AutoOpt: A Dataset and a Unified Framework for Automating Optimization Problem Solving

Ankur Sinha, Shobhit Arora, and Dhaval Pujara
Brij Disa Centre for Data Science and AI
Indian Institute of Management Ahmedabad
Ahmedabad, Gujarat 380015
{asinha,shobhita,dhavalp}@iima.ac.in

Abstract

This study presents *AutoOpt-11k*, a unique image dataset of over 11,000 handwritten and printed mathematical optimization models corresponding to single-objective, multi-objective, multi-level, and stochastic optimization problems exhibiting various types of complexities such as non-linearity, non-convexity, non-differentiability, discontinuity, and high-dimensionality. The labels consist of the LaTeX representation for all the images and modeling language representation for a subset of images. The dataset is created by 25 experts following ethical data creation guidelines and verified in two-phases to avoid errors. Further, we develop *AutoOpt* framework, a machine learning based automated approach for solving optimization problems, where the user just needs to provide an image of the formulation and *AutoOpt* solves it efficiently without any further human intervention. *AutoOpt* framework consists of three Modules: (i) M1 (*Image_to_Text*)- a deep learning model performs the Mathematical Expression Recognition (MER) task to generate the LaTeX code corresponding to the optimization formulation in image; (ii) M2 (*Text_to_Text*)- a small-scale fine-tuned LLM generates the PYOMO script (optimization modeling language) from LaTeX code; (iii) M3 (*Optimization*)- a Bilevel Optimization based Decomposition (BOBD) method solves the optimization formulation described in the PYOMO script. We use *AutoOpt-11k* dataset for training and testing of deep learning models employed in *AutoOpt*. The deep learning model for MER task (M1) outperforms ChatGPT, Gemini and Nougat on BLEU score metric. BOBD method (M3), which is a hybrid approach, yields better results on complex test problems compared to common approaches, like interior-point algorithm and genetic algorithm.

Keywords: Mathematical Programming, Optimization Formulation, Deep Learning, Mathematical Expression Recognition

1 Introduction

Optimization is an active field of research due to its potential to deliver substantial and sustainable benefits to users by providing efficient solutions to complex optimization problems that commonly arise in practice. There are efforts to generate mathematical programs from verbal explanations using the large-language models (LLMs) [1, 52], which currently works for simple problems and often requires iterations and further refinements to arrive at the right formulation. Little attention has been given to automate the solution of optimization problems stored in image-based formats, such as figures in research articles, scanned pages from books, handwritten notes, or whiteboard snapshots. Humans can easily read and

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

interpret the information provided in images in the form of mathematical programming formulations, also known as mathematical models, mathematical programs, mathematical formulations, or simply optimization problems. However, this is not the case for machines as images lack the semantic structure [66], making it difficult for machines to understand the image content, i.e., poor machine readability. Therefore, such formulations always need to be represented in the form of a structured modeling language for the computer or an optimization solver to understand and solve it.

Very often in a classroom setting, research setting, or industrial setting, when a business or engineering problem is discussed, a mathematical formulation is worked out on the whiteboard, tablet, or paper and saved in the form of an image. After this, the tasks of converting the problem into a machine-readable format and solving it with a suitable solver, still remains. In most of the engineering and business schools, there are quite a few sessions devoted on solving such problems after the mathematical model is ready. Such tasks are usually mechanical and can be automated. In the current era of Artificial Intelligence (AI), humans aim to leverage machine learning by developing automated systems [43], in which machines are primarily responsible for executing tasks with limited human intervention. In the context of the current study, it means that we want the machines to learn to interpret the mathematical formulations and participate in problem-solving tasks. However, to enable a machine to learn, it requires datasets that map mathematical formulations in images to their machine-readable representation in text format, which is currently lacking in the optimization community. There exist studies, where researchers perform Optical Character Recognition (OCR) task [12], in which the content within an image is recognized and converted into machine-encoded text, a machine-readable format that can be easily understood and processed by machines. Application of OCR to identify and convert text within images into machine-readable format is also known as text recognition [15], and in case of mathematical expressions, it is referred to as Mathematical Expression Recognition (MER) [24, 67]. Optimization problems are commonly described through complex verbal explanations or large mathematical programs. When expressed as mathematical programs, it is not straightforward to apply existing OCRs and MERs to completely understand complex formulations and convert them into a machine-readable format. This study attempts to bridge this gap, which is of significant importance to the optimization community.

Tesseract OCR [74], a well-known text recognition method, follows one-dimensional and line-by-line approach. It detects a single line of text from an image or PDF, sequentially identifies letters, words, or spacing in a considered line, and then moves to the next line. Such a one-dimensional or horizontal approach is not sufficient for MER, as mathematical expressions consist of several structural components such as subscripts, superscripts (exponents), fractions, matrices, etc. These components are characterized by the relative spatial positions of characters and symbols in expression, and they convey specific semantic relationships and mathematical meaning. To detect and preserve this meaning into machine-encoded text, methods need to analyze the content not only horizontally but also vertically, in a two-dimensional manner [32]. Further, MERs which work with single line mathematical expressions may also not suffice for mathematical programs, which are often multilined and contain interlinked information, such as variable(s), parameter(s), objective function(s) and constraint(s). Therefore, there is a need for MERs that have the ability to understand mathematical programs as a whole rather than in pieces.

In the case of optimization problems, after converting a mathematical model into machine-readable format (for example, LaTeX) using MER, there is a scope of further value addition by creating a setup that can extract the relevant data from the converted optimization problem and fit it into a predefined programming structure (for example, mathematical modeling languages, like AMPL, PYOMO, etc.). This program structure can be passed to a mathematical solver or an optimization technique implemented in the computational system to solve the respective optimization problem effectively. In this way, the task of solving optimization problems can be automated, where user only needs to provide an image of the mathematical program, and an efficient solution to the respective mathematical program can be retrieved with little human intervention. This study achieves the same by proposing *AutoOpt*, an automated optimization framework. The dataset and the code associated

with *AutoOpt* framework is being made publicly available[1]. The workflow of the *AutoOpt* framework is provided in Figure 1, which contains three modules described next:

- **M1 (*Image_to_Text*)**: A deep learning module that takes an image as an input and generates LaTeX code corresponding to the mathematical model in image.
  Contribution: Releasing an Image to LaTeX dataset (*AutoOpt-11k*) with 11,554 mathematical programs that are a mix of handwritten and typeset (printed) images. A deep learning architecture is also proposed to capitalize on this dataset.
- **M2 (*Text_to_Text*)**: A deep learning model that takes LaTeX code from module M1 as input and generates a PYOMO script.
  Contribution: Releasing an Image/LaTeX to PYOMO dataset with 1,018 mathematical programs (a subset of *AutoOpt-11k*). A pre-trained deep learning model is fine-tuned to develop this module.
- **M3 (*Optimization*)**: An effective bilevel optimization based decomposition (BOBD) method, implemented in Python, solves the problem using the PYOMO script obtained from module M2.
  Contribution: We build up on a recently proposed approach [72] by automating the decomposition task using machine learning.

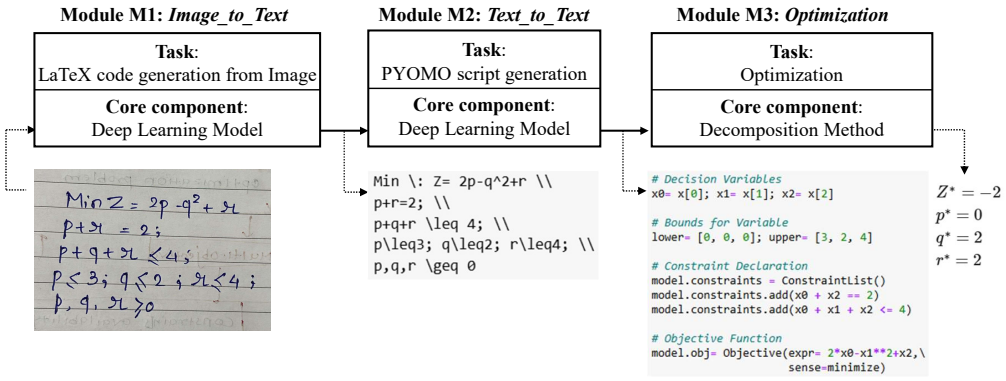


Figure 1: *AutoOpt* framework: workflow demonstration using an example

Note that it is possible to generate the final PYOMO script directly from an image using a single deep learning model. However, we adopt a two-stage approach, first generating LaTeX and then converting it to PYOMO, for several practical reasons. A two-step design enhances the ease of verification, as the intermediate LaTeX output serves as a human-readable checkpoint to inspect the accuracy of the model interpretation prior to code generation. Moreover, not all generated scripts are guaranteed to be executable due to errors of incompleteness in mathematical model description.

The deep learning models used in *AutoOpt* framework (modules M1 & M2) must be trained on a rich, diverse, well-curated, and representative dataset of mathematical models to ensure effectiveness of the proposed approach. The literature offers a wide range of image datasets designed to train AI models for MER [24, 32, 67, 82, 59]. These datasets contain images of single-lined and small mathematical expressions typically drawn from various mathematical streams such as algebra, calculus, geometry, probability, statistics, etc. Hence, from the perspective of mathematics, these datasets are quite general in nature. However, the objective of automating optimization requires a rich dataset containing images of mathematical programs corresponding to optimization problems with varying levels of complexity. To the best of our knowledge, such a dataset does not exist currently, i.e., a dataset specifically for the optimization domain. We bridge this gap by developing *AutoOpt-11k* image dataset, a collection of more than 11,000 mathematical programs. *AutoOpt-11k* covers small-to-large scale theoretical and real-life problems from various categories of optimization problems

[1]The *AutoOpt-11k* dataset can be accessed through the link `https://www.kaggle.com/datasets/ankurzing/autoopt-11k`, and the code for *AutoOpt* framework can be accessed through the link `https://github.com/Shobhit1201/AutoOpt`.

such as constrained, unconstrained, linear, non-linear, convex, non-convex, single-objective, multi-objectives, etc. For each image of a mathematical program, that can be handwritten, printed or a mix of handwritten and printed, we provide its LaTeX representation. We provide a PYOMO representation for a subset of these images. The dataset has been created with the support of 25 experts and has been verified in two-phases to avoid errors.

The paper is organized as follows: The dataset development procedure and characteristics of the *AutoOpt-11k* dataset are provided in Section 2. Mechanism for solving an optimization problem using the automated optimization framework, *AutoOpt*, is demonstrated using an example along with a detailed description of the modules, M1, M2 and M3, in Section 3. The experimental results are provided in Section 3 and also in the Appendices. Finally, concluding remarks, future research directions and limitations are discussed in Section 4.

