access* 11 (2023), 45181–45193. DOI: 10.1109/ACCESS.2023.3274199

[140] Matteo Manica, Jannis Born, Joris Cadow, Dimitrios Christofidellis, Ashish Dave, Dean Clarke, Yves Gaetan Nana Teukam, Giorgio Giannone, Samuel C. Hoffman, Matthew Buchan, et al. 2023. Accelerating material design with the generative toolkit for scientific discovery. *npj Computational Materials* 9, 1 (1 5 2023), 69. DOI: 10.1038/s41524-023-01028-1

[141] James Manyika and Sissie Hsiao. 2023. An Overview of Bard: An Early Experiment with Generative AI. AI. Google Static Documents. Retrieved June 27, 2024 from https://ai.google/static/documents/google-about-bard.pdf

[142] Jinzhu Mao, Dongyun Zou, Li Sheng, Siyi Liu, Chen Gao, Yue Wang, and Yong Li. 2024. Identify critical nodes in complex network with large language models. arXiv:2403.03962. Retrieved from https://arxiv.org/abs/2403.03962 (2024).

[143] Alicja Martinek, Szymon Łukasik, and Amir H. Gandomi. 2024. Large language models as tuning agents of metaheuristics. In *Proceedings of the 32nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2024)*. i6doc.com, Bruges, Belgium, 631–636. DOI: 10.14428/ESANN/2024.ES2024-209

[144] Rafael Martí, Marc Sevaux, and Kenneth Sörensen. 2024. Fifty years of metaheuristics. *European Journal of Operational Research* 318, 2 (2024), 18 pages. DOI: 10.1016/j.ejor.2024.04.004

[145] Jordan Meadows and Andre Freitas. 2024. A survey in mathematical language processing. arXiv:2205.15231. Retrieved from https://arxiv.org/abs/2205.15231 (2024).

[146] Kostis Michailidis, Dimos Tsouros, and Tias Guns. 2024. Constraint modelling with LLMs using in-context learning. In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 307)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 20:1–20:27. DOI: 10.4230/LIPIcs.CP.2024.20

[147] Kostis Michailidis, Dimos Tsouros, and Tias Guns. 2026. DCP-bench-open: Evaluating LLMs for constraint modelling of discrete combinatorial problems. arXiv:2506.06052. Retrieved from https://arxiv.org/abs/2506.06052 (2026).

[148] Kangtong Mo, Wenyan Liu, Fangzhou Shen, Xuanzhen Xu, Letian Xu, Xiran Su, and Ye Zhang. 2024. Precision kinematic path optimization for high-dof robotic manipulators utilizing advanced natural language processing models. In *Proceedings of the 2024 5th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*. IEEE, Shenzhen, China, 649–654. DOI: 10.1109/ICECAI62591.2024.10675146

[149] David Moher, Deborah J. Cook, Susan Eastwood, Ingram Olkinm, Drummond Rennie, and Donna F. Stroup. 1999. Improving the quality of reports of meta-analyses of randomised controlled trials: The QUOROM statement. *The Lancet* 354, 9193 (1999), 1896–1900. DOI: 10.1016/S0140-6736(99)04149-5

[150] David Moher, Alessandro Liberati, Jennifer Tetzlaff, Douglas G. Altman, and The PRISMA Group. 2009. Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *PLOS Medicine* 6, 7 (07 2009), 1–6. DOI: 10.1371/journal.pmed.1000097

[151] Maximilian Moser, Nysret Musliu, Andrea Schaerf, and Felix Winter. 2022. Exact and metaheuristic approaches for unrelated parallel machine scheduling. *Journal of Scheduling* 25, 5 (2022), 507–534. DOI: 10.1007/s10951-021-00714-6

[152] Mahdi Mostajabdaveh, Timothy T. Yu, Samarendra Chandan Bindu Dash, Rindranirina Ramamonjison, Jabo Serge Byusa, Giuseppe Carenini, Zirui Zhou, and Yong Zhang. 2024. Evaluating LLM reasoning in the operations research domain with ORQA. arXiv:2412.17874. Retrieved from https://arxiv.org/abs/2412.17874 (2024).

[153] Mahdi Mostajabdaveh, Timothy T. Yu, Rindranirina Ramamonjison, Giuseppe Carenini, Zirui Zhou, and Yong Zhang. 2024. Optimization modeling and verification from problem specifications using a multi-agent multi-stage LLM framework. *INFOR: Information Systems and Operational Research* 62, 4 (2024), 599–617. DOI: 10.1080/03155986.2024.2381306

[154] Yves Gaetan Nana Teukam, Federico Zipoli, Teodoro Laino, Emanuele Criscuolo, Francesca Grisoni, and Matteo Manica. 2024. Integrating genetic algorithms and language models for enhanced enzyme design. *Briefings in Bioinformatics* 26, 1 (Nov. 2024), bbae675. DOI: 10.1093/bib/bbae675

[155] Lise-Marie Nassen, Heidi Vandebosch, Karolien Poels, and Kathrin Karsay. 2023. Opt-out, abstain, unplug. A systematic review of the voluntary digital disconnection literature. *Telematics and Informatics* 81 (2023), 101980. DOI: 10.1016/j.tele.2023.101980

[156] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. 2007. MiniZinc: Towards a standard CP modelling language. In *Principles and Practice of Constraint Programming – CP 2007*. Springer Berlin 529–543. DOI: 10.1007/978-3-540-74970-7_38

[157] Yuting Ning, Jiayu Liu, Longhu Qin, Tong Xiao, Shangzi Xue, Zhenya Huang, Qi Liu, Enhong Chen, and Jinze Wu. 2023. A novel approach for auto-formulation of optimization problems. arXiv:2302.04643. Retrieved from https://arxiv.org/abs/2302.04643 (2023).

[158] Kazuma Obata, Tatsuya Aoki, Takato Horii, Tadahiro Taniguchi, and Takayuki Nagai. 2025. LiP-LLM: Integrating linear programming and dependency graph with large language models for multi-robot task planning. *IEEE Robotics and Automation Letters* 10, 2 (2025), 1122–1129. DOI: 10.1109/LRA.2024.3518105

[159] Object Management Group. 2011. Business Process Model and Notation (BPMN), Version 2.0. https://www.omg.org/spec/BPMN/2.0/PDF Retrieved from the Object Management Group website.

[160] Gabriela Ochoa, Katherine M. Malan, and Christian Blum. 2021. Search trajectory networks: A tool for analysing and visualising the behaviour of metaheuristics. *Applied Soft Computing* 109 (2021), 107492. DOI: 10.1016/j.asoc.2021.107492

[161] OpenAI. 2024. GPT-4 technical report. arXiv:2303.08774. Retrieved from https://arxiv.org/abs/2303.08774 (2024).

[162] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf

[163] Matthew J. Page, Joanne E. McKenzie, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, et al. 2021. The PRISMA 2020 statement: An updated guideline for reporting systematic reviews. *BMJ* 372 (2021), 9 pages. DOI: 10.1136/bmj.n71

[164] Matthew J. Page, David Moher, Patrick M. Bossuyt, Isabelle Boutron, Tammy C. Hoffmann, Cynthia D. Mulrow, Larissa Shamseer, Jennifer M. Tetzlaff, Elie A. Akl, Sue E. Brennan, et al. 2021. PRISMA 2020 explanation and elaboration: Updated guidance and exemplars for reporting systematic reviews. *BMJ* 372 (2021), 36 pages. DOI: 10.1136/bmj.n160

[165] Federico Pagnozzi and Thomas Stützle. 2021. Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems with additional constraints. *Operations Research Perspectives* 8 (2021), 100180. DOI: 10.1016/j.orp.2021.100180

[166] Vishal Pallagani, Bharath Chandra Muppasani, Kaushik Roy, Francesco Fabiano, Andrea Loreggia, Keerthiram Murugesan, Biplav Srivastava, Francesca Rossi, Lior Horesh, and Amit Sheth. 2025. On the prospects of incorporating large language models (LLMs) in automated planning and scheduling (APS). In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS '24)*. AAAI Press, Alberta, Canada, Article 50, 13 pages. DOI: 10.1609/icaps.v34i1.31503

[167] Rui Pan, Shuo Xing, Shizhe Diao, Wenhe Sun, Xiang Liu, KaShun Shum, Jipeng Zhang, Renjie Pi, and Tong Zhang. 2024. Plum: Prompt learning using metaheuristics. In *Findings of the ACL: ACL 2024*. ACL, Bangkok, Thailand, 2177–2197. DOI: 10.18653/v1/2024.findings-acl.129

[168] José Antonio Parejo, Antonio Ruiz-Cortés, Sebastián Lozano, and Pablo Fernandez. 2012. Metaheuristic optimization frameworks: A survey and benchmarking. *Soft Computing* 16, 3 (mar 2012), 527–561. DOI: 10.1007/s00500-011-0754-8

[169] Laurent Perron and Vincent Furnon. 2024. *OR-Tools*. Google. Retrieved March 9, 2026 from https://developers.google.com/optimization/

[170] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the ACL: Human Language Technologies, Volume 1 (Long Papers)*. ACL, New Orleans, Louisiana, 2227–2237. DOI: 10.18653/v1/N18-1202

[171] Michal Pluhacek, Anezka Kazikova, Tomas Kadavy, Adam Viktorin, and Roman Senkerik. 2023. Leveraging large language models for the generation of novel metaheuristic optimization algorithms. In *Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (Lisbon, Portugal) *(GECCO '23 Companion)*. ACM, New York, NY, USA, 1812–1820. DOI: 10.1145/3583133.3596401

[172] Michal Pluhacek, Anezka Kazikova, Adam Viktorin, Tomas Kadavy, and Roman Senkerik. 2023. Investigating the potential of AI-driven innovations for enhancing differential evolution in optimization tasks. In *Proceedings of the 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, Honolulu, Oahu, HI, USA, 1070–1075. DOI: 10.1109/SMC53992.2023.10394233

[173] Michal Pluhacek, Jozef Kovac, Adam Viktorin, Peter Janku, Tomas Kadavy, and Roman Senkerik. 2024. Using LLM for automatic evolvement of metaheuristics from swarm algorithm SOMA. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (Melbourne, VIC, Australia) *(GECCO '24 Companion)*. ACM, New York, NY, USA, 2018–2022. DOI: 10.1145/3638530.3664181

[174] Petrică C. Pop, Ovidiu Cosma, Cosmin Sabo, and Corina Pop Sitar. 2024. A comprehensive survey on the generalized traveling salesman problem. *European Journal of Operational Research* 314, 3 (2024), 819–835. DOI: 10.1016/j.ejor.2023.07.022

[175] Alibaba Group Qwen Team. 2023. Qwen technical report. arXiv:2309.16609. Retrieved from https://arxiv.org/abs/2309.16609 (2023).