2 AutoOpt-11k Dataset

In this section, we discuss the key characteristics and the development process of *AutoOpt-11k*, an image dataset of mathematical programs created in this study. *AutoOpt-11k* consists of 11,554 images, each illustrating a distinct mathematical model corresponding to an optimization problem drawn from domains such as science, engineering, business, and related fields. Of these, 5,070 images feature handwritten mathematical models created manually by human annotators, while the remaining 6,484 images present typeset models taken from printed sources or generated using computer system. Apart from writing on paper, it is now common for people to write on tablets, electronic boards or touch screens, where typeset and handwritten text may appear together. The created data set incorporates these kinds of variations. A small sample of images that are part of the dataset is shown in Figure 2.

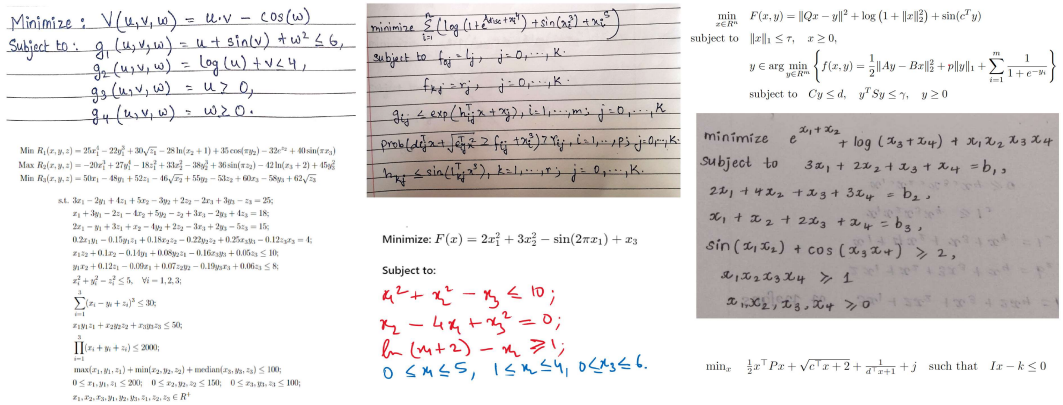


Figure 2: A sample of 7 images from the *AutoOpt-11k* dataset.

The *AutoOpt-11k* dataset has been systematically compiled to capture the broad diversity observed in optimization problems across theoretical and applied contexts. It includes single-objective optimization problems [33], which involve the maximization or minimization of a single criterion, as well as multi-objective problems [19], where multiple, often conflicting, objectives must be optimized simultaneously. The dataset also contains bilevel/multi-level optimization problems [71], which are hierarchical in nature and involve nested decision-making structures, and stochastic problems [9], which involve uncertainty in the problem definition. From the perspective of constraints [27], both constrained and unconstrained optimization problems are included, enabling the training of models that can interpret a wide range of structural conditions.

In mathematical modeling, optimization problems are typically described in either a general form, where parameters such as coefficients in the objective function and constraints are left symbolic, or in a numerical form, where all such values are explicitly provided. This distinction is crucial in both theoretical exposition and practical implementation. Accordingly, *AutoOpt-11k* includes problem statements in both general and numerical forms. Further, mathematical models can be expressed either using compact matrix-vector notation or in a scalar format involving only scalar operations. Matrix-vector notation is often preferred

Table 1: Composition of *AutoOpt-11k* dataset based on the characteristics of optimization problems

	Type	Count	Description
AutoOpt-11k	Handwritten	5,070	Written by hand on paper, tablet, electronic book, etc.
	Typeset	6,484	Printed format extracted from books, articles, etc.
	Total	11,554	
Types of problems	Single objective	10,838	Only one objective function is defined
	Multi-objective	159	Contains multiple objective functions
	Multi-level	399	Contains two or more levels of optimization
	Uncertainty	158	Contains some form of parameter or variable uncertainty
Constraint availability	Unconstrained	155	Constraints are absent
	Constrained	11,399	One or more constraints are present
Model form	General form	7,349	Contains undefined parameters, functions, etc.
	Fully defined	4,205	Completely defined with all necessary parameters
Presentation form	Vector form	608	Defined in a form containing vector and matrix operations
	Scaler form	10,246	Defined in a form containing only scalar operations
	Scalable form	804	Problem is scalable in terms of variables, objectives, etc.
Other	Linear	2,130	All objectives and constraints are linear
	Non-linear	9,122	One or more objectives or constraints are non-linear
	Continuous	10,806	All variables and functions are continuous
	Discontinuous	424	Involves integer variables or contains discontinuities
	Convex	2,580	Belongs to the class of convex optimization
	Non-convex	3,574	Belongs to the class of non-convex optimization
	Differentiable	9,502	All the functions defined are differentiable
	Non-differentiable	502	Some functions defined are not differentiable

There are formulations that belong to multiple categories and also formulations that cannot be classified appropriately.

in compact representations, especially in linear algebraic formulations, while algebraic expressions are commonly used in detailed, problem-specific contexts. *AutoOpt-11k* includes mathematical programs expressed in both styles.

The functional form of the objective function and constraints contributes significantly to the complexity of an optimization problem. These functions may exhibit various mathematical properties, such as being linear or non-linear, convex or non-convex, continuous or discontinuous, and differentiable or non-differentiable. These characteristics directly influence the selection of appropriate solution methods and the computational difficulty of solving the problem. In addition, the dimensionality of the problem—defined by the number of decision variables and constraints—plays a key role in determining its scale and complexity. To ensure comprehensive coverage, *AutoOpt-11k* incorporates problems spanning a wide range of functional behaviors and scales, thus including mathematical programs across a broad landscape of optimization scenarios. The diverse composition of *AutoOpt-11k* dataset is provided in Table 1 along with the count of images of each type. The optimization problems have been taken from or inspired by the sources mentioned in Table 2.

Table 2: Sources used in creating *AutoOpt-11k* dataset.

Source Type	**References**
Books	[28, 8, 57, 27, 63, 35, 65, 26, 13, 53, 29, 9, 75, 63, 5]
Research Papers and Reports	[39, 10, 25, 62, 69, 40, 4, 85, 34, 6, 14, 42, 73, 2, 7, 61, 11, 51, 46, 76, 22]
Mathematical Modeling Software Documentations	AMPL [3], GAMS [31], PYOMO [38], JuMP [44], LINDO [47]
Solver Documentations	CPLEX [41], Gurobi [37], CBC [16], CLP [30], Ipopt [81]
Online Repositories	COIN-OR [17], MIPLIB [55], OR-Library [58], UCI [64], NEOS [68], Netlib [56]

In the literature, mathematical models are expressed in a variety of notational and formatting styles. For instance, in the in-line style, the objective function and constraints are written horizontally in a single line, whereas in the multi-line style, they are arranged vertically with each component on a separate line. Additional stylistic variations include differences in objective function declarations (e.g., *min/max* vs. *minimize/maximize*), separators for

constraints and objectives (e.g., *s.t.*, *w.r.t.*, *subject to*), constraint indexing conventions (e.g., $i \in [1, K]$ vs. $i = 1, \ldots, K$), and text formatting aspects such as indentation, alignment, font type and size, line spacing, and use of bold or italic styles. Variations also occur in mathematical expressions (e.g., $x^{1/2}$, $x^{0.5}$, $\sqrt{x}$) and in variable or parameter naming conventions (e.g., $p/q/r$, $a/b/c$, $x_1/x_2/x_3$, or a mix such as $p/a/x_1$). *AutoOpt-11k* incorporates mathematical programs reflecting this broad spectrum of notational and stylistic differences.

In the case of handwritten images, human involvement introduces additional layers of variability beyond those found in typeset representations. These include differences in handwriting style (e.g., font size, font type), paper type (such as plain or ruled), ink color (typically blue or black, and other colors on digital writing devices), and conditions under which the images are captured. Image captures vary in terms of camera angle, distance, orientation, lighting conditions, and camera specifications (such as resolution). Some of the images are also captured through a scanner. The dataset incorporates handwritten images reflecting this full range of variations that we obtained working with multiple annotators. Overall, 25 annotators, having engineering or business background and mathematics exposure up to the bachelors, were employed to form the dataset. We followed a two-phase process for dataset generation. In the first phase, 20 annotators identified the optimization problems from various sources and also submitted handwritten versions of some of those optimization problems. Thereafter, 5 annotators with background in programming, were recruited to prepare the LaTeX code of all the images and PYOMO script for a subset of images.

To minimize errors in dataset generation and ensure high-quality annotations, the annotators regenerated each image from the respective LaTeX code and visually compared the generated image against the original image. This cross-verification step ensured consistency between the image and its LaTeX representation, helping to identify and correct any discrepancies introduced during the initial annotation. In this second phase of annotation process, each of the 5 annotators (A1, A2, A3, A4, A5) annotated 30% of the images and there was a 16.6% overlap between any pair of annotators on average. The Inter Annotator Agreement (IAA) score for each pair of annotator is provided in Table 3. The reason for discrepancies between annotators was often because the code generated by them had syntactic differences for the same image.

Table 3: Inter-Annotator Agreement Scores (BLEU and CER)

Pair	BLEU		CER	
	Mean	Std	Mean	Std
A1 vs A2	0.8187	0.1066	0.1784	0.1154
A1 vs A3	0.8185	0.1065	0.1788	0.1153
A1 vs A4	0.8195	0.1071	0.1776	0.1167
A1 vs A5	0.8201	0.1068	0.1782	0.1155
A2 vs A3	0.8588	0.1031	0.1267	0.1020
A2 vs A4	0.8581	0.1042	0.1273	0.1040
A2 vs A5	0.8189	0.1062	0.1791	0.1148
A3 vs A4	0.8574	0.1044	0.1286	0.1050
A3 vs A5	0.8192	0.1064	0.1787	0.1150
A4 vs A5	0.8197	0.1070	0.1779	0.1159

AutoOpt-11k dataset contains the LaTeX representation for all 11,554 images. The number of unique mathematical programs in the dataset is 7,637 out of 11,554, as we chose to include the typeset as well as handwritten versions of a variety of mathematical programs in our dataset. For a subset of 1,018 unique mathematical programs in the *AutoOpt-11k* dataset, we also provide the PYOMO scripts. Table 4 summarizes key statistics, and further details are available in Appendix A.

Table 4: Summary Statistics for Image, LaTeX, and PYOMO Samples

Metric	Min	Max	Mean	Median	Total Samples
Image Width (px)	159	3611	783.91	753.50	11,554
Image Height (px)	24	2670	338.89	295.00	
Aspect Ratio (W/H)	0.25	18.29	2.73	2.40	
File Size (KB)	3.06	1399.98	95.58	41.68	
LaTeX Length (chars)	14	1,620	212.23	180.00	11,554
PYOMO Length (chars)	192	1,087	390.30	362.00	1,018

3 Framework for Automating Optimization Problem Solving

This section details the development of *AutoOpt* framework, illustrated in Figure 1. The framework is composed of three sequential modules—M1, M2, and M3—each responsible for a specific task in the automated optimization pipeline. The output from each module serves as the input for the subsequent one. Details on computational set up and infrastructure, along with detailed results from multiple runs are relegated to Appendices B, C and D.

3.1 Module M1: Image to LaTeX Code Generation

In this module, we propose a hybrid deep learning architecture suitable for MER. The proposed model extends the NOUGAT architecture [12, 83], a DONUT-based framework comprising a vision encoder and a text decoder, specifically designed for typeset scientific documents. Given the inherent complexity of two-dimensional mathematical notation, which may be handwritten or typeset, we design a hybrid vision encoder by integrating ResNet and Swin Transformer [50] thereby leveraging the strengths of both Convolutional Neural Networks (CNNs) and Transformers [60]. CNNs are effective at capturing local visual patterns, while Transformers excel at modeling long-range dependencies and global structure. The overall architecture is shown in Figure 3.

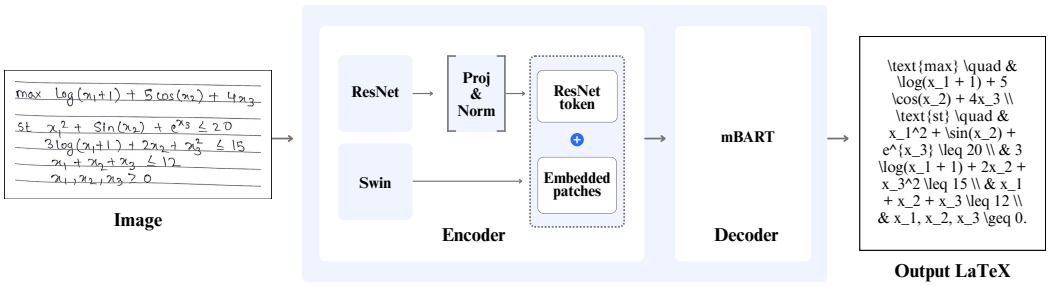


Figure 3: Architecture of the deep learning model developed for MER

The hybrid encoder combines ResNet-101 and Swin Transformer. ResNet-101 serves as a backbone for local feature extraction, producing a 2048-dimensional feature vector via average pooling. These features capture fine-grained local characteristics such as symbol shape and stroke patterns—crucial for both printed and handwritten expressions. In parallel, the Swin Transformer processes the input image by applying hierarchical self-attention within and across non-overlapping windows. This enables it to capture long-range dependencies and spatial layouts such as superscripts, subscripts, matrices, and fractions. The output of the Swin Transformer is a sequence of patch-level embeddings. To combine both streams, the ResNet-generated feature vector is projected to match the Transformer's hidden dimension and prepended to the sequence of patch embeddings, enabling joint learning of global and local context. To integrate CNN-derived features with Transformer-based patch embeddings, we introduce a lightweight fusion strategy provided below.

$$\mathbf{f}_{\text{ResNet}} = \alpha \cdot \text{LayerNorm}(\text{Proj}(\text{ResNet}_{\text{feat}})),$$

where $\alpha \in \mathbb{R}$ is a learnable scalar initialized to zero, acting as a gating parameter during early training. This vector is then prepended to the sequence of Swin Transformer embeddings.

The decoder in our proposed architecture is based on the mBART [49, 45] architecture—a pre-trained Transformer-based autoregressive decoder. The decoder generates LaTeX code token-by-token, attending to the fused encoder outputs via cross-attention and to past generated tokens via causal self-attention. The NOUGAT model uses the same decoder, therefore we initialize the decoder with pre-trained NOUGAT weights while training it on our task-specific dataset.

3.1.1 Experimental Results

To ensure uniformity in input dimensions and improve model robustness, we implement a tailored preprocessing pipeline. Each input image is first resized so that its longer side fits within a 768×1024 canvas while maintaining the aspect ratio. The resized image is then center-padded on a white background to match the target dimensions. This standardization ensures consistent input representation regardless of the original aspect ratio. Given the complexity and density of the mathematical expressions in our dataset, we apply contrast enhancement to improve visual clarity. Additionally, an unsharp mask filter is used to accentuate symbol boundaries and fine pen strokes, which are critical for handwritten mathematical notation.

We adopt a transfer learning approach to train our model. ResNet-101 is initialized with ImageNet weights [23], while the Swin Transformer and mBART decoder are initialized using pre-trained weights from the NOUGAT model. These pre-trained models, trained on large-scale scientific corpora, provide a good starting point, enabling faster convergence and better generalization while training on *AutoOpt-11k*. In our experiments a training, validation, and test split of 80%, 10%, and 10% is used.

We compare our model that we refer to as AutoOpt-M1 against Nougat, ChatGPT and Gemini. Nougat has been fine-tuned on *AutoOpt-11k* dataset. For ChatGPT we use the GPT 4o model, and for Gemini we use Gemini 2.0 Flash model, both through their APIs. The ChatGPT and Gemini models were not fine-tuned but were given appropriate prompts with examples. Figure 4 compares these models with respect to BLUE Score (larger is better), and Table 5 compares them with respect to Character Error Rate (smaller is better). Clearly, Nougat and AutoOpt-M1 are much smaller and better performing models as compared to ChatGPT and Gemini models. Between Nougat and AutoOpt-M1, the latter outperforms the prior on all metrics except on Character Error Rate for Printed. However, note that there is a possibility to produce slightly different LaTeX code for the same image; therefore, for all models there are situations where the predicted LaTeX code is semantically correct, but the ground truth LaTeX is different. For further details, refer to Appendix B.

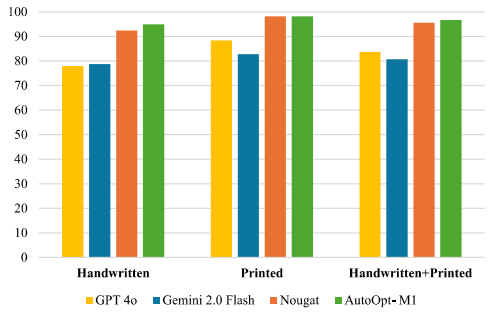


Figure 4: BLUE Score

Table 5: Performances of various approaches considered for MER task

Model (Model Size)	**HW**	**PR**	**HW+PR**
	Character Error Rate		
GPT 4o (Large)	0.1465	0.0664	0.1017
Gemini 2.0 Flash (Large)	0.1607	0.1047	0.1338
Nougat (348.7M)	0.0752	**0.0168**	0.0440
AutoOpt-M1 (393.3M)	**0.0412**	0.0176	**0.0286**
HW: Handwritten; PR: Printed; HW+PR: Handwritten+Printed			

We arrived at the hybrid encoder architecture based on an ablation study, in which we tried Deep Learning (DL) models with different architectures such as DL1 (with CNN, without Transformer): BLEU- 16.10, CER- 0.8812; DL2 (without CNN, with Transformer): BLEU- 95.51, CER- 0.0440; and DL3 (with CNN, with Transformer): BLEU- 96.70, CER- 0.0286. Finally, based on the comparison of BLEU and CER performance metrics, DL3 (Figure 3) is selected.

3.2 Module M2: LaTeX to PYOMO Script Generation

This module generates the model-specific PYOMO script from LaTeX code. To operationalize this task, we fine-tune a causal decoder-only transformer model using the instruction-style data. We specifically considered the DeepSeek-Coder 1.3B [36], a pre-trained language model as the base. This instruct model has strong code generation capability and pre-trained alignment to instruction-following tasks, and it is smaller as compared to other coding LLMs. Fine-tuning is performed on 80% of 1,018 mathematical models, while the remaining 20% is used for testing. We refer to the fine-tuned model as AutoOpt-M2, for which we obtained a BLUE Score of 88.25 and Character Error Rate of 0.0825. Refer to Appendix C for additional details.

3.3 Module M3: Optimization using Bilevel Optimization based Decomposition Method

In module M3, we implement an optimization method capable of efficiently solving a wide range of small-to-large scale optimization problems. Based on the nature of delivered solution (i.e., optimal or approximate), optimization methods are broadly classified into two categories: exact and approximation methods. Classical mathematical programming based techniques (such as linear programming [18], integer programming [79], etc.) fall into the category of exact methods. These methods guarantee optimality but often require certain regularities, like linearity, continuity, differentiability, etc. On the other side, approximate methods like heuristics and metaheuristics can lead to a satisfactory solution on irregular problems but may not scale well and do not guarantee optimality. Interestingly, some recent studies [48, 80, 84] explore how LLMs can be used to design problem-specific heuristics. An alternative line of study [72] proposes a bilevel optimization based decomposition strategy to utilize metaheuristic and classical approaches simultaneously to solve a wide variety of problems. Our implementation in module M3 is an extension of work by Sinha et al. [72].

In Figure 5, we demonstrate the procedure of BOBD method by solving the optimization problem considered in Figure 1. The optimization problem in the first tab of Figure 5 contains three variables p, q and r. This is a non-convex optimization problem[2] because of the presence of the term $-q^2$ in the objective function that is to be minimized. However, note that if the value if q is fixed, this problem becomes a linear program. By using an intelligent sampling approach like a metaheuristic for q and solving a linear program with respect to p and r for each sample of q one can find an approximate optimal solution. Such a decomposition approach breaks the problem into a bilevel optimization structure where the upper level is handled by one optimization algorithm and the lower (nested) level is handled by another optimization algorithm. The second tab in Figure 5 shows how the same problem can be written as a bilevel optimization problem by representing q as u_1, p as l_1 and r as l_2. In our implementation of BOBD, we use a genetic algorithm at the upper level and rely on a convex optimization solver at the lower level. Within the iterations of the genetic algorithm, we classify the variables into upper and lower levels using a machine learning approach.

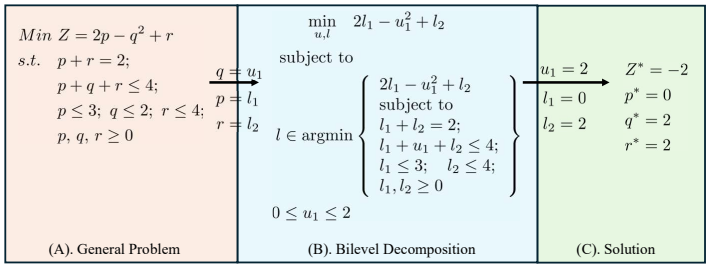


Figure 5: Problem solving using Bilevel optimization-based decomposition approach

[2]Note that despite the example being a non-convex problem it can be solved efficiently by exact methods because of the quadratic nature of the objective function and linear nature of the constraints. However, we have chosen this problem for the ease of discussion.