[176] Alibaba Group Qwen Team. 2024. Qwen2 technical report. arXiv:2407.10671. Retrieved from https://arxiv.org/abs/2407.10671 (2024).

[177] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. arXiv:2305.18290. Retrieved from https://arxiv.org/abs/2305.18290 (2024).

[178] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J. Liu, et al. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. arXiv:1910.10683. Retrieved from https://arxiv.org/abs/1910.10683 (2023).

[179] Rindra Ramamonjison, Haley Li, Timothy Yu, Shiqi He, Vishnu Rengan, Amin Banitalebi-dehkordi, Zirui Zhou, and Yong Zhang. 2022. Augmenting operations research with auto-formulation of optimization models from problem descriptions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*. ACL, Abu Dhabi, UAE, 29–62. DOI: 10.18653/v1/2022.emnlp-industry.4

[180] Rindranirina Ramamonjison, Timothy T. Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, et al. 2023. NL4Opt competition: Formulating optimization problems based on their natural language descriptions. arXiv:2303.08233. Retrieved from https://arxiv.org/abs/2303.08233 (2023).

[181] Florian Régin, Elisabetta De Maria, and Alexandre Bonlarron. 2024. Combining constraint programming reasoning with large language model predictions. In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP 2024) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 307)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 25:1–25:18. DOI: 10.4230/LIPIcs.CP.2024.25

[182] Wesley F. Reinhart and Antonia Statt. 2024. Large language models design sequence-defined macromolecules via evolutionary optimization. *NPJ Computational Materials* 10, 1 (2024), 262. DOI: 10.1038/s41524-024-01449-6

[183] Oleksandr Romanko, Akhilesh Narayan, and Roy H. Kwon. 2023. ChatGPT-based investment portfolio selection. *SN Operations Research Forum* 4, 4 (December 2023), 1–27. DOI: 10.1007/s43069-023-00277-6

[184] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, et al. 2024. Mathematical discoveries from program search with large language models. *Nature* 625, 7995 (1 1 2024), 468–475. DOI: 10.1038/s41586-023-06924-6

[185] Jamie Ross, Fiona Stevenson, Rosa Lau, and Elizabeth Murray. 2016. Factors that influence the implementation of e-health: A systematic review of systematic reviews (an update). *Implementation Science* 11, 1 (26 Oct 2016), 146. DOI: 10.1186/s13012-016-0510-7

[186] Francesca Rossi, Peter Van Beek, and Toby Walsh. 2006. *Handbook of Constraint Programming*. Elsevier, Amsterdam. Retrieved from https://www.dcs.gla.ac.uk/~pat/cpM/papers/CP_Handbook-20060315-final.pdf

[187] Abdullahi Saka, Ridwan Taiwo, Nurudeen Saka, Babatunde Abiodun Salami, Saheed Ajayi, Kabiru Akande, and Hadi Kazemi. 2024. GPT models in construction industry: Opportunities, limitations, and a use case validation. *Developments in the Built Environment* 17 (2024), 100300. DOI: 10.1016/j.dibe.2023.100300

[188] Camilo Chacón Sartori, Christian Blum, Filippo Bistaffa, and Guillem Rodríguez Corominas. 2025. Metaheuristics and large language models join forces: Toward an integrated optimization approach. *IEEE Access* 13 (2025), 2058–2079. DOI: 10.1109/ACCESS.2024.3524176

[189] Max Schäfer, Sarah Nadi, Aryaz Eghbali, and Frank Tip. 2024. An empirical evaluation of using large language models for automated unit test generation. *IEEE Transactions on Software Engineering* 50, 1 (2024), 85–105. DOI: 10.1109/TSE.2023.3334955

[190] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, et al. 2024. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. arXiv:2402.03300. Retrieved from https://arxiv.org/abs/2402.03300 (2024).

[191] Paul Shaw. 1998. Principles and practice of constraint programming — CP98. In *Proceedings of the 4th International Conference, CP98, Pisa, Italy, October 26-30, 1998: Proceedings (Lecture Notes in Computer Science, Vol. 1520)*. Springer, Berlin, 480 pages. DOI: 10.1007/3-540-49481-2

[192] Akash Singirikonda, Serdar Kadioglu, and Karthik Uppuluri. 2025. Text2Zinc: A cross-domain dataset for modeling optimization and satisfaction problems in MiniZinc. arXiv:2503.10642. Retrieved from https://arxiv.org/abs/2503.10642 (2025).

[193] Aditi Singla, Aditya Singh, and Kanishk Kukreja. 2023. A bi-objective $\epsilon$-constrained framework for quality-cost optimization in language model ensembles. arXiv:2312.16119. Retrieved from https://arxiv.org/abs/2312.16119 (2023).

[194] Michael Soprano, Kevin Roitero, David La Barbera, Davide Ceolin, Damiano Spina, Gianluca Demartini, and Stefano Mizzaro. 2024. Cognitive biases in fact-checking and their countermeasures: A review. *Information Processing & Management* 61, 3 (2024), 103672. DOI: 10.1016/j.ipm.2024.103672

[195] Kenneth Sörensen, Marc Sevaux, and Fred Glover. 2018. *A History of Metaheuristics*. Springer International Publishing, Cham, 791–808. DOI: 10.1007/978-3-319-07124-4_4

[196] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv:2206.04615. Retrieved from https://arxiv.org/abs/2206.04615 (2022).

[197] Biplav Srivastava and Vishal Pallagani. 2024. The case for developing a foundation model for planning-like tasks from scratch. arXiv:2404.04540. Retrieved from https://arxiv.org/abs/2404.04540 (2024).

[198] Elmar Steiner, Ulrich Pferschy, and Andrea Schaerf. 2024. Curriculum-based university course timetabling considering individual course of studies. *Central European Journal of Operations Research* 32, 3 (21 6 2024), 38 pages. DOI: 10.1007/s10100-024-00923-2

[199] Jingyan Sui, Shizhe Ding, Xulin Huang, Yue Yu, Ruizhi Liu, Boyang Xia, Zhenxin Ding, Liming Xu, Haicang Zhang, Chungong Yu, et al. 2024. A survey on deep learning-based algorithms for the traveling salesman problem. *Frontiers of Computer Science* 19, 6 (2024), 196322. DOI: 10.1007/s11704-024-40490-y

[200] Yiwen Sun, Furong Ye, Xianyin Zhang, Shiyu Huang, Bingzhen Zhang, Ke Wei, and Shaowei Cai. 2024. AutoSAT: Automatically optimize SAT solvers via large language models. arXiv:2402.10705. Retrieved from https://arxiv.org/abs/2402.10705 (2024).

[201] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, et al. 2022. Challenging BIG-bench tasks and whether chain-of-thought can solve them. arXiv:2210.09261. Retrieved from https://arxiv.org/abs/2210.09261 (2022).

[202] Jerry Swan, Steven Adriaensen, Mohamed Bishr, Edmund K. Burke, John A. Clark, Patrick De Causmaecker, Juanjo Durillo, Kevin Hammond, Emma Hart, Colin G. Johnson, et al. 2015. A research agenda for metaheuristic standardization. In *Proceedings of the 11th Metaheuristics International Conference (MIC 2015)*. University of Nottingham, Agadir, Morocco, 1–3. DOI: 10.1007/978-3-031-62912-9

[203] Jerry Swan, Steven Adriaensen, Alexander E. I. Brownlee, Kevin Hammond, Colin G. Johnson, Ahmed Kheiri, Faustyna Krawiec, J. J. Merelo, Leandro L. Minku, Ender Özcan, et al. 2022. Metaheuristics "In the Large". *European Journal of Operational Research* 297, 2 (2022), 393–406. DOI: 10.1016/j.ejor.2021.05.042

[204] Jerry Swan, Steven Adriænsen, Adam D. Barwell, Kevin Hammond, and David R. White. 2019. Extending the "Open-closed principle" to automated algorithm configuration. *Evolutionary Computation* 27, 1 (03 2019), 173–193. DOI: 10.1162/evco_a_00245

[205] Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. arXiv:2403.05530. Retrieved from https://arxiv.org/abs/2403.05530 (2024).

[206] Gemini Team. 2024. Gemini: A family of highly capable multimodal models. arXiv:2312.11805. Retrieved from https://arxiv.org/abs/2312.11805 (2024).

[207] Gemma Team. 2024. Gemma 2: Improving open language models at a practical size. arXiv:2408.00118. Retrieved from https://arxiv.org/abs/2408.00118 (2024).

[208] Meta AI Team. 2024. Code llama: Open foundation models for code. arXiv:2308.12950. Retrieved from https://arxiv.org/abs/2308.12950 (2024).

[209] Yi Team. 2024. Yi: Open foundation models by 01.AI. arXiv:2403.04652. Retrieved from https://arxiv.org/abs/2403.04652 (2024).

[210] Nguyen Van Thieu and Seyedali Mirjalili. 2023. MEALPY: An open-source library for latest meta-heuristic algorithms in python. *Journal of Systems Architecture* 139 (2023), 102871. DOI: 10.1016/j.sysarc.2023.102871

[211] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. LLaMA: Open and efficient foundation language models. arXiv:2302.13971. Retrieved from https://arxiv.org/abs/2302.13971 (2023).

[212] Thanh V. T. Tran and Truong Son Hy. 2024. Protein design by directed evolution guided by large language models. *IEEE Transactions on Evolutionary Computation* 29, 2 (2024), 418–428. DOI: 10.1109/TEVC.2024.3439690

[213] Dimos Tsouros, Hélène Verhaeghe, Serdar Kadıoğlu, and Tias Guns. 2023. Holy Grail 2.0: From natural language to constraint models. arXiv:2308.01589. Retrieved from https://arxiv.org/abs/2308.01589 (2023).

[214] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of LM alignment. arXiv:2310.16944. Retrieved from https://arxiv.org/abs/2310.16944 (2023).