Note that in module M3, one can use any other approach for solving the optimization problem specified in the PYOMO script. However, we use the BOBD approach because it allows one to capitalize on two mathematical optimization paradigms simultaneously to solve optimization problems, thereby reducing human intervention. Additional implementation details of the BOBD approach and results are provided in Appendix D. The results demonstrate superior performance of BOBD approach in handling a large variety of problems compared to other popular approaches. However, note that optimization methods are evaluated on optimality guarantees and convergence rates, and we do not make any claims of our BOBD implementation being better than any specialized optimization technique on these aspects.

3.4 Performance of AutoOpt Framework

The performance of *AutoOpt* framework is evaluated using two approaches: (i) module-level evaluation and (ii) framework-level evaluation. In module-level evaluation, we consider the individual performance of each module to estimate the reliability of entire framework. For module M1, the Character Error Rate (CER) is 0.0286; hence, reliability of module M1 is estimated as (1-CER)×100 = (1-0.0286)×100 = 97.14%. For module M2, CER rate is 0.0825; hence, reliability of module M2 is estimated as (1-0.0825)×100 = 91.75%. Module M3 contains the optimization solver that solves exactly what is provided in PYOMO script, i.e., there is no prediction task or error-prone task associated with this module. Thus, the reliability or success-rate of entire *AutoOpt* framework can be estimated as (0.9714×0.9175)×100= 89.12%. This estimate is actually a lower bound, as in many cases where the LaTeX or PYOMO is syntactically different from the expected output, the CER metric incorrectly counts such differences as character errors.

In framework-level evaluation, we measure the performance of the complete pipeline (M1–M2–M3) on 500 sample problems outside the *AutoOpt-11k* dataset. The overall success rate (i.e., ability to correctly read the problem in LaTeX and PYOMO and subsequently deploy the solver successfully) was observed to be 94.20%.

4 Conclusions

This study introduces *AutoOpt*, an end-to-end automated framework that enables optimization problem-solving directly from images of mathematical formulations, thereby significantly reducing human intervention. Central to this framework is *AutoOpt-11k*, a curated dataset comprising over 11,554 images of handwritten and typeset mathematical programs, labeled with corresponding LaTeX code for all images and modeling language script for a subset of images. This dataset addresses a longstanding gap in image-based optimization data resources that has the potential to automate optimization problem solving.

The proposed framework consists of three integrated modules: M1 (Image-to-LaTeX), M2 (LaTeX-to-PYOMO), and M3 (Optimization Solver). Each module is powered by custom-developed or fine-tuned deep learning and optimization methods, achieving strong performance across tasks. The deep learning model in M1 outperforms the existing state-of-the-art tools like ChatGPT, Gemini, and Nougat. Additionally, the BOBD method in M3 demonstrates superior performance in solving a wider variety of optimization problems compared to other approaches. By automating the complete pipeline—from image acquisition to solution generation—*AutoOpt* framework offers a powerful and accessible solution for both researchers and practitioners. The public release of the dataset and the framework is expected to encourage future research at the intersection of computer vision, natural language processing, and mathematical optimization. Future research will also address some of the limitations of this study, for instance, handling ill-defined optimization problems effectively, or handling optimization problem definitions that span multiple pages or images.

References

[1] Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Scalable optimization modeling with (mi) lp solvers and large language models. *arXiv preprint arXiv:2402.10172*, 2024.

[2] Shabbir Ahmed and Alexander Shapiro. The sample average approximation method for stochastic programs with integer recourse. *SIAM Journal on Optimization*, 12(2):479–502, 2002.

[3] AMPL. *AMPL: A Modeling Language for Mathematical Programming*. AMPL Optimization Inc., 2023. Version 2023.1.

[4] Neculai Andrei. An unconstrained optimization test functions collection. *Adv. Model. Optim*, 10(1):147–161, 2008.

[5] Mokhtar S Bazaraa, Hanif D Sherali, and Chitharanjan M Shetty. *Nonlinear programming: theory and algorithms*. John wiley & sons, 2006.

[6] John E. Beasley. Or-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[7] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. Robust optimization. *Mathematics of Operations Research*, 34(2):1–38, 2009.

[8] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena scientific Belmont, MA, 1997.

[9] John R Birge and Francois Louveaux. *Introduction to stochastic programming*. Springer Science & Business Media, 2011.

[10] Robert E. Bixby. The zib challenge and the state of mixed-integer programming. *Annals of Operations Research*, 149(1):37–41, 2007.

[11] Jacek Blazewicz, Jan Karel Lenstra, and Alexander H. G. Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

[12] Lukas Blecher, Guillem Cucurull, Thomas Scialom, and Robert Stojnic. Nougat: Neural optical understanding for academic documents. *arXiv preprint arXiv:2308.13418*, 2023.

[13] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[14] Alberto Caprara, Matteo Fischetti, and Paolo Toth. Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371, 2000.

[15] Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021.

[16] COIN-OR Foundation. *CBC User Guide*, 2023. Version 2.10.10, Accessed: 2025-05-02.

[17] COIN-OR Foundation. COIN-OR: Computational infrastructure for operations research, 2023. Accessed: 2025-05-02.

[18] George B Dantzig. *Linear programming and extensions*. Princeton university press, 2016.

[19] Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[20] Kalyanmoy Deb, Ram Bhushan Agrawal, et al. Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148, 1995.

[21] Kalyanmoy Deb and Debayan Deb. Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing*, 4(1):1–28, 2014.

[22] Kalyanmoy Deb, Ankur Sinha, and Saku Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1141–1148, 2006.

[23] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[24] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M Rush. Image-to-markup generation with coarse-to-fine attention. In *International Conference on Machine Learning*, pages 980–989. PMLR, 2017.

[25] Sneha Dhyani Bhatt, Sachin Jayaswal, Ankur Sinha, and Navneet Vidyarthi. Alternate second order conic program reformulations for hub location under stochastic demand and congestion. *Annals of Operations Research*, 304:481–527, 2021.

[26] Max Fehr. Optimization methods in finance by gerard cornuejols, reha tutuncu, 2007.

[27] Roger Fletcher. *Practical methods of optimization*. John Wiley & Sons, 2000.

[28] Christodoulos A Floudas and Panos M Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer, 1990.

[29] Christodoulos A Floudas, Panos M Pardalos, Claire Adjiman, William R Esposito, Zeynep H Gümüs, Stephen T Harding, John L Klepeis, Clifford A Meyer, and Carl A Schweiger. *Handbook of test problems in local and global optimization*, volume 33. Springer Science & Business Media, 2013.

[30] John J. Forrest. Clp user guide, 2005. COIN-OR Linear Programming (Clp) Solver.

[31] GAMS Development Corporation. *General Algebraic Modeling System (GAMS) Documentation*. GAMS Development Corporation, 2023. GAMS Documentation, Version 41.

[32] Philippe Gervais, Asya Fadeeva, and Andrii Maksai. Mathwriting: A dataset for handwritten mathematical expression recognition. *arXiv preprint arXiv:2404.10690*, 2024.

[33] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[34] Andrea Grosso, A. Reza Jamali, and Marco Locatelli. Finding multiple local minima in chemical and biochemical engineering problems. *Computers & Chemical Engineering*, 33(7):1133–1142, 2009.

[35] Bertrand Guenin, Jochen Könemann, and Levent Tuncel. *A gentle introduction to optimization*. Cambridge University Press, 2014.

[36] Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y.K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *arXiv preprint arXiv:2401.14196*, 2024.

[37] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. Gurobi Optimization, LLC, 2025. `https://docs.gurobi.com/projects/optimizer/en/current/index.html`.

[38] William E. Hart, Carl D. Laird, Jean-Paul Watson, David L. Woodruff, Gabriel A. Hackebeil, Bethany L. Nicholson, and John D. Siirola. *Pyomo – Optimization Modeling in Python*. Sandia National Laboratories, 2023. Pyomo Documentation, Version 6.6.2.

[39] David M. Himmelblau. Application of nonlinear programming in chemical engineering. *Chemical Engineering Science*, 41(8):1973–1987, 1986.

[40] Simon Huband, Philip Hingston, Luigi Barone, and Lyndon While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10(5):477–506, 2006.

[41] IBM Corporation. *User's Manual for CPLEX*. IBM, 2025. https://www.ibm.com/docs/en/icos/22.1.2?topic=optimizers-users-manual-cplex.

[42] Eitan Israeli and R. Kevin Wood. Shortest-path network interdiction. *Networks*, 40(2):97–111, 2002.

[43] Stanislav Hristov Ivanov. Automated decision-making. *foresight*, 25(1):4–19, 2023.

[44] JuMP Developers. *JuMP Documentation*. JuMP Community, 2023. JuMP.jl, Version 1.9.

[45] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[46] Jing J Liang, Thomas Philip Runarsson, Efren Mezura-Montes, Maurice Clerc, Ponnuthurai Nagaratnam Suganthan, CA Coello Coello, and Kalyanmoy Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. *Journal of Applied Mechanics*, 41(8):8–31, 2006.

[47] LINDO Systems Inc. *LINDO API User Manual*. LINDO Systems Inc., 2023. Version 14.0.

[48] Fei Liu, Xialiang Tong, Mingxuan Yuan, Xi Lin, Fu Luo, Zhenkun Wang, Zhichao Lu, and Qingfu Zhang. Evolution of heuristics: Towards efficient automatic algorithm design using large language model. *arXiv preprint arXiv:2401.02051*, 2024.

[49] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:726–742, 2020.

[50] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[51] Ladislav Lukšan and Jan Vlcek. Test problems for nonsmooth unconstrained and linearly constrained optimization. Technical report, Technical report, 2000.

[52] Zeyuan Ma, Hongshu Guo, Jiacheng Chen, Guojun Peng, Zhiguang Cao, Yining Ma, and Yue-Jiao Gong. Llamoco: Instruction tuning of large language models for optimization code generation. *arXiv preprint arXiv:2403.01131*, 2024.

[53] Kaj Madsen, Hans Bruun Nielsen, and Ole Tingleff. *Optimization with constraints*. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, 2004.

[54] Brad L Miller, David E Goldberg, et al. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems*, 9(3):193–212, 1995.

[55] MIPLIB. Miplib: Mixed integer programming library, 2023. Accessed: 2025-05-02.

[56] Netlib. Netlib linear programming library, 2023. Accessed: 2025-05-02.

[57] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[58] OR-Library. Or-library: Operations research test problems, 2023. Accessed: 2025-05-02.

[59] Aida Pearson. Aida calculus math handwriting recognition dataset. https://www.kaggle.com/datasets/aidapearson/ocr-data, 2020. Synthetic handwritten calculus math expressions for recognition and OCR tasks.

[60] Zhiliang Peng, Wei Huang, Shanzhi Gu, Lingxi Xie, Yaowei Wang, Jianbin Jiao, and Qixiang Ye. Conformer: Local features coupling global representations for visual recognition. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 367–376, 2021.

[61] Michael Pinedo and Xiuli Chao. Operations scheduling with applications in manufacturing and services. *International Transactions in Operational Research*, 6(5):441–453, 1999.

[62] Prasanna Ramamoorthy, Sachin Jayaswal, Ankur Sinha, and Navneet Vidyarthi. Multiple allocation hub interdiction and protection problems: Model formulations and solution approaches. *European Journal of Operational Research*, 270(1):230–245, 2018.

[63] Singiresu S. Rao. *Engineering Optimization: Theory and Practice*. John Wiley & Sons, Hoboken, NJ, 4 edition, 2009.

[64] UCI Machine Learning Repository. Uci machine learning repository, 2023. Accessed: 2025-05-02.

[65] R Clark Robinson. *Introduction to mathematical optimization*. Department of Mathematics, Northwestern University, Illinois US, 2013.

[66] Felix M Schmitt-Koopmann, Elaine M Huang, and Alireza Darvishy. Accessible pdfs: applying artificial intelligence for automated remediation of stem pdfs. In *Proceedings of the 24th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–6, 2022.

[67] Felix M Schmitt-Koopmann, Elaine M Huang, Hans-Peter Hutter, Thilo Stadelmann, and Alireza Darvishy. Mathnet: a data-centric approach for printed mathematical expression recognition. *IEEE Access*, 2024.

[68] NEOS Server. Neos server: The optimization server, 2023. Accessed: 2025-05-02.

[69] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Test problem construction for single-objective bilevel optimization. *Evolutionary computation*, 22(3):439–477, 2014.

[70] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. Towards understanding bilevel multi-objective optimization with deterministic lower level decisions. In *Proceedings of the Eighth International Conference on Evolutionary Multi-Criterion Optimization (EMO-2015)*. Berlin, Germany: Springer-Verlag, 2015.

[71] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: From classical to evolutionary approaches and applications. *IEEE transactions on evolutionary computation*, 22(2):276–295, 2017.

[72] Ankur Sinha, Dhaval Pujara, and Hemant Kumar Singh. Decomposition of difficulties in complex optimization problems using a bilevel approach. *arXiv preprint arXiv:2407.03454*, 2024.

[73] James C. Smith and Yinyu Song. A survey of network interdiction models and algorithms. *European Journal of Operational Research*, 201(3):1–14, 2008.

[74] Ray Smith. An overview of the tesseract ocr engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633, 2007.

[75] James C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley-Interscience, Hoboken, NJ, 2003.

[76] George Stephanopoulos and Arthur W Westerberg. The use of hestenes' method of multipliers to resolve dual gaps in engineering system optimization. *Journal of Optimization Theory and Applications*, 15:285–309, 1975.

[77] Gilbert Syswerda. A study of reproduction in generational and steady-state genetic algorithms. In *Foundations of genetic algorithms*, volume 1, pages 94–101. Elsevier, 1991.

[78] Luis N. Vicente and Paul H. Calamai. Bilevel and multilevel programming: a bibliography review. *Journal of Global Optimization*, 5:291–306, 1994.

[79] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

[80] Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 2024.

[81] Andreas Wächter and Lorenz T Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[82] Yejing Xie, Harold Mouchère, Foteini Simistira Liwicki, Sumit Rakesh, Rajkumar Saini, Masaki Nakagawa, Cuong Tuan Nguyen, and Thanh-Nghia Truong. Icdar 2023 crohme: Competition on recognition of handwritten mathematical expressions. In *Document Analysis and Recognition - ICDAR 2023*, volume 14234 of *Lecture Notes in Computer Science*, pages 541–551. Springer Nature Switzerland, 2023.

[83] Norm Xu. Nougat-latex-ocr: Fine-tuning and evaluation of nougat-based image-to-latex models, 2025. Accessed: 2025-05-08.

[84] Shunyu Yao, Fei Liu, Xi Lin, Zhichao Lu, Zhenkun Wang, and Qingfu Zhang. Multi-objective evolution of heuristic using large language model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):27144–27152, 2025.

[85] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation*, 8(2):173–195, 2000.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and the introduction clearly represent the contribution and scope of the paper.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper briefly discusses the limitations of the work in the conclusions section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not involve theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All the information about the experimental setup that will allow reproducibility have been provided in great detail in the main paper and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The dataset and the code have been made public. Necessary documentation and details are also provided for the ease of readability and reproducibility.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All these details are provided in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Error bars, where applicable, are reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All the details have been provided in the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and in our best judgement there are no deviations.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: This work is on automating optimization problem solving. The automation would benefit researchers and practitioner and we do not see any negative societal impact of our work. The benefits have been discussed in the paper at multiple places.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: There are no risks posed by the data and models developed in this study.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All the sources that have been used in or have inspired the creation of the dataset are properly credited in the paper. The dataset is being released under the CC by NC-SA 4.0 license.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: In this paper, we have created a dataset using ethical guidelines and have cited all sources that have been used or have inspired the creation of the dataset. All the details of the dataset and the models created have been provided in great detail in the paper.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The dataset in this study has been created by hiring 25 experts who worked independently and collaboratively under the guidance of the authors. There are no human subjects involved and the dataset created is not by crowdsourcing.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: No human subjects were involved in this study.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [No]

    Justification: The LLM is used only for writing, editing or formatting purposes.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

A Appendix: Dataset

Figures 6, 7, and 8 offer deeper insights into the structure and content of the *AutoOpt-11k* dataset developed in this study. These visualizations collectively help characterize the dataset along multiple dimensions—expression complexity, comparative scale, and token diversity.

Figure 6 presents a histogram of LaTeX expression lengths, indicating the number of samples corresponding to different expression sizes. This figure is particularly useful for understanding the distribution of expression complexity in our dataset. Unlike many datasets that contain shorter and simpler expressions, our dataset encompasses a broad range of lengths, including a substantial number of longer and more elaborate expressions. This distribution is indicative of real-world mathematical programs. Figure 7 provides a comparative analysis of LaTeX expression lengths across multiple publicly available datasets [59, 82, 24, 32] for mathematical expression recognition. It is evident from the figure that our dataset distinguishes itself by including a significantly higher proportion of longer, multi-line expressions. This unique characteristic enhances its applicability to practical use cases that require parsing long mathematical expressions. Figure 8 illustrates the 100 most frequent LaTeX tokens in our dataset, underscoring its syntactic richness and diversity. The tokens span a wide range of categories, including comparison operators, set theory notations, mathematical operators, syntactic elements, Latin letters, numbers, Blackboard capital letters, Greek symbols, mathematical constructs, modifiers, matrix environments, delimiters, arrows, dots, punctuations and various other symbols. This token diversity confirms the dataset's relevance for training models that must generalize across diverse types of notation.

Furthermore, Table 6 and Table 7 provide concrete examples from the dataset, showcasing images of optimization formulations along with their corresponding LaTeX and PYOMO representations. These examples demonstrate the alignment between visual representations and their semantic counterparts, highlighting the dataset's utility for a variety of tasks. Together, the figures and tables substantiate the comprehensiveness and quality of our dataset, validating its potential to support robust training and evaluation of machine learning models targeting advanced mathematical understanding and code generation tasks.