[215] Jose Tupayachi, Haowen Xu, Olufemi A. Omitaomu, Mustafa Can Camur, Aliza Sharmin, and Xueping Li. 2024. Towards next-generation urban decision support systems through AI-powered construction of scientific ontology using large language models—a case in optimizing intermodal freight transportation. *Smart Cities* 7, 5 (2024), 2392–2421. DOI: 10.3390/smartcities7050094

[216] María Cora Urdaneta-Ponte, Amaia Mendez-Zorrilla, and Ibon Oleagordia-Ruiz. 2021. Recommendation systems for education: Systematic review. *Electronics* 10, 14 (2021), 21 pages. DOI: 10.3390/electronics10141611

[217] Vyacheslav Ustyugov. 2024. On different methods for automated MILP solver configuration. In *2024 20th International Asian School-Seminar on Optimization Problems of Complex Systems (OPCS)*. IEEE, Issyk-Kul Lake, Kyrgyzstan, 24–27. DOI: 10.1109/OPCS63516.2024.10720437

[218] Niki van Stein and Thomas Bäck. 2024. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation* 29, 2 (2024), 331–345. DOI: 10.1109/TEVC.2024.3497793

[219] Niki van Stein, Diederick Vermetten, and Thomas Bäck. 2024. In-the-loop hyper-parameter optimization for LLM-based automated design of heuristics. arXiv:2410.16309. Retrieved from https://arxiv.org/abs/2410.16309 (2024).

[220] Johannes Vass, Marie-Louise Lackner, Christoph Mrkvicka, Nysret Musliu, and Felix Winter. 2022. Exact and meta-heuristic approaches for the production leveling problem. *Journal of Scheduling* 25, 3 (01 Jun 2022), 339–370. DOI: 10.1007/s10951-022-00721-1

[221] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All You Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., Long Beach, CA, USA, 5998–6008. DOI: 10.5555/3295222.3295349

[222] Kristian Verduin, Thomas Weise, and Daan van den Berg. 2023. Why is the traveling tournament problem not solved with genetic algorithms?. In *Evo* 2023 – Late-Breaking Abstracts Volume*. Species, Brno, Czech Republic, 13–18. Retrieved from https://arxiv.org/pdf/2403.13950

[223] Florentina Voboril, Vaidyanathan Peruvemba Ramaswamy, and Stefan Szeider. 2024. Generating streamlining constraints with large language models. arXiv:2408.10268. Retrieved from https://arxiv.org/abs/2408.10268 (2024).

[224] Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language?. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) *(NIPS '23)*. Curran Associates Inc., Red Hook, NY, USA, Article 1345, 22 pages. DOI: 10.5555/3666122.3667467

[225] Kangxu Wang, Ze Chen, and Jiewen Zheng. 2023. OPD@NL4Opt: An ensemble approach for the NER task of the optimization problem. arXiv:2301.02459. Retrieved from https://arxiv.org/abs/2301.02459 (2023).

[226] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* 18, 6 (March 2024), 42 pages. DOI: 10.1007/s11704-024-40231-1

[227] Yuhui Wang, Junaid Farooq, Hakim Ghazzai, and Gianluca Setti. 2024. Multi-UAV Placement for Integrated Access and Backhauling Using LLM-Driven Optimization. TechRxiv. DOI: 10.36227/techrxiv.172833400.07230719/v1

[228] Yuan Wang, Lokesh Kumar Sambasivan, Mingang Fu, and Prakhar Mehrotra. 2024. Pivoting retail supply chain with deep generative techniques: Taxonomy, survey and insights. arXiv:2403.00861. Retrieved from https://arxiv.org/abs/2403.00861 (2024).

[229] Segev Wasserkrug, Leonard Boussioux, Dick den Hertog, Farzaneh Mirzazadeh, Ilker Birbil, Jannis Kurtz, and Donato Maragno. 2024. From large language models and optimization to decision optimization CoPilot: A research manifesto. arXiv:2402.16269. Retrieved from https://arxiv.org/abs/2402.16269 (2024).

[230] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022. Emergent abilities of large language models. arXiv:2206.07682. Retrieved from https://arxiv.org/abs/2206.07682 (2022).

[231] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Richter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, Vol. 35. Curran Associates, Inc., Virtual Conference, 24824–24837. Retrieved from https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[232] Setyo Tri Windras Mara, Rachmadi Norcahyo, Panca Jodiawan, Luluk Lusiantoro, and Achmad Pratama Rifai. 2022. A survey of adaptive large neighborhood search algorithms and applications. *Computers & Operations Research* 146 (2022), 105903. DOI: 10.1016/j.cor.2022.105903

[233] Felix Winter, Nysret Musliu, Emir Demirović, and Christoph Mrkvicka. 2019. Solution approaches for an automotive paint shop scheduling problem. In *Proceedings of the International Conference on Automated Planning and Scheduling*, Vol. 29. AAAI Press, Virtual Conference, 573–581. DOI: 10.1609/icaps.v29i1.3524

[234] D. H. Wolpert and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82. DOI: 10.1109/4235.585893

[235] Chao Wu, Yiling Ge, Xinyan Zhang, Yanling Du, Shizhe He, Zhaohua Ji, and Hongjuan Lang. 2021. The combined effects of Lamaze breathing training and nursing intervention on the delivery in primipara: A PRISMA systematic review meta-analysis. *Medicine* 100, 4 (2021), 8 pages. DOI: 10.1097/MD.0000000000023920

[236] Xuan Wu, Di Wang, Lijie Wen, Yubin Xiao, Chunguo Wu, Yuesong Wu, Chaoyu Yu, Douglas L. Maskell, and You Zhou. 2024. Neural combinatorial optimization algorithms for solving vehicle routing problems: A comprehensive survey with perspectives. arXiv:2406.00415. Retrieved from https://arxiv.org/abs/2406.00415 (2024).

[237] Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. 2024. Evolutionary computation in the era of large language model: Survey and roadmap. arXiv:2401.10034. Retrieved from https://arxiv.org/abs/2401.10034 (2024).

[238] Xingyu Wu, Yan Zhong, Jibin Wu, Bingbing Jiang, and Kay Chen Tan. 2024. Large language model-enhanced algorithm selection: Towards comprehensive algorithm representation. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI '24)*. International Joint Conferences on Artificial Intelligence, Jeju Island, South Korea, 10 pages. DOI: 10.24963/ijcai.2024/579

[239] Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, et al. 2024. Chain-of-experts: When LLMs meet complex operations research problems. In *Proceedings of the 12th International Conference on Learning Representations (ICLR 2024)*. OpenReview, Virtual Conference, 19 pages. Retrieved from https://openreview.net/forum?id=HobyL1B9CZ

[240] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. arXiv:2309.03409. Retrieved from https://arxiv.org/abs/2309.03409 (2024).

[241] Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. 2024. OptiBench meets ReSocratic: Measure and improve LLMs for optimization modeling. arXiv:2407.09887. Retrieved from https://arxiv.org/abs/2407.09887 (2024)

[242] Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. 2024. Multi-objective evolution of heuristic using large language model. arXiv:2409.16867. Retrieved from https://arxiv.org/abs/2409.16867 (2024).

[243] Wang Yatong, Pei Yuchen, and Zhao Yuqi. 2024. TS-EoH: An edge server task scheduling algorithm based on evolution of heuristic. arXiv:2409.09063. Retrieved from https://arxiv.org/abs/2409.09063 (2024).

[244] Haoran Ye, Jiarui Wang, Zhiguang Cao, Federico Berto, Chuanbo Hua, Haeyeon Kim, Jinkyoo Park, and Guojie Song. 2024. ReEvo: Large language models as hyper-heuristics with reflective evolution. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems (NeurIPS 2024)*. Neural Information Processing Systems Foundation, Vancouver, Canada, 1–32. Retrieved from https://openreview.net/forum?id=483IPG0HWL

[245] Hengxu You, Yang Ye, Tianyu Zhou, Qi Zhu, and Jing Du. 2023. Robot-enabled construction assembly with automated sequence planning based on ChatGPT: RoboGPT. *Buildings* 13, 7 (2023), 24 pages. DOI: 10.3390/buildings13071772

[246] He Yu and Jing Liu. 2024. AutoRNet: Automatically optimizing heuristics for robust network design via large language models. arXiv:2410.17656. Retrieved from https://arxiv.org/abs/2410.17656 (2024).

[247] He Yu and Jing Liu. 2024. Deep insights into automated optimization with large language models and evolutionary algorithms. arXiv:2410.20848. Retrieved from https://arxiv.org/abs/2410.20848 (2024).

[248] Eugenia Zanazzo, Sara Ceschia, and Andrea Schaerf. 2024. Solving the integrated patient-to-room and nurse-to-patient assignment by simulated annealing. In *Proceedings of the Metaheuristics: 15th International Conference, MIC 2024, Lorient, France, June 4–7, 2024, Proceedings, Part I* (Lorient, France). Springer-Verlag, Berlin, 158–163. DOI: 10.1007/978-3-031-62912-9_15

[249] Jihai Zhang, Wei Wang, Siyan Guo, Li Wang, Fangquan Lin, Cheng Yang, and Wotao Yin. 2024. Solving general natural-language-description optimization problems with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the ACL: Human Language Technologies (Volume 6: Industry Track)*. ACL, Mexico City, Mexico, 483–490. DOI: 10.18653/v1/2024.naacl-industry.42

[250] Mengyuan Zhang, Wotao Yin, Mengchang Wang, Yangbin Shen, Peng Xiang, You Wu, Liang Zhao, Junqiu Pan, Hu Jiang, and KuoLing Huang. 2023. MindOpt tuner: Boost the performance of numerical software by automatic parameter tuning. arXiv:2307.08085. Retrieved from https://arxiv.org/abs/2307.08085 (2023).

[251] Rui Zhang, Fei Liu, Xi Lin, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. 2024. Understanding the importance of evolutionary search in automated heuristic design with large language models. In *Parallel Problem Solving from Nature – PPSN XVIII*. Springer Nature Switzerland, Cham, 185–202. DOI: 10.1007/978-3-031-70068-2_12

[252] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating text generation with BERT. arXiv:1904.09675. Retrieved from https://arxiv.org/abs/1904.09675 (2020).