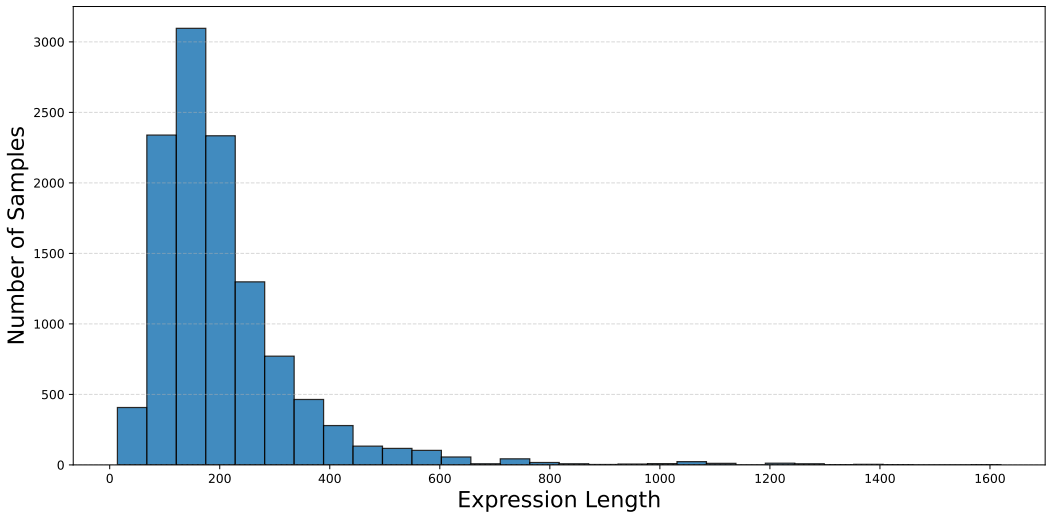


Figure 6: Histogram of LaTeX expression lengths

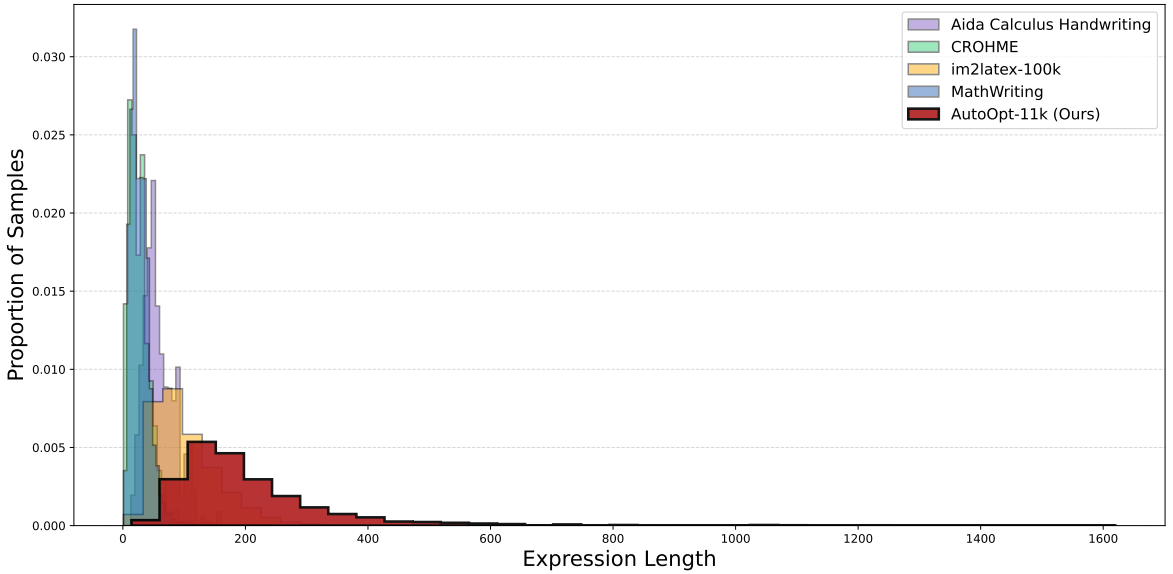


Figure 7: Normalized comparison of LaTeX expression lengths

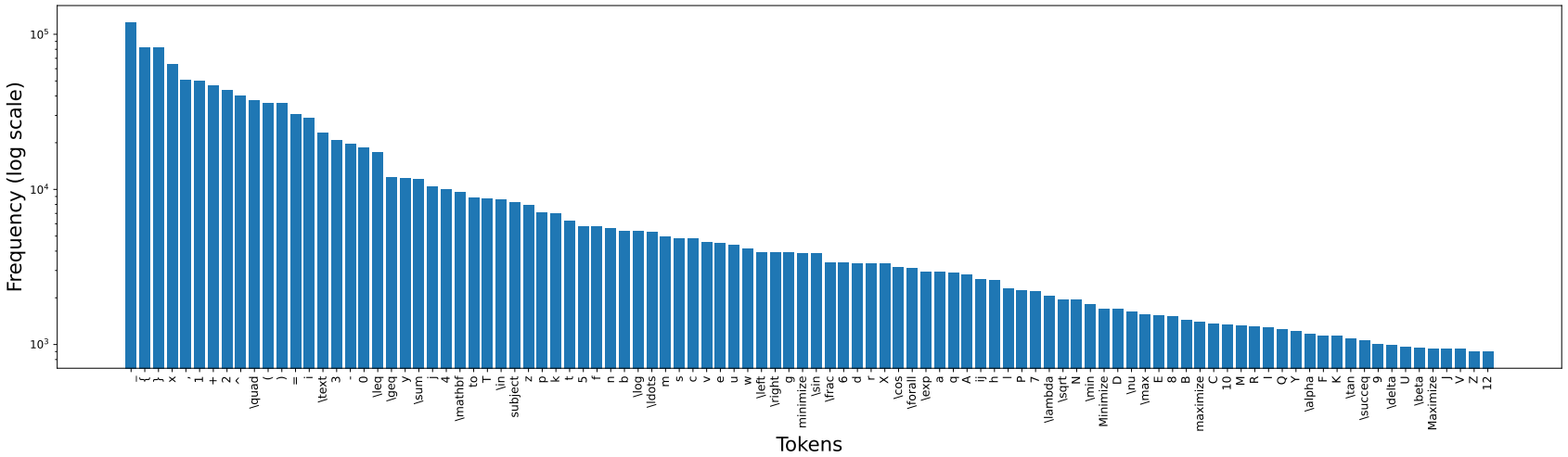


Figure 8: Top 100 most frequent tokens in LaTeX

Table 6: Samples from *AutoOpt-11k* dataset: Images and Labels

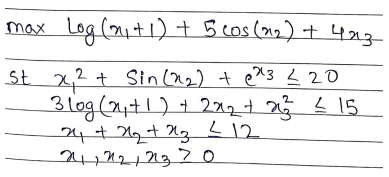
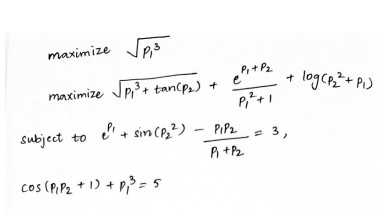
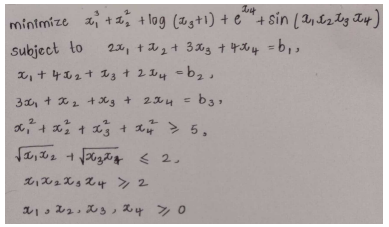
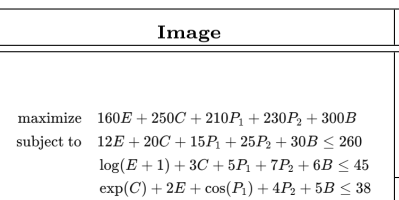
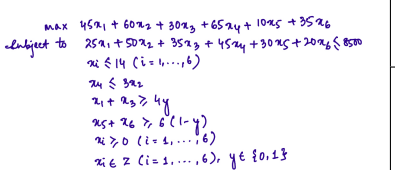
Image	LaTex and PYOMO
	\text{max} \quad & \log(x_1 + 1) + 5 \cos(x_2) + 4x_3 \\ \text{st} \quad & x_1^2 + \sin(x_2) + e^{x_3} \leq 20 \\ & 3 \log(x_1 + 1) + 2x_2 + x_3^2 \leq 15 \\ & x_1 + x_2 + x_3 \leq 12 \\ & x_1, x_2, x_3 \geq 0.
	model.x1 = Var(within=NonNegativeReals)\nmodel.x2 = Var(within=NonNegativeReals)\nmodel.x3 = Var(within=NonNegativeReals)\n def objective_function(model):\n return log(x1 + 1) + 5*cos(x2) + 4*x3\nmodel.OF = Objective(rule=objective_function,sense=maximize) \nmodel.Constraint1 = Constraint(expr = x1**2 + sin(x2) + exp(x3) - 20 <= 0)\nmodel.Constraint2 = Constraint(expr = 3*log(x1 + 1) + 2*x2 + x3**2 - 15 <= 0)\nmodel.Constraint3 = Constraint(expr = x1 + x2 + x3 - 12 <= 0)
	\text{maximize} \quad & \sqrt{p_1^3 + \tan(p_2)} + \frac{e^{p_1 + p_2}}{p_1^2 + 1} + \log(p_2^2 + p_1) \\ \text{subject to} \quad & e^{p_1} + \sin(p_2^2) - \frac{p_1 p_2}{p_1 + p_2} = 3, \\ & \cos(p_1 p_2 + 1) + p_1^3 = 5
	model.p1 = Var()\nmodel.p2 = Var()\ndef objective_rule(model):\n return sqrt(p1**3 + tan(p2)) + exp(p1 + p2)/(p1**2 + 1) + log(p2**2 + p1)\n model.objective = Objective(rule=objective_rule, sense=maximize)\nmodel. Constraint1 = Constraint(expr = exp(p1) + sin(p2**2) - (p1*p2)/(p1 + p2) == 3)\n model.Constraint2 = Constraint(expr = cos(p1*p2 + 1) + p1**3 == 5)
Minimize $f(x_1, x_2) = \log(x_1) + \log(x_2)$ subject to $g_1(x_1, x_2) = x_1 + x_2^2 - 7 \leq 0$ $g_2(x_1, x_2) = x_1^2 + x_2 - 3 \leq 0$	& \text{Minimize} \quad f(x_1, x_2) = \log(x_1) + \log(x_2) \\ & \text{subject to} \\ & \quad g_1(x_1, x_2) = x_1 + x_2^2 - 7 \leq 0 \\ & \quad g_2(x_1, x_2) = x_1^2 + x_2 - 3 \leq 0
	model.x1 = Var(within=PositiveReals)\nmodel.x2 = Var(within=PositiveReals)\ndef objective_function(model):\n return log(x1) + log(x2)\nmodel.obj = Objective(rule=objective_function, sense=minimize)\nmodel.Constraint1 = Constraint(expr = x1 + x2**2 <= 7)\n model.Constraint2 = Constraint(expr = x1**2 + x2 <= 3)
	\text{minimize} \quad & x_1^3 + x_2^2 + \log(x_3 + 1) + e^{x_4} + \sin(x_1 x_2 x_3 x_4) \\ \text{subject to} \quad & 2x_1 + x_2 + 3x_3 + 4x_4 = b_1, \\ & x_1 + 4x_2 + x_3 + 2x_4 = b_2, \\ & 3x_1 + x_2 + x_3 + 2x_4 = b_3, \\ & x_1^2 + x_2^2 + x_3^2 + x_4^2 \geq 5, \\ & \sqrt{x_1 x_2} + \sqrt{x_3 x_4} \leq 2, \\ & x_1 x_2 x_3 x_4 \geq 2, \\ & x_1, x_2, x_3, x_4 \geq 0
	model.x1 = Var(within=NonNegativeReals)\nmodel.x2 = Var(within=NonNegativeReals)\nmodel.x3 = Var(within=NonNegativeReals)\n model.x4 = Var(within=NonNegativeReals)\ndef objective_rule(model):\n return x1**3 + x2**2 + log(x3 + 1) + exp(x4) + sin(x1 * x2 * x3 * x4)\nmodel.objective = Objective(rule=objective_rule, sense=minimize)\nmodel.Constraint1 = Constraint(expr= 2*x1 + x2 + 3*x3 + 4*x4 == b1)\nmodel.Constraint2 = Constraint(expr = x1 + 4*x2 + x3 + 2*x4 == b2)\nmodel.Constraint3 = Constraint(expr= 3*x1 + x2 + x3 + 2*x4 == b3)\nmodel.Constraint4 = Constraint(expr = x1**2 + x2**2 + x3**2 + x4**2 >= 5)\n model.Constraint5 = Constraint(expr= sqrt(x1 * x2) + sqrt(x3 * x4) <= 2)\nmodel.Constraint6 = Constraint(expr= x1 * x2 * x3 * x4 >= 2)

Table 7: Samples from *AutoOpt-11k* dataset: Images and Labels (Cont.)

Image	LaTex and PYOMO
	`\text{maximize} & 160E + 250C + 210P_1 + 230P_2 + 300B \\ \text{subject to} & 12E + 20C + 15P_1 + 25P_2 + 30B \leq 260 \\ & \log(E + 1) + 3C + 5P_1 + 7P_2 + 6B \leq 45 \\ & \exp(C) + 2E + \cos(P_1) + 4P_2 + 5B \leq 38 \\ & 6E + 7C + 8P_1 + 9P_2 + 10B \leq 50 \\ & 4E + 6C + 2P_1 - 3P_2 + 4B \leq 18 \\ & P_1 - P_2 = -6 \\ & E, C, P_1, P_2, B \geq 0.`  `model.E = Var(within=NonNegativeReals)\nmodel.C = Var(within=NonNegativeReals)\nmodel.P1 = Var(within=NonNegativeReals)\nmodel.P2 = Var(within=NonNegativeReals)\nmodel.B = Var(within=NonNegativeReals)\ndef objective_function(model):\n return 160*E + 250*C + 210*P1 + 230*P2 + 300*B\nmodel.obj = Objective(rule=objective_function, sense=maximize)\nmodel.Constraint1 = Constraint(expr = 12*E + 20*C + 15*P1 + 25*P2 + 30*B <= 260)\nmodel.Constraint2 = Constraint(expr = log(E + 1) + 3*C + 5*P1 + 7*P2 + 6*B <= 45)\nmodel.Constraint3 = Constraint(expr = exp(C) + 2*E + cos(P1) + 4*P2 + 5*B <= 38)\nmodel.Constraint4 = Constraint(expr = 6*E + 7*C + 8*P1 + 9*P2 + 10*B <= 50)\nmodel.Constraint5 = Constraint(expr = 4*E + 6*C + 2*P1 - 3*P2 + 4*B <= 18)\nmodel.Constraint6 = Constraint(expr = P1 - P2 == -6)`
	`\text{max} \quad & 45x_1 + 60x_2 + 30x_3 + 65x_4 + 10x_5 + 35x_6 \\ \text{subject to} \quad & 25x_1 + 50x_2 + 35x_3 + 45x_4 + 30x_5 + 20x_6 \leq 8500 \\ & x_i \leq 14 \quad (i = 1, \ldots, 6) \\ & x_4 \leq 3x_2 \\ & x_1 + x_3 \geq 4y \\ & x_5 + x_6 \geq 6(1 - y) \\ & x_i \geq 0 \quad (i = 1, \ldots, 6) \\ & x_i \quad \text{integer} \quad (i = 1, \ldots, 6) \\ & y \in \{0, 1\}.`  `model.x1 = Var(within=NonNegativeReals, domain=Integers)\\nmodel.x2 = Var(within=NonNegativeReals, domain=Integers)\\nmodel.x3 = Var(within=NonNegativeReals, domain=Integers)\\nmodel.x4 = Var(within=NonNegativeReals, domain=Integers)\\nmodel.x5 = Var(within=NonNegativeReals, domain=Integers)\\nmodel.x6 = Var(within=NonNegativeReals, domain=Integers)\\nmodel.y = Var(domain=Binary)\\n def objective_function(model):\\n return 45*x1 + 60*x2 + 30*x3 + 65*x4 + 10*x5 + 35*x6\\nmodel.OF = Objective(rule=objective_function, sense=maximize)\\nmodel.Constraint1 = Constraint(expr = 25*x1 + 50*x2 + 35*x3 + 45*x4 + 30*x5 + 20*x6 <= 8500)\\nmodel.Constraint2 = Constraint(expr = x1 <= 14)\\nmodel.Constraint3 = Constraint(expr = x2 <= 14)\\nmodel.Constraint4 = Constraint(expr = x3 <= 14)\\nmodel.Constraint5 = Constraint(expr = x4 <= 14)\\nmodel.Constraint6 = Constraint(expr = x5 <= 14)\\nmodel.Constraint7 = Constraint(expr = x6 <= 14)\\nmodel.Constraint8 = Constraint(expr = x4 <= 3*x2)\\nmodel.Constraint9 = Constraint(expr = x1 + x3 >= 4*y)\\nmodel.Constraint10 = Constraint(expr = x5 + x6 >= 6*(1 - y))`

Minimize $f(x_1, x_2) = \sqrt{x_1} + \sqrt{x_2}$
subject to
$x_1 + 2x_2 \leq 13$
$1 \leq x_i \leq 9, \quad i = 1, 2$
	`\text{Minimize} \quad f(x_1, x_2) = \sqrt{x_1} + \sqrt{x_2} \\ & \text{subject to} \\ & \quad x_1 + 2x_2 \leq 13 \\ & \quad 1 \leq x_i \leq 9, \quad i = 1, 2`
	
	`model.x1 = Var(bounds=(1,9))\nmodel.x2 = Var(bounds=(1,9))\ndef objective_function(model):\n  return sqrt(x1) + sqrt(x2)\nmodel.obj = Objective(rule=objective_function, sense=minimize)\nmodel.Constraint1 = Constraint(expr = x1 + 2*x2 <= 13)`

B Appendix: Module M1

We did 5 runs for AutoOpt-M1, Nougat, ChatGPT, and Gemini by randomly splitting the data with a training, validation, and test split of 80%, 10% and 10%, respectively. The models corresponding to the median BLUE score for AutoOpt-M1, Nougat, ChatGPT and Gemini are reported in Section 3.1.1. The standard deviations in BLUE score and Character Error Rate are reported in Table 8 and Table 9, respectively.

Table 8: Standard deviation of BLUE score from 5 runs

Model	**HW**	**PR**	**HW+PR**
GPT-4o	0.76	0.48	0.52
Gemini 2.0 Flash	0.88	0.51	1.18
Nougat	1.16	0.8	1.07
AutoOpt-M1	1.14	0.87	1.04

Table 9: Standard deviation of Character Error Rate from 5 runs

Model	**HW**	**PR**	**HW+PR**
GPT-4o	0.0068	0.0084	0.0032
Gemini 2.0 Flash	0.0224	0.0054	0.0113
Nougat	0.0098	0.0087	0.0058
AutoOpt-M1	0.0122	0.0086	0.0072

All experiments were conducted on Google Colab Pro using NVIDIA A100 GPU. The AutoOpt-M1 model was trained for 180 epochs. We used AdamW optimizer with learning rate $2e^{-5}$, weight decay 0.02, and with a cosine scheduler. The batch size was set to 8 with gradient accumulation of 2. Each epoch took approximately 15-20 minutes. Figure 9 shows the convergence plot for the model for a particular run.

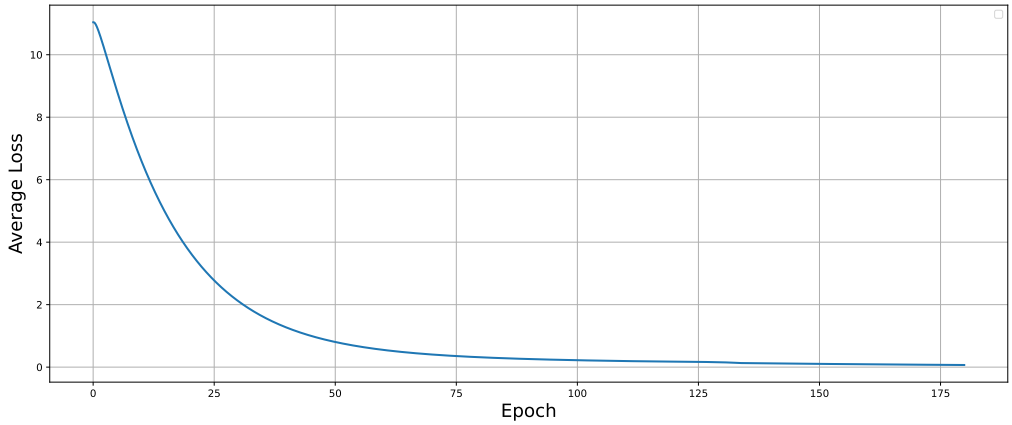


Figure 9: Convergence plot for AutoOpt-M1

C Appendix: Module M2

The median BLEU score and median Character Error Rate (CER) correspond to the same run among the five runs we performed. The median scores are reported in Section 3.2. The standard deviations for BLUE score (in percent) and Character Error Rate over 5 runs are 1.08 and 0.0051, respectively.

We fine-tuned the model for 15 epochs on Google Colab Pro using an NVIDIA A100 GPU with mixed-precision (fp16). For training, the batch size was set to 2 with gradient accumulation of 4 (effective batch size 8). The learning rate was set to $5e^{-5}$ with a weight decay of 0.01. Each run took approximately 30–35 minutes.

D Appendix: Module M3

In this study, we consider a Bilevel Optimization based Decomposition (BOBD) method [72] for optimization task in module M3 (Section 3.3). BOBD method offers the advantages of both exact and approximation methods, by providing efficient solutions within a reasonable time frame. BOBD method attains this advantage by using a bilevel optimization structure that allows it to simultaneously use exact and approximation methods for problem solving. To explain the working procedure of the BOBD method, we first review the structures of general and bilevel optimization problems based on their formal definitions.

Definition 1. *A general optimization problem can be represented using its basic elements, decision variables $x = (x_1, \ldots, x_n)$, objective function $F(x)$, and constraints $G(x)$ and $H(x)$, as follows:*

$$\min_{x} \quad F(x) \tag{1}$$

$$\textit{subject to} \quad G_i(x) \leq 0, \quad i = 1, \ldots, I \tag{2}$$

$$H_j(x) = 0, \quad j = 1, \ldots, J \tag{3}$$

Bilevel optimization problem is characterized by a unique structure in which the primary or upper level optimization problem contains an additional optimization problem, lower level optimization problem, nested within it as a constraint [78, 70, 71].

Definition 2. *A bilevel optimization problem, with upper level and lower level decision variables (u and l), objective functions ($F(u,l)$ and $f(u,l)$), and constraints ($G(u,l)$ & $H(u,l)$ and $g(u,l)$ & $h(u,l)$), can be represented as follows:*

$$\min_{u,l} \quad F(u,l) \tag{4}$$

$$\textit{subject to}$$

$$l \in \underset{l}{\operatorname{argmin}}\{f(u,l) : g_p(u,l) \leq 0, \quad p = 1, \ldots, P,$$

$$h_q(u,l) = 0, \quad q = 1, \ldots, Q\} \tag{5}$$

$$G_i(u,l) \leq 0, \quad i = 1, \ldots, I \tag{6}$$

$$H_j(u,l) = 0, \quad j = 1, \ldots, J \tag{7}$$

To solve a bilevel problem (Definition 2) in a nested manner, the values of upper level variables u are fixed, and the lower level problem is solved with respect to l. By intelligently sampling u and solving the lower level problem repeatedly, it is possible to converge to the bilevel optimum.

BOBD method involves the variable classification task, in which we classify each decision variable ($x_i \in x;\ i = 1...n$) into upper level (u) or lower level (l) variables category. It allows to express the general optimization problem (Definition 1) in the form of bilevel optimization problem (Definition 2), which is called a bilevel decomposition process. In this study, we develop a Logistic Regression based Variable Classification Model (LR-VCM), a method to perform the variable classification task. To begin with, a population is generated and all the variables are initialized randomly. Thereafter, we start with a random approach of variable classification, where each decision variable is classified randomly into upper level (tag 0) or lower level (tag 1). Every single trial of level selection for all variables is referred to as Level Configuration (LC), and the collection of corresponding tag values constitutes a single observation for logistic regression. For given LC, we evaluate if it leads to an improvement in the objective function, when the lower level problem is solved. The improvement results are recorded on a binary scale: 0 indicates little improvement, and 1 indicates notable improvement. This binary score (0 or 1) acts as a label for a given LC. A pair of LC (observation) and corresponding improvement score (label) creates a single training sample for LR-VCM. We construct a training dataset by repeating the same procedure for the entire population. We perform a logistic regression using the developed training set and classify variables with statistically significant p-values into lower level and the remaining variables into upper level. After variable classification (u,l), we intelligently sample the values of upper level variables u using a genetic algorithm, and for each sample, the corresponding lower level problem is solved using a suitable classical optimization method, which provides

the values of lower level variables l. The obtained values of upper level and lower level variables yield a particular solution for a given bilevel problem. This procedure is repeated several times to generate multiple solutions and an efficient solution (u^*, l^*) is then identified from the generated solutions, as outlined in the pseudocode provided in Algorithm 1.