[253] Zeyang Zhang, Xin Wang, Ziwei Zhang, Haoyang Li, Yijian Qin, and Wenwu Zhu. 2024. LLM4DyG: Can large language models solve spatial-temporal problems on dynamic graphs? arXiv:2310.17110. Retrieved from https://arxiv.org/abs/2310.17110 (2024).

[254] Zhigen Zhao, Shuo Cheng, Yan Ding, Ziyi Zhou, Shiqi Zhang, Danfei Xu, and Ye Zhao. 2024. A survey of optimization-based task and motion planning: From classical to learning approaches. *IEEE/ASME Transactions on Mechatronics* 29, 5 (2024), 1–27. DOI: 10.1109/TMECH.2024.3452509

[255] Hatice Çalık, Tony Wauters, and Greet Vanden Berghe. 2024. The exam location problem: Mathematical formulations and variants. *Computers & Operations Research* 161 (2024), 106438. DOI: 10.1016/j.cor.2023.106438

# Appendices

## A PRISMA Checklists

This appendix provides the "PRISMA 2020 for Abstracts Checklist" and the "PRISMA 2020 Checklist" [163, 164], which were used to guide our systematic review on the applications of LLMs in the field of Combinatorial Optimization (Section 5).

**PRISMA 2020 for Abstracts Checklist**

| Section and Topic | Item # | Checklist item | Reported (Yes/No) |
|---|---|---|---|
| **TITLE** | | | |
| Title | 1 | Identify the report as a systematic review. | Yes |
| **BACKGROUND** | | | |
| Objectives | 2 | Provide an explicit statement of the main objective(s) or question(s) the review addresses. | Yes |
| **METHODS** | | | |
| Eligibility criteria | 3 | Specify the inclusion and exclusion criteria for the review. | Yes |
| Information sources | 4 | Specify the information sources (e.g. databases, registers) used to identify studies and the date when each was last searched. | Yes |
| Risk of bias | 5 | Specify the methods used to assess risk of bias in the included studies. | No |
| Synthesis of results | 6 | Specify the methods used to present and synthesise results. | Yes |
| **RESULTS** | | | |
| Included studies | 7 | Give the total number of included studies and participants and summarise relevant characteristics of studies. | Yes |
| Synthesis of results | 8 | Present results for main outcomes, preferably indicating the number of included studies and participants for each. If meta-analysis was done, report the summary estimate and confidence/credible interval. If comparing groups, indicate the direction of the effect (i.e. which group is favoured). | Yes |
| **DISCUSSION** | | | |
| Limitations of evidence | 9 | Provide a brief summary of the limitations of the evidence included in the review (e.g. study risk of bias, inconsistency and imprecision). | No |
| Interpretation | 10 | Provide a general interpretation of the results and important implications. | Yes |
| **OTHER** | | | |
| Funding | 11 | Specify the primary source of funding for the review. | Within paper body |
| Registration | 12 | Provide the register name and registration number. | Not relevant |

*From:* Page MJ, McKenzie JE, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ 2021;372:n71. doi: 10.1136/bmj.n71

For more information, visit: http://www.prisma-statement.org/

**PRISMA 2020 Checklist**

| Section and Topic | Item # | Checklist item | Location where item is reported |
|---|---|---|---|
| **TITLE** | | | |
| Title | 1 | Identify the report as a systematic review. | Title |
| **ABSTRACT** | | | |
| Abstract | 2 | See the PRISMA 2020 for Abstracts checklist. | Abstract, see PRISMA for Abstracts checklist |
| **INTRODUCTION** | | | |
| Rationale | 3 | Describe the rationale for the review in the context of existing knowledge. | Section 1 |
| Objectives | 4 | Provide an explicit statement of the objective(s) or question(s) the review addresses. | Section 2 |
| **METHODS** | | | |
| Eligibility criteria | 5 | Specify the inclusion and exclusion criteria for the review and how studies were grouped for the syntheses. | Section 5.3.3 |
| Information sources | 6 | Specify all databases, registers, websites, organisations, reference lists and other sources searched or consulted to identify studies. Specify the date when each source was last searched or consulted. | Section 5.3.2 |
| Search strategy | 7 | Present the full search strategies for all databases, registers and websites, including any filters and limits used. | Section 5.3.2 |
| Selection process | 8 | Specify the methods used to decide whether a study met the inclusion criteria of the review, including how many reviewers screened each record and each report retrieved, whether they worked independently, and if applicable, details of automation tools used in the process. | Section 5.3.3 |
| Data collection process | 9 | Specify the methods used to collect data from reports, including how many reviewers collected data from each report, whether they worked independently, any processes for obtaining or confirming data from study investigators, and if applicable, details of automation tools used in the process. | Section 5.3.4 |
| Data items | 10a | List and define all outcomes for which data were sought. Specify whether all results that were compatible with each outcome domain in each study were sought (e.g. for all measures, time points, analyses), and if not, the methods used to decide which results to collect. | Section 6 and Appendix B |
| | 10b | List and define all other variables for which data were sought (e.g. participant and intervention characteristics, funding sources). Describe any assumptions made about any missing or unclear information. | Section 6 and Appendix B |
| Study risk of bias assessment | 11 | Specify the methods used to assess risk of bias in the included studies, including details of the tool(s) used, how many reviewers assessed each study and whether they worked independently, and if applicable, details of automation tools used in the process. | Section 5.3 |
| Effect measures | 12 | Specify for each outcome the effect measure(s) (e.g. risk ratio, mean difference) used in the synthesis or presentation of results. | Not relevant |
| Synthesis methods | 13a | Describe the processes used to decide which studies were eligible for each synthesis (e.g. tabulating the study intervention characteristics and comparing against the planned groups for each synthesis (item #5)). | Section 5.3.3 |
| | 13b | Describe any methods required to prepare the data for presentation or synthesis, such as handling of missing summary statistics, or data conversions. | Not relevant |
| | 13c | Describe any methods used to tabulate or visually display results of individual studies and syntheses. | Section 6 |
| | 13d | Describe any methods used to synthesize results and provide a rationale for the choice(s). If meta-analysis was performed, describe the model(s), method(s) to identify the presence and extent of statistical heterogeneity, and software package(s) used. | Section 6 |
| | 13e | Describe any methods used to explore possible causes of heterogeneity among study results (e.g. subgroup analysis, meta-regression). | Not relevant |
| | 13f | Describe any sensitivity analyses conducted to assess robustness of the synthesized results. | Not relevant |
| Reporting bias assessment | 14 | Describe any methods used to assess risk of bias due to missing results in a synthesis (arising from reporting biases). | Section 8 |
| Certainty assessment | 15 | Describe any methods used to assess certainty (or confidence) in the body of evidence for an outcome. | Not relevant |
| **RESULTS** | | | |
| Study selection | 16a | Describe the results of the search and selection process, from the number of records identified in the search to the number of studies included in the review, ideally using a flow diagram. | Figure 2 |

**PRISMA 2020 Checklist**

| Section and Topic | Item # | Checklist item | Location where item is reported |
|---|---|---|---|
| | 16b | Cite studies that might appear to meet the inclusion criteria, but which were excluded, and explain why they were excluded. | Section 5.3.4 |
| Study characteristics | 17 | Cite each included study and present its characteristics. | Appendix B |
| Risk of bias in studies | 18 | Present assessments of risk of bias for each included study. | Not relevant |
| Results of individual studies | 19 | For all outcomes, present, for each study: (a) summary statistics for each group (where appropriate) and (b) an effect estimate and its precision (e.g. confidence/credible interval), ideally using structured tables or plots. | Not relevant |
| Results of syntheses | 20a | For each synthesis, briefly summarise the characteristics and risk of bias among contributing studies. | Section 6 |
| | 20b | Present results of all statistical syntheses conducted. If meta-analysis was done, present for each the summary estimate and its precision (e.g. confidence/credible interval) and measures of statistical heterogeneity. If comparing groups, describe the direction of the effect. | Not relevant |
| | 20c | Present results of all investigations of possible causes of heterogeneity among study results. | Section 6 |
| | 20d | Present results of all sensitivity analyses conducted to assess the robustness of the synthesized results. | Not relevant |
| Reporting biases | 21 | Present assessments of risk of bias due to missing results (arising from reporting biases) for each synthesis assessed. | Section 8 |
| Certainty of evidence | 22 | Present assessments of certainty (or confidence) in the body of evidence for each outcome assessed. | Not relevant |
| **DISCUSSION** | | | |
| Discussion | 23a | Provide a general interpretation of the results in the context of other evidence. | Section 6 |
| | 23b | Discuss any limitations of the evidence included in the review. | Section 8 |
| | 23c | Discuss any limitations of the review processes used. | Section 8 |
| | 23d | Discuss implications of the results for practice, policy, and future research. | Section 7 and Section 9 |
| **OTHER INFORMATION** | | | |
| Registration and protocol | 24a | Provide registration information for the review, including register name and registration number, or state that the review was not registered. | Not relevant |
| | 24b | Indicate where the review protocol can be accessed, or state that a protocol was not prepared. | Not relevant |
| | 24c | Describe and explain any amendments to information provided at registration or in the protocol. | Not relevant |
| Support | 25 | Describe sources of financial or non-financial support for the review, and the role of the funders or sponsors in the review. | End of the paper, before references |
| Competing interests | 26 | Declare any competing interests of review authors. | End of the paper, before references |
| Availability of data, code and other materials | 27 | Report which of the following are publicly available and where they can be found: template data collection forms; data extracted from included studies; data used for all analyses; analytic code; any other materials used in the review. | The paper is self contained and all the material is included either in the paper or in the appendix. |

*From:* Page MJ, McKenzie JE, Bossuyt PM, Boutron I, Hoffmann TC, Mulrow CD, et al. The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. BMJ 2021;372:n71. doi: 10.1136/bmj.n71
For more information, visit: http://www.prisma-statement.org/

## B Description of Included Studies

This appendix provides the full list of the 103 studies found in the literature using the methodology described in Section 5 and analyzed in Section 6. For each study, we report its type (INCL4-EXCL4) and provide a brief description of the application of LLMs in the field of CO. When applicable, we also mention the specific COP involved.

- Abdullin et al. [1] published a research study introducing a goal-oriented conversational agent to assist users in constructing accurate LP models. The LLM is used in the dialogue generation phase to automate the interaction between a conversational agent and a simulated user, effectively extracting necessary information to formulate LP models.
- AhmadiTeshnizi et al. [3] published a research study investigating the usage of LLM to formulate and solve LP and MILP problems from NL descriptions. The LLM is employed to develop an agent that recognizes entities and translates NL descriptions into code. This code is subsequently used by a solver to find solutions and potentially debug them. The work has been extended by AhmadiTeshnizi et al. [2].
- Ahmed and Choudhury [4] published a research study to investigate the usage of LLMs for recognizing entities and translating NL descriptions in optimization problem formulations. A LLM is used for this task in both zero-shot and one-shot settings.
- Alipour-Vaezi and Tsui [6] proposed a research study on using LLMs to extract domain knowledge in the context of the motion picture industry for portfolio optimization.
- Almonacid [7] published a research study investigating LLM capabilities of generating optimization code. The LLM is used to generate the model and its code using MiniZinc, eventually debugging it.
- Amarasinghe et al. [8] published a research study introducing a framework based on Copilot, designed to automate the generation of code from NL descriptions. The LLM is used to automate the generation of Python code employing the CPMpy library starting from business descriptions.
- Bohnet et al. [19] published a research study on using LLMs in generating solutions for planning problems.
- Borazjanizadeh et al. [20] published a research study on solving search problems using LLMs. The framework involves generating solution code and producing the solution in a specific format. The results were evaluated in terms of feasibility, correctness, and optimality. The study also introduces a new benchmark called SearchBench.
- Cai et al. [23] published a position paper on the possible advantages LLMs can bring to evolutionary computation, including evolutionary-based optimization.
- Chacón Sartori et al. [28] published a research study integrating LLMs into a web-based tool for visualizing the behavior of optimization algorithms. The LLM generates prompts for the tool, enabling users to comprehend how multiple algorithms behave when applied to specific instances of a COP.
- Chen et al. [30] published a research study related to FunSearch [184]; specifically, it introduces the concept of uncertainty to maintain diversity in the population of codes while ensuring a balance between exploration and intensification during the search. The results are compared to FunSearch and Evolution of Heuristic [126].
- Chin et al. [32] published a research study addressing the resolution of a particular CO, the **Montreal Capacitated Vehicle Routing Problem (FM-MCVRP)**. The LLM is used to train a model in a supervised manner on computationally inexpensive, sub-optimal solutions obtained algorithmically.

- Da et al. [37] published a research study on integrating LLMs into traffic and routing contexts. Specifically, the framework enables user—LLM conversations, generates solutions based on domain knowledge, and allows for solution visualization and validation.
- De La Rosa et al. [39] published a research study on using LLMs for travel planning. The LLM retrieves information about a given city and generates a route to visit specified landmarks.
- De Zarzà et al. [40] published a research study addressing the resolution of problems in a specific CO domain (financial and budget optimization). The LLM is used to provide domain-specific advice to the user.
- Dhanaraj et al. [45] published a research study on integrating human preferences into task scheduling for human-robot teams using CP and LLMs.
- Doan [47] published a research study on the topic of recognizing optimization entities. The LLM is used to recognize such entities from problem descriptions expressed in NL.
- Elhenawy et al. [52] proposed a research study on solving the TSP using LLMs. Differently from other studies, this one uses the visual capabilities of LLMs.
- Fan et al. [53] published a literature review exploring the integration of AI within the OR process, focusing on enhancing various stages such as parameter generation and model formulation.
- Freuder [57] published a position paper discussing the role of LLMs in constraint satisfaction problems, where traditionally a crucial component is the dialogue between an optimization expert and a domain expert. The LLM is used to replace the optimization expert.
- Gangwar and Kani [58] published a research study describing a method to translate NL descriptions into LP problem formulations.
- Ghiani et al. [63] proposed a research study on using LLMs to integrate stakeholders' preferences in decision-making processes. The methodology is applied to a last-mile delivery problem, and the LLM is used in two ways: to act as a construction heuristic and to set the weights of the optimizer (i.e., parameter tuning).
- Guo et al. [69] published a research study on using LLMs to generate solutions. Specifically, the LLM is asked to mimic the behavior of well-known algorithms, such as HC.
- Hao et al. [71] published a research study on using LLMs in planning. Specifically, LLMs are used to recognize entities, formulate the model, generate code, and catch errors.
- Hao Chen and Li [72] published a research study on the usage of LLM in detecting model infeasibility. The LLM is used to provide NL descriptions of the optimization model itself, identify potential sources of infeasibility, and offer suggestions to make the model feasible.
- He et al. [74] published a research study describing a method to generate LP problem formulations from NL descriptions. The LLM automates the transformation of text-based LP problem descriptions into structured formats, including entity recognition tasks.
- Hu et al. [80] published a research study on finding solutions for graph-related problems. Additionally, an explainability layer is included to outline the reasoning process.
- Huang et al. [81] published a research study on using LLMs to generate solutions for COPs. The experiments are conducted on the TSP.
- Huang et al. [82] published a research study proposing a tool called OR-Instruct, used to create synthetic data tailored to optimization modeling. They test the approach by developing the IndustryOR dataset.
- Huang et al. [84] published a literature review exploring the integration of LLM and optimization, considering both the perspective of LLM for optimization and optimization for LLM. The LLM is used as a black-box optimization search model or to generate optimization algorithms.

- Huang et al. [86] published a research study proposing an optimization framework based on LLMs in the context of CVRP. The LLM is used to generate solutions using textual and visual prompts.
- Huang et al. [87] published a research study addressing a specific class of COPs, the VRPs. The LLM is used to generate Python code from NL descriptions.
- Jang [89] published a research study introducing a method to generate LP problem formulations. The LLM is used to translate NL description into LP formulation with entity recognition.
- Jiang et al. [92] published a research study on using LLMs in an end-to-end approach: the LLM generates solution code directly from NL problem descriptions.
- Jiang et al. [93] published a research study on using LLMs as optimizers, i.e., generating solutions in the context of planning.
- Jin et al. [94] proposed a research study on using LLMs for problem formulation and code generation in the context of energy management.
- Jobson and Li [95] published a research study on using LLMs to create feasible schedules for conferences. The LLM either groups presentation titles or generates solutions.
- Ju et al. [96] published a research study on using LLMs for travel planning. LLMs recognize entities, create a corresponding MILP model, and solve the code.
- Khan and Hamad [98] published a research study investigating LLMs's capabilities in solving COPs. Specifically, LLMs either generate code or produce solutions. The study considers the TSP, the assignment problem, the transportation problem, and the shortest path problem.
- Kikuta et al. [99] published a research study introducing a framework designed to explain the decision-making process for a certain class of problems. The LLM is used to explain the influence of each route edge and integrate counterfactual explanations. The COP addressed is the VRP.
- Lai et al. [109] reviewed the literature on LLMs regarding their mathematical capabilities and briefly discussed their applications in CO, along with math word problems, geometry problems, and theorem proving.
- Lawless et al. [110] published a research study on using LLMs to decide which cutting-plane strategy to use, thus assisting the MILP solver in choosing the appropriate parameters.
- Lawless et al. [111] published a research study introducing a hybrid framework that integrates LLMs with CP to enable interactive decision support. The LLM is used in the preference elicitation phase to translate user input into structured constraint functions that the CP model can understand. The COP addressed is the Meeting Scheduling Problem, a specific **Constraint Satisfaction Problem (CSP)**.
- Li et al. [114] published a research study investigating the usage of LLM for reasoning about supply chain optimization. The LLM translates human queries into code, which is then utilized by an optimization solver. The final answer is returned via the LLM.
- Li et al. [115] published a research study on using LLMs to generate solutions for a travel planning problem. The solutions are evaluated and validated with user feedback.
- Li et al. [116] published a research study describing a method for synthesizing MILP models directly from NL descriptions. The LLM is used in the initial modeling phase to identify and classify decision variables and constraints.
- Li et al. [118] published a research study on generating MILP codes (and instances) with LLMs to train learning-based methods. The pipeline is called MILP-Evolve.
- Li et al. [119] published a research study on reasoning with graphs and networks. LLMs are used for knowledge extraction, entity recognition, and code generation in Python.

- Liu et al. [124] published a research study on automating the design of search operators within multi-objective evolutionary procedures. The LLM is used to generate new individuals for each subproblem within the decomposition approach.
- Liu et al. [125] published a research study on the usage of LLM in automatic heuristic design. The LLM is used to generate and evolve heuristic algorithm components, tested on TSPs, PFSP, and online **Bin Packing Problem (BPP)**.
- Liu et al. [127] published a research study on automatically generating optimization algorithms through an evolutionary framework. The LLM creates the initial solutions—each individual represents an algorithm—and applies mutation and crossover. The COP addressed is the TSP.
- Liu et al. [128] published a literature review on using LLMs for algorithm design, including optimization algorithms.
- Liu et al. [129] published a research study on algorithm design (heuristic code generation in Python), comparing results with other approaches such as FunSearch [184].
- Liu et al. [130] published a research study investigating LLMs as evolutionary combinatorial optimizers. A LLM selects parent solutions from a given population, performing crossover and mutation to generate offspring. The COP addressed is the TSP.
- Liu et al. [131] published a research study investigating the integration of LLM in a delivery route optimization problem (a variation of the TSP). The LLM is used to gather knowledge regarding delivery patterns.
- Long et al. [134] published a literature review on Network Operations and Performance Optimization, discussing potential roles for LLMs.
- Mao et al. [142] published a research study investigating the usage of LLM to enhance evolutionary procedures for identifying critical nodes in a graph. The LLM crosses and mutates given functions to generate new code snippets.
- Martinek et al. [143] proposed a research study in which LLMs are used to set MHs parameters, tested on the TSP and the Graph Coloring Problem.
- Michailidis et al. [146] proposed a research study examining LLMs from entity recognition to solution generation. In the context of entity recognition, the authors used Ner4Opt [38].
- Mo et al. [148] published a research study on generating solutions using LLMs in the context of planning.
- Mostajabdaveh et al. [152] published a research study analyzing the capabilities of LLMs in CO compared to human experts. A benchmark dataset called ORQUA is introduced, evaluating LLMs by asking them to answer questions based on NL problem descriptions.
- Mostajabdaveh et al. [153] published a research study on using LLMs in an end-to-end optimization process, where the LLM generates solution code from NL problem descriptions.
- Nana Teukam et al. [154] published a research study introducing a framework for optimizing enzymes. The LLM learns relationships between amino acid residues linked to structure and function, which are then used as input for an evolutionary search procedure aimed at improved catalytic performance.
- Ning et al. [157] published a research study on recognizing optimization entities and providing LP problem formulations. The LLM is used to identify these entities from NL problem descriptions and to generate the LP model.
- Obata et al. [158] proposed a research study on integrating LP and Dependency Graph approaches with LLMs for robot planning tasks.
- Pallagani et al. [166] proposed a review on the applications of LLMs in planning, including language translation, plan generation, model construction, multi-agent planning, interactive planning, heuristic optimization, tool integration, and brain-inspired planning.

- Ramamonjison et al. [179] published a research study to recognize entities and generate LP formulations from NL descriptions of LP problems.
- Ramamonjison et al. [180] published a technical report on the NL4Opt competition, describing tasks, datasets, metrics for evaluation, and competition statistics.
- Régin et al. [181] proposed a research study introducing GenCP, a framework combining CP with LLMs to handle structural constraints, including the semantic meaning in text generation.
- Reinhart and Statt [182] proposed a research study on using LLMs to generate solutions for macromolecule design. The LLM was also used to explain or justify the proposed solution.
- Romanko et al. [183] published a research study evaluating LLM stock-picking capability. The LLM is used to generate financial assets (solutions) for which a Mean-Variance Cardinality-Constrained Portfolio Optimization Model is computed.
- Romera-Paredes et al. [184] published a research study proposing a new evolutionary procedure. The LLM is integrated to find new solutions and heuristics for COPs by evolving the code of an initial program skeleton at each step. The COPs addressed are the Cap Set problem and online bin packing.
- Saka et al. [187] published a literature review investigating the usage of **Generative Pre-trained Transformer (GPT)** models in the **Architecture, Engineering, and Construction Industry (AEC)** industry. The review identifies opportunities, evaluates limitations, and validates a LLM use case for solution generation in AEC.
- Sartori et al. [188] published a research study proposing a tool called OptiPattern. The LLM extracts knowledge from instances and integrates it into the MH (GA) process. The methodology is applied to a network problem.
- Srivastava and Pallagani [197] published a position paper on the application of LLMs to planning and scheduling.
- Sui et al. [199] published a literature review addressing deep learning methods for the TSP, which also mentions the role of LLMs.
- Sun et al. [200] presented a research study on leveraging LLMs to solve SAT problems through a framework called AutoSAT. The framework automatically selects and optimizes heuristics within a predefined search space, building on conflict-driven clause learning solvers.
- Tran and Hy [212] published a research study on using LLMs in protein design. The LLM generates feasible solutions from incomplete ones, acting similarly to a repair operator in **local neighborhood search** (**LNS**).
- Tsouros et al. [213] published a research study introducing a framework for translating NL descriptions into CP. The LLM is used to produce executable code that can be run and debugged.
- Tupayachi et al. [215] published a research study on applying LLMs to domain knowledge and entity recognition. Specifically, the LLM builds knowledge graphs on the problem at hand (e.g., transportation networks).
- Ustyugov [217] published a research study on using LLMs for parameter tuning. Their methodology leverages BERT-based embeddings to predict solver parameters for MILP problems, aiming to automate and improve efficiency in Gurobi.
- van Stein et al. [219] published a research study on integrating LLM-generated code with parameter tuning. The code is generated via LLaMEA [218], and the tuning is performed using SMAC [122]. While LLaMEA was originally designed for black-box optimization, this study also addresses CO (TSP and online bin packing).
- Voboril et al. [223] proposed a research study introducing StreamLLM. The LLM enriches existing CP code in Python with additional constraints, aiming to reduce the search space.
- Wang et al. [225] published a research study on the topic of recognizing optimization entities. The LLM identifies such entities from NL problem descriptions.

- Wang et al. [227] published a research study on using LLMs as optimizers, i.e., generating solutions for drone placement.
- Wasserkrug et al. [229] published a position paper advocating for introducing LLMs into CO to democratize optimization practices, helping non-experts across different sectors.
- Wu et al. [236] published a literature review on vehicle routing, noting LLMs among other deep learning methods for TSP.
- Wu et al. [237] published a literature review on the intersection between LLM and evolutionary computation, offering insights on possible research directions.
- Wu et al. [238] published a research study on using LLMs for algorithm selection. More specifically, the LLM extracts features related to the underlying optimization algorithms.
- Xiao et al. [239] published a research study introducing a multi-agent framework that automates the translation of problem descriptions into mathematical formulations and executable code. The LLM formulates models from NL descriptions and generates runnable code.
- Yang et al. [240] published a research study on LLMs as optimizers. The LLM generates candidate solutions based on the problem description and previously evaluated solutions in the meta-prompt. The COP addressed is the TSP.
- Yang et al. [241] proposed a research study on using LLMs for entity recognition and Python code generation. The final LLM output also includes the solution, though not directly computed by the LLM. Two datasets, Optibench and ReSocratic-29k, are introduced.
- Yao et al. [242] published a research study on generating code that balances two objectives: efficiency of the code and the quality of solutions generated by it, tested on TSP and online bin packing.
- Yatong et al. [243] published a research study on generating heuristic code for task scheduling in edge servers.
- Ye et al. [244] published a research study on integrating LLM into hyper-heuristics generation. The LLM is used to generate heuristic algorithms in Python, tested on well-known COPs like TSP, CVRP, **Orienteering Problem (OP)**, **Decap Placement Problem (DPP)**, **Multiple Knapsack Problem (MKP)**, and BPP.
- You et al. [245] published a research study on a domain-specific COP (robot task sequencing). The LLM is used as a black-box optimizer.
- Yu and Liu [246] published a research study on the usage of LLMs for robust network design. The LLM generates heuristic code.
- Yu and Liu [247] published a position paper on the evolution of optimization toward automation, mentioning the integration of LLMs.
- Zhang et al. [249] published a research study proposing OptLLM, a system that accepts user queries in natural language, converts them into mathematical formulations and programming code, and calls solvers for decision-making. OptLLM supports multi-round dialogues to iteratively refine modeling and solving.
- Zhang et al. [251] published a research study on automated heuristic design (code generation). Results are compared with ReEvo [244] and FunSearch [184].
- Zhao et al. [254] published a literature review on optimization-based task and motion planning within a domain-specific CO. The review discusses how LLMs might generate domain knowledge and goal descriptions for planning methods.

## C Classification of Studies by Optimization Process Step

This appendix provides a breakdown of the identified studies with respect to the optimization process (Table 4). Columns display the general tasks (e.g., problem modeling), further divided into activities (e.g., domain knowledge). Furthermore, we report the optimization paradigm (e.g., CP) and technical details on the implementation (i.e., programming languages and commercial solvers). Each row groups together studies that share the same characteristics. The checkmark symbol (✓) identifies whether a study performs/is related to the column item.

Table 4. Classification of the Studies by Combinatorial Optimization Task Within the Optimization Process

| Studies | Prob. Mod. | | | Sol. Meth. | | | | Ben. | | Valid. | | Models/Algorithms Types | | | | | | | | | | | | Prog. Lang. | Lib./ Solv. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Domain Know. | Entity Rec. | Model Creation | Code Gen. | Solution Gen. | Param. Tuning | Alg. Selection | Explain. | Visual Analysis | Sol. Valid. | Model Val. | CP | LP | ILP | MILP | Heuristic | HH | EA | GA | QAP | MOO | Non Linear | MH (general) | SAT | | |
| [40, 131, 254] | ✓ | | | | | | | | | | | | | | | | | | | | | | | | |
| [188] | ✓ | | | | | | | | | | | | | | | | | | ✓ | | | | | | |
| [39] | ✓ | | | | ✓ | | | | | | | | | | | | | | | | | | | | |
| [95] | ✓ | | | | ✓ | | | | | | | | | ✓ | | | | | | | | | | | |
| [6, 215] | ✓ | ✓ | | | | | | | | | | | | | | | | | | | | | | | |
| [37] | ✓ | ✓ | | | ✓ | | | | ✓ | ✓ | | | | | | | | | | | | | | | |
| [45] | ✓ | ✓ | ✓ | | | | | | | | | ✓ | | | | | | | | | | | | | CP-SAT | |
| [119] | ✓ | ✓ | | ✓ | | | | | | | | | | | | | | | | | | | | | Python | |
| [47, 225] | | ✓ | | | | | | | | | | | ✓ | | | | | | | | | | | | |
| [87] | | ✓ | | ✓ | | | | | | ✓ | | | | | | | | | | | | | | | Python | |
| [116] | | ✓ | ✓ | | | | | | | | | | | | ✓ | | | | | | | | | | |
| [4, 74, 89, 157, 179] | | ✓ | ✓ | | | | | | | | | | ✓ | | | | | | | | | | | | |
| [158] | | ✓ | ✓ | | ✓ | | | | | | | | ✓ | | | | | | | | | | | | |
| [213] | | ✓ | ✓ | ✓ | | | | | | | | ✓ | | | | | | | | | | | | | Python | CMPy [67] |
| [96] | | ✓ | ✓ | ✓ | | | | | | | | | | | ✓ | | | | | | | | | | |
| [153] | | ✓ | ✓ | ✓ | | | | | | | ✓ | | ✓ | ✓ | | | | | | ✓ | | | | | | Zimpl |
| [71] | | ✓ | ✓ | ✓ | | | | | | | ✓ | | | | ✓ | | | | | | | | | ✓ | Python | MST, Gurobi |
| [2, 3, 92, 241] | | ✓ | ✓ | ✓ | | | | | | | | | ✓ | | ✓ | | | | | | | | | | Python | Pyomo [73], pyscipopt, Gurobi [70] (gurobipy) |
| [249] | | ✓ | ✓ | ✓ | | | | | | ✓ | ✓ | | ✓ | | ✓ | | | | | | | | ✓ | | MAPL code | Min-dOpt [250] |
| [146] | | ✓ | ✓ | ✓ | ✓ | | | | | | | ✓ | | | | | | | | | | | | | Python | CMPy [67] |
| [181] | | | ✓ | | | | | | | | | ✓ | | | | | | | | | | | | | | |
| [82] | | | ✓ | ✓ | | | | | | | | ✓ | ✓ | | ✓ | | | | | | ✓ | ✓ | | | Python | coptpy |
| [94] | | | ✓ | ✓ | | | | | | | | | | | ✓ | | | | | | | | | | Python | CVXPY [46] |
| [98] | | | | ✓ | ✓ | | | | | | | | | | | | | | | | | | | | Python | |
| [63] | | | | | ✓ | ✓ | | ✓ | | | | | | | | | | | | | ✓ | | | | | |
| [80, 182] | | | | | ✓ | | | ✓ | | | | | | | | | | | | | | | | | | |
| [52] | | | | | ✓ | | | | ✓ | ✓ | ✓ | | | | | | | | | | | | | | | |

(Continued)

Table 4. Continued

| Studies | Prob. Mod. | | | Sol. Meth. | | | | Ben. | | Valid. | | Models/Algorithms Types | | | | | | | | | | | | Prog. Lang. | Lib./ Solv. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Domain Know. | Entity Rec. | Model Creation | Code Gen. | Solution Gen. | Param. Tuning | Alg. Selection | Explain. | Visual Analysis | Sol. Valid. | Model Val. | CP | LP | ILP | MILP | Heuristic | HH | EA | GA | QAP | MOO | Non Linear | MH (general) | SAT | | |
| [217] | | | | | | ✓ | | | | | | | | | ✓ | | | | | | | | | | | |
| [115] | | | | | ✓ | | | | | ✓ | | | | | | | | | | | | | | | | |
| [19, 32, 69, 81, 86, 93, 148, 187, 212, 227, 240, 245] | | | | | ✓ | | | | | | | | | | | | | | | | | | | | |
| [143] | | | | | | ✓ | | | | | | | | | | | | | ✓ | | | | ✓ | | Python | Mealpy [210] |
| [110] | | | | | | ✓ | | | | | | | | | ✓ | | | | | | | | | | | |
| [30, 129, 242, 243, 246, 251] | | | | ✓ | | | | | | | | | | | | ✓ | | | | | | | | | Python | |
| [200] | | | | ✓ | | | | | | | | | | | | ✓ | | | | | | | | | Cpp | |
| [118] | | | | ✓ | | | | | | | | | | | ✓ | | | | | | | | | | Python | |
| [219] | | | | ✓ | | | | | | | | | | | | | | ✓ | | | | | | | Python | |
| [20] | | | | ✓ | ✓ | | | | | | | | | | | | | | | | | | | | Python | |
| [238] | | | | | | | ✓ | | | | | | | | | | | | | | | | | | | |
| [1, 58] | | | ✓ | | | | | | | | | | ✓ | | | | | | | | | | | | | |
| [239] | | | ✓ | ✓ | | | | | | | ✓ | | ✓ | | ✓ | | | | | | | | | | Python | Gurobi [70] (gurobipy) |
| [111] | | | ✓ | ✓ | | | | | | | | | ✓ | | | | | | | | | | | | Python | |
| [127, 184] | | | | ✓ | | | | | | | | | | | | ✓ | | | | | | | | | Python | |
| [244] | | | | ✓ | | | | | | | | | | | | | ✓ | | | | | | | | Python | |
| [114] | | | | ✓ | | | | | | | | | | | ✓ | | | | | | | | | | Python | Gurobi [70] (gurobipy) |
| [142] | | | | ✓ | | | | | | | | | | | | | | ✓ | | | | | | | Python | |
| [7] | | | | ✓ | | | | | | | | | | | ✓ | | | | | | | | | | MiniZ-inc | |
| [125] | | | | ✓ | | | | | | | | | | | | | ✓ | | | | | | | | Python | |
| [8] | | | | ✓ | | | | | | | | ✓ | | | | | | | | | | | | | Python | CMPy [67] |
| [223] | | | | ✓ | | | | | | | | ✓ | | | | | | | | | | | | | Python | MiniZinc |
| [183] | | | | | ✓ | | | | | | | | | | ✓ | | | | | | | | | | Python | CPLEX [88] (CVXPY [46]) |
| [130] | | | | | ✓ | | | | | | | | | | | | ✓ | | | | | | | | | |
| [124] | | | | | ✓ | | | | | | | | | | | | | ✓ | | | ✓ | | | | Python | PyMoo [17] |
| [154] | | | | | ✓ | | | | | | | | | | | | | | ✓ | | | | | | Python | GT4SD [140] |
| [99] | | | | | | | | ✓ | | | | | | | | | | | | | | | | | | |
| [28] | | | | | | | | | ✓ | | | | | | | | | | | | | | | | R/Web Int. | |
| [72] | | | | | | | | | | | ✓ | | | | ✓ | | | | | | | | | | Python | Gurobi [70] (Pyomo [73]) |

## D Classification of Studies by LLM Architecture

This appendix provides the analysis of the retrieved studies in terms of LLMs (Table 5). We categorize each LLM by its architecture, the tasks researchers employed it for during optimization, its release date, evaluation metrics, and corresponding studies. The table also highlights the access type (F/P, indicating free or paid) and source availability (O/C, indicating open or closed). Overlapping naming conventions often create confusion, as the same term can refer to both a general architecture and specific models. Additionally, it is sometimes unclear whether researchers fine-tuned a model directly or used a version adapted for conversational purposes. When researchers did not specify the exact model in their studies, we refer to the base architecture and denote it with the keyword Family, as is commonly done with GPT-4.

Table 5. Classification of Studies by the Role of 70 LLMs in Combinatorial Optimization Tasks. The F/P column indicates free or paid access type, while O/C denotes open or closed source availability.

| Architecture | LLM | Date | Task | F/P | O/C | Metrics | Studies |
|---|---|---|---|---|---|---|---|
| T5 [178] | T5-Base | 10/2019 | Problem Modeling, Solution Method | F | O | / | [32, 74] |
| | CodeT5-fine-tuned_CodeRL [112] | 07/2022 | Problem Modeling, Solution Method | F | O | Training Loss, Success Rate, and Correctness | [8] |
| BART [113] | BART-Base | 10/2019 | Problem Modeling | F | O | Accuracy | [58, 179] |
| | BART-Large | 10/2019 | Problem Modeling | F | O | Accuracy | [58, 89] |
| UnixCoder [68] | UnixCoder | 03/2022 | Solution Method | F | O | PAR10 | [238] |
| GPT-3.5 [21] | Text-Davinci-Edit-001 | 07/2022 | Solution Method | P | C | / | [7] |
| | Text-Davinci-003 | 11/2022 | Problem Modeling, Solution Method | P | C | Accuracy | [7, 111, 114] |
| | GPT-3.5 (Family) | 11/2022 | Problem Modeling, Solution Method, Benchmarking, and Validation | P | C | # API Call Rate, API Mismatching Rate, Error Raise Rate, Throughput, Average Travel Time, and GAP | [3, 20, 37, 249, 251] |
| | ChatGPT 3.5 | 11/2022 | Problem Modeling | F | C | Accuracy | [116] |
| | ChatGPT 3.5-Turbo | 11/2022 | Problem Modeling, Solution Method | P | C | / | [124, 127, 240] |
| | GPT-3.5-turbo | 11/2022 | Problem Modeling, Solution Method, and Validation | P | C | Accuracy, Hypervolume, Inverted Generational Distance, Compile Error Rate, and Runtime Error Rate | [125, 129, 239, 241–244] |
| | GPT-3.5-turbo-0613 | 06/2023 | Problem Modeling, Solution Method | P | C | F1-Score, ANC, Rank, Uncertainty, Policy Metric, and Goal Metric | [4, 69, 130, 142] |
| | GPT-3.5-Turbo-1106 | 11/2023 | Solution Method | P | C | Correctness, Output Format Consistency | [81] |

(Continued)

Table 5. Continued

| Architecture | LLM | Date | Task | F/P | O/C | Metrics | Studies |
|---|---|---|---|---|---|---|---|
| GPT-4 [161] | GPT-4 (Family) | 03/2023 | Problem Modeling, Solution Method, Benchmarking, and Validation | P | C | Accuracy, Feasibility, Optimality, Efficiency, ROUGE [120], BERTScore [252], Completeness Score, Homogeneity Score, No API Call Rate, API Mismatching Rate, Error Raise Rate, Throughput, Average Travel Time, Code Compilation Success, Debugging Success Rate, and Code Generation Efficiency | [1, 3, 20, 37, 39, 72, 87, 94, 95, 111, 114, 215, 240, 241, 249, 251] |
| | ChatGPT 4 | 03/2023 | Problem Modeling, Solution Method, and Benchmarking | P | C | Macro-F1 Score, Time | [6, 99, 183, 245] |
| | GPT-4-0613 | 06/2023 | Problem Modeling, Solution Method | P | C | F1-Score, Correctness, Output Format Consistency | [4, 40, 81] |
| | GPT-4-Turbo | 11/2023 | Problem Modeling, Solution Method, and Benchmarking | P | C | Score | [28, 98, 127, 173, 227, 244, 246] |
| | GPT-4-Vision-Preview | 12/2023 | Solution Method | P | C | / | [86] |
| | GPT-4o | 05/2024 | Problem Modeling, Solution Method, and Validation | P | C | Execution Rate, Solving Accuracy, and Average Solving Times | [2, 71, 92, 98, 118, 148, 188, 223] |
| | ChatGPT-4o | 05/2024 | Problem Modeling, Solution Method, Benchmarking, and Validation | P | C | Accuracy, Consistency, Stability, Solution Consistency, Constraints Robustness, Runtime and Efficiency | [52, 63] |
| | GPT-4o-2024-05-13 | 05/2024 | Solution Method | P | C | / | [219] |
| | ChatGPT-4-Turbo | 07/2024 | Problem Modeling, Solution Method | P | C | / | [80] |
| | GPT-4o-mini | 07/2024 | Solution Method | P | C | / | [80, 119, 129, 227] |
| | ChatGPT-4o-mini | 07/2024 | Problem Modeling, Solution Method | P | C | / | [80] |
| | GPT-4o-2024-08-06 | 08/2024 | Solution Method | P | C | Solved Instances, Penalized Average Runtime | [200] |
| PaLM 2 [66] | Bard | 05/2023 | Problem Modeling | F | C | Accuracy | [116] |
| | Codey | 05/2023 | Solution Method | P | C | / | [184] |
| | PaLM 2-L | 05/2023 | Solution Method | P | C | Accuracy | [240] |
| | PaLM 2-L-IT | 05/2023 | Solution Method | P | C | Accuracy | [240] |
| | Text-Bison | 05/2023 | Solution Method | P | C | Accuracy | [240] |

(Continued)

Table 5. Continued

| Architecture | LLM | Date | Task | F/P | O/C | Metrics | Studies |
|---|---|---|---|---|---|---|---|
| LLaMa 2 [62] | LLaMa 2 (Family) | 07/2023 | Solution Method | F | O | Accuracy | [32] |
| | LLaMa 2-7b | 07/2023 | Problem Modeling, Solution Method, Benchmarking, and Validation | F | O | F1-Score, Throughput, Average Travel Time, No API Call Rate, API Mismatching Rate, and Error Raise Rate | [4, 37] |
| | LLaMa 2-13b | 07/2023 | Problem Modeling, Solution Method, Benchmarking, and Validation | F | O | Throughput, Average Travel Time, No API Call Rate, API Mismatching Rate, and Error Raise Rate | [37] |
| | LLaMa 2-13b-Chat | 07/2023 | Solution Method | F | O | Correctness, Output Format Consistency | [200] |
| | CodeLlama [208] | 08/2023 | Solution Method | F | O | / | [20, 125, 251] |
| | CodeLlama-Instruct [208] | 08/2023 | Problem Modeling, Solution Method, and Validation | F | O | Accuracy, Feasibility, and Efficiency | [20, 153] |
| | Tulu-v2-dpo-7b [61] | 11/2023 | Benchmarking | F | O | Score | [28] |
| Mistral [90] | Mistral-7B | 09/2023 | Problem Modeling, Solution Method | F | O | Accuracy, Pass@K | [82] |
| | Zephyr-7B-beta [214] | 10/2023 | Problem Modeling, Solution Method, and Validation | F | O | / | [153] |
| | Le Chat | 07/2024 | Problem Modeling | P | C | / | [143] |
| Qwen [175] | Qwen1.5-14B | 10/2023 | Problem Modeling, Solution Method | F | O | Execution Rate, Solving Accuracy, and Average Solving Times | [92] |
| | Qwen-Turbo | 08/2024 | Solution Method | F | O | / | [129] |
| | Qwen (LoRA Fine-Tuned) | 08/2024 | Problem Modeling, Solution Method | P | C | / | [249] |
| Gemini [206] | Gemini 1.0 Pro | 12/2023 | Problem Modeling, Solution Method, and Validation | P | C | Feasibility, Optimality, Efficiency, and F1-Score | [87, 125] |
| | Gemini 1.5 Pro | 02/2024 | Problem Modeling | P | C | / | [19] |
| | Gemini 1.5 Flash | 02/2024 | Problem Modeling | P | C | / | [19] |
| | Gemini 2.0 Flash | 08/2024 | Problem Modeling | P | C | / | [143] |
| | Gemini 2.0 Pro | 08/2024 | Solution Method | P | C | / | [81] |
| Mixtral [91] | Mixtral-8x7b-instruct-v0.1 | 01/2024 | Benchmarking | P | C | Score | [28] |
| | Mixtral-8x22B | 07/2024 | Problem Modeling, Solution Method | P | C | / | [188] |

(Continued)

Table 5. Continued

| Architecture | LLM | Date | Task | F/P | O/C | Metrics | Studies |
|---|---|---|---|---|---|---|---|
| DeepSeek [41] | DeepSeek-LLM-7B-Base | 01/2024 | Solution Method | P | C | / | [125] |
| | DeepSeek-Math-7B [190] | 06/2024 | Problem Modeling, Solution Method | P | C | Accuracy, Pass@K | [82] |
| | DeepSeek-Coder-33B | 07/2024 | Solution Method | P | C | / | [30, 251] |
| | DeepSeek-V2 [42] | 08/2024 | Problem Modeling, Solution Method | P | C | / | [241] |
| | DeepSeek-Coder-V2 [43] | 08/2024 | Solution Method | P | C | / | [243] |
| Claude 3.5 [10] | Claude 3.5 Sonnet | 06/2024 | Problem Modeling, Solution Method, Benchmarking, and Validation | P | C | Accuracy, Solution Quality, # Good Solutions, Convergence Rate, Instruction Adherence, Consistency, Stability, Prompt Sensitivity, Stochastic Variability, Constraints Robustness, and Runtime Efficiency | [71, 182, 223] |
| | Claude 3.5 Opus | 06/2024 | Problem Modeling, Solution Method | P | C | / | [188, 251] |
| | Claude 3.5 Haiku | 06/2024 | Solution Method | P | C | / | [129, 129] |
| | Claude-3.5-Sonnet-20241022 | 10/2024 | Solution Method | P | C | / | [93] |
| Cohere [5] | Command-R+ | 06/2024 | Problem Modeling, Solution Method | P | C | / | [188] |
| LLaMa 3 [132] | LLaMa 3-70B | 07/2024 | Problem Modeling, Solution Method | F | O | / | [2, 96, 241, 244] |
| | LLaMa 3-8B | 08/2024 | Problem Modeling, Solution Method | F | O | Accuracy, Compilation Error Rate, and Runtime Error Rate | [82] |
| | LLaMa 3-70B-Instruct | 08/2024 | Solution Method | F | O | Accuracy, Pass@K | [243] |
| | LLaMa 3.1-8B | 08/2024 | Solution Method | F | O | / | [129] |
| Qwen2 [176] | Qwen2.5-7B | 07/2024 | Problem Modeling, Solution Method | F | O | Accuracy, Pass@K | [82] |
| StarCoder [117] | StarCoder2 [136] | 07/2024 | Solution Method | P | C | / | [251] |
| GLM [51] | GLM-3-Turbo | 07/2024 | Solution Method | P | C | / | [129, 243] |
| OpenCoder [83] | OpenCoder-8B-Instruct | 07/2024 | Solution Method | P | C | / | [30] |
| Yi [209] | Yi-34B-Chat | 07/2024 | Solution Method | P | C | / | [129] |
| Gemma [207] | Gemma 2 27B | 07/2024 | Problem Modeling | P | O | / | [19] |
| InternLM2 [24] | InternLM2-20B-Chat | 08/2024 | Solution Method | P | C | Correctness, Output Format Consistency | [81] |

## E Classification of Studies by Benchmark Dataset

This appendix provides the tabular analysis related to the classification of studies by dataset (Table 6). Please refer to Section 6.3 for further context.

Table 6. Classification of Studies by Benchmark Dataset

| Name | Source | Studies | # |
|---|---|---|---|
| LPWP or NL4Opt | Ramamonjison et al. [179] | [1, 3, 4, 47, 58, 74, 82, 89, 92, 116, 146, 157, 179, 225, 239, 249] | 16 |
| ComplexOR | Xiao et al. [239] | [3, 92, 239] | 3 |
| NLP4LP | AhmadiTeshnizi et al. [2, 3] | [2, 3, 92] | 3 |
| IndustryOR | Huang et al. [82] | [82, 92] | 2 |
| Mamo | Huang et al. [85] | [82, 92] | 2 |
| GraphInstruct | Luo et al. [137] | [80, 119] | 2 |
| AI-copilot-data | Amarasinghe et al. [8] | [8] | 1 |
| Almonacid | Almonacid [7] | [7] | 1 |
| Safeguard, Code Generation | Lawless et al. [111] | [111] | 1 |
| Huang et al. | Huang et al. [87] | [87] | 1 |
| OptiChat | Hao Chen and Li [72] | [72] | 1 |
| Michailidis et al. | Michailidis et al. [146] | [146] | 1 |
| Optibench, ReScratic-29k | Yang et al. [241] | [241] | 1 |
| Mostajabdaveh et al. | Mostajabdaveh et al. [153] | [153] | 1 |
| Zhang et al. | Zhang et al. [249] | [249] | 1 |
| SearchBench | Borazjanizadeh et al. [20] | [20] | 1 |
| Ju et al. | Ju et al. [96] | [96] | 1 |
| Talk Like A Graph | Fatemi et al. [54] | [119] | 1 |
| LLM4DyG | Zhang et al. [253] | [119] | 1 |
| GraphViz | Chen et al. [29] | [119] | 1 |
| NLGraph | Wang et al. [224] | [119] | 1 |
| GNN-AutoGL | Li et al. [119] | [119] | 1 |
| ORQUA | Mostajabdaveh et al. [152] | [152] | 1 |

## F Classification of Studies by Application Domain

This appendix provides the tabular analysis related to the classification of studies by application domain (Table 7). Please refer to Section 6.4 for further context.

Table 7. Classification of Studies by Application Domain

| Domain | COP/Detail | Studies | # |
|---|---|---|---|
| Routing | Traveling Salesperson | [30, 52, 81, 98, 124, 125, 127, 129, 131, 143, 199, 219, 240, 242, 244, 251] | 26 |
| | Vehicle Routing | [32, 37, 63, 86, 87, 98, 99, 129, 236, 244] | |
| | Orienteering | [244] | |
| | Travel | [39, 96, 115] | |
| Scheduling and Planning | Permutation Flowshop Scheduling | [8, 125] | 14 |
| | Meeting and Conferece Scheduling | [95, 111] | |
| | Server Scheduling | [243] | |
| | Planning | [19, 45, 71, 93, 148, 158, 166, 227, 245] | |
| Network and Graphs | Network Design | [80, 119, 134, 215, 246] | 10 |
| | Critical Node Identification | [142] | |
| | Coloring | [143] | |
| | Social Network | [188] | |
| | Shortest Path, Assignement | [98] | |
| | Path Finding | [20] | |
| Packing | Bin Packing | [30, 125, 129, 184, 219, 242, 244, 251] | 8 |
| | Multiple Knapsack | [244] | |
| Combinatorics | Cap Set | [184] | 4 |
| | Admissible Set | [251] | |
| | Puzzles | [20, 146] | |
| | Subset Sum, Sorting, Under-determined Systems | [20] | |
| Engineering | Construction | [187] | 3 |
| | Circuit Design | [244] | |
| | Energy Management | [94] | |
| Finance | Portfolio Optimization | [6, 40, 183] | 3 |
| Bioinformatics | Enzyme Design | [154, 182, 212] | 3 |
| Supply Chain | Warehousing | [114] | 1 |
| Strings | Text Generation | [181] | 1 |

Received 3 September 2024; revised 11 March 2025; accepted 3 March 2026