Algorithm 1 Bilevel Optimization-based Decomposition (BOBD)

Input:	$F(x)$, $G(x)$, $H(x)$: single level optimization problem (i.e., original problem)
Output:	$x^* = (u^*, l^*)$: the best solution found for single level optimization problem
Step 1:	Generate a population ($\mathcal{P}$) of random initial solutions.
Step 2:	Develop a Logistic Regression based Variable Classification Model (LR-VCM).
Step 3:	Perform a bilevel decomposition of original problem into upper level and lower level using LR-VCM.
Step 4:	*for* $g = 1$ to *number_of_generations*:
Step 5:	If g is divisible by *variable_classification_alternation_number* (C=10):
Step 6:	Develop a new LR-VCM using updated dataset.
Step 7:	Perform a new bilevel decomposition of original problem.
Step 8:	Sample the values of upper level variables u using genetic algorithm.
Step 9:	For a given u, obtain l by solving the corresponding lower level problem using the interior point or the linear programming methods.
Step 10:	Update population $\mathcal{P}$ with new solutions if they are better than the worst solutions in the population.

We bring novelty to BOBD method by incorporating LR-VCM for variable classification task. To evaluate the performance of the BOBD method, we consider a test suite of 10 optimization Test Problems (TP), TP1-TP10 [72]. These test problems are derived from the real-world applications and exhibit various types of complexities such as non-convexity, non-linearity, non-differentiability, discreteness, high-dimensionality, etc. The description of test problems (TP1-TP10) and computational experiments are discussed next.

TP1 (Structural Sensitivity Problem in a Chemical System [76]):

$$
\begin{aligned}
\min_{x} \quad & F(x) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} - 4x_3 + 2x_4 + 5x_5 - x_6 \\
\text{s.t.} \quad & x_2 - 3x_1 - 3x_4 = 0; \\
& x_3 - 2x_2 - 2x_5 = 0; \\
& 4x_4 - x_6 = 0; \\
& x_1 + 2x_4 \leq 4; \\
& x_2 + x_5 \leq 4; \\
& x_3 + x_6 \leq 6; \\
& x_1 \leq 3;\ x_3 \leq 4;\ x_5 \leq 2;\ x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}
$$

TP2 (Heat Exchanger Design Problem [46]):

$$
\begin{aligned}
\min_{x} \quad & F(x) = x_1 \\
\text{s.t.} \quad & 35x_2^{0.6} + 35x_3^{0.6} - x_1 \leq 0; \\
& -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0; \\
& 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0; \\
& -x_5 + \ln(-x_4 + 900) = 0; \\
& -x_6 + \ln(x_4 + 300) = 0; \\
& -x_7 + \ln(-2x_4 + 700) = 0; \\
& 0 \leq x_1 \leq 1000; \quad 0 \leq x_2, x_3 \leq 40; \quad 100 \leq x_4 \leq 300; \\
& 6.3 \leq x_5 \leq 6.7; \quad 5.9 \leq x_6 \leq 6.4; \quad 4.5 \leq x_7 \leq 6.25
\end{aligned}
$$

TP3 (More complexities added to TP1 [72]):

$$\begin{aligned}
\min_{x} \quad & F(x) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} - 4x_3 + 2x_4 + 5x_5 - x_6 + \frac{x_3^2}{16} - 2\cos(2\pi x_2) \\
\text{s.t.} \quad & x_2 - 3x_1 - 3x_4 = 0; \\
& x_3 - 2x_2 - 2x_5 = 0; \\
& 4x_4 - x_6 = 0; \\
& x_1 + 2x_4 \leq 4; \\
& x_2 + x_5 \leq 4; \\
& x_3 + x_6 \leq 6; \\
& x_1 \leq 3; \quad x_5 \leq 2; \quad x_3 \leq 4; \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}$$

TP4 (Scalable variables y and constraints added to TP3 [72]):

$$\begin{aligned}
\min_{x} \quad & F(x) = x_1^{0.6} + x_2^{0.6} + x_3^{0.4} - 4x_3 + 2x_4 + 5x_5 - x_6 + \frac{x_3^2}{16} - \frac{x_2^2}{16} \\
& \qquad - 2\cos(2\pi x_3) - 2\cos(2\pi x_2) + \sum_{p=1}^{P} y_p \cdot x_1^{0.6} \\
\text{s.t.} \quad & x_2 - 3x_1 - 3x_4 = 0; \\
& x_3 - 2x_2 - 2x_5 = 0; \\
& x_1 + 2x_4 \leq 4; \\
& x_2 + x_5 \leq 4; \\
& x_3 + x_6 \leq 6; \\
& \sqrt{x_1 + x_2 + x_3} - y_p \leq 0, \quad \forall p; \\
& x_1 \leq 3; \quad x_5 \leq 2; \quad x_3 \leq 4; \\
& 1 \leq y_p \leq 5, \, \forall p; \\
& x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{aligned}$$

TP5 (Scalable variables y and constraints added to TP2 [72]):

$$\begin{aligned}
\min_{x} \quad & F(x) = x_1 - 50\cos(2\pi x_4) + \sum_{p=1}^{P} \frac{{y_p}^2}{x_4} \\
\text{s.t.} \quad & 35x_2^{0.6} + 35x_3^{0.6} - x_1 \leq 0; \\
& -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0; \\
& 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0; \\
& -x_5 + \ln(-x_4 + 900) = 0; \\
& -x_6 + \ln(x_4 + 300) = 0; \\
& -x_7 + \ln(-2x_4 + 700) = 0; \\
& x_4^{0.2} + x_5 + x_6 - y_p \leq 0, \ \forall p; \\
& 0 \leq x_1 \leq 1000; \quad 0 \leq x_2, x_3 \leq 40; \quad 100 \leq x_4 \leq 300; \\
& 6.3 \leq x_5 \leq 6.7; \quad 5.9 \leq x_6 \leq 6.4; \quad 4.5 \leq x_7 \leq 6.25; \quad 10 \leq y_p \leq 30, \, \forall p
\end{aligned}$$

TP6 (Scalable variables (y, z) and constraints added to existing problem [28]):

$$\begin{aligned}
\min_{x} \quad & F(x) = -25\left(x_1-2\right)^2 - \left(x_2-2\right)^2 - \left(x_3-1\right)^2 - \left(x_4-4\right)^2 - \left(x_5-1\right)^2 - \left(x_6-4\right)^2 \\
& \qquad + \sum_{p=1}^{P}\left(x_3-y_p\right)^2 - \sum_{\mathrm{q}=1}^{Q}\left(x_5-z_q\right)^2 \\
\text{s.t.} \quad & -\left(x_3-3\right)^2 - x_4 + 4 \leq 0; \\
& -\left(x_5-3\right)^2 - x_6 + 4 \leq 0; \\
& -x_1 - x_2 + 2 \leq 0; \\
& x_1 - 3x_2 \leq 2; \\
& x_2 - x_1 \leq 2; \\
& x_1 + x_2 \leq 6; \\
& y_p - x_3 + 1 \leq 0, \ \forall p; \\
& z_q^2 - x_3^2 - x_5^2 \leq 0, \ \forall q; \\
& 0 \leq x_1; \quad 0 \leq x_2; \quad 1 \leq x_3 \leq 5; \quad 0 \leq x_4 \leq 6; \\
& 1 \leq x_5 \leq 5; \quad 0 \leq x_6 \leq 10; \quad 0 \leq y_p \leq 5, \forall p; \quad 0 \leq z_q \leq 5, \forall q
\end{aligned}$$

TP7 (Pool blending Problem with additional complexities [46]):

$$\begin{aligned}
\min_{x} \quad & F(x) = 6x_1 + 16x_2 - 9x_5 + 10\left(x_6+x_7\right) - 15x_8 + x_9^2 + 50\cos\left(\pi x_9\right) - 25\cos\left(\pi x_8\right) \\
& \qquad - \ln\left(x_8-x_9\right) - \sum_{p=1}^{P}\left(y_p-x_9\right)^2 + \sum_{q=1}^{Q}\left(z_q-x_8\right)^2 \\
\text{s.t.} \quad & x_1 + x_2 - x_3 - x_4 = 0; \\
& x_3 + x_6 - x_5 = 0; \\
& x_4 + x_7 - x_8 = 0; \\
& 0.03x_1 + 0.01x_2 - x_3x_9 - x_4x_9 = 0; \\
& x_3x_9 + 0.02x_6 - 0.025x_5 \leq 0; \\
& x_4x_9 + 0.02x_7 - 0.015x_8 \leq 0; \\
& x_9^2 - y_p^2 \leq 0, \ \forall p; \\
& x_8^2 - z_q^2 \leq 0, \ \forall q; \\
& 0 \leq x_1, x_2, x_6 \leq 300; \quad 0 \leq x_3, x_5, x_7 \leq 100; \quad 0 \leq x_4, x_8 \leq 200; \\
& 0.01 \leq x_9 \leq 0.03; \quad 0 \leq y_p \leq 1, \forall p; \quad 1 \leq z_q \leq 200, \forall q
\end{aligned}$$

TP8 (Existing problem with additional complexities [72]):

$$\begin{aligned}
\min_{x} \quad & F(x) = 5\sum_{i=1}^{4}x_i - 5\sum_{i=1}^{4}x_i^2 - \sum_{i=5}^{13}x_i - 20e^{-0.1\sqrt{\sum_{i=1}^{4}x_i^2}} - e^{0.25\sum_{\mathrm{i}=1}^{4}\cos(2\pi x_i)} \\
& \qquad + \sum_{p=1}^{P}\left(y_p^2 + \sum_{\mathrm{i}=1}^{4}\cos\left(2\pi x_i\right)\right) \\
\text{s.t.} \quad & 2x_1 + 2x_2 + x_{10} + x_{11} \leq 10; \\
& 2x_1 + 2x_3 + x_{10} + x_{12} \leq 10; \\
& 2x_2 + 2x_3 + x_{11} + x_{12} \leq 10;
\end{aligned}$$

$$
\begin{aligned}
& x_{10} - 8x_1 \leq 0; \\
& x_{11} - 8x_2 \leq 0; \\
& x_{12} - 8x_3 \leq 0; \\
& x_{10} - x_5 - 2x_4 \leq 0; \\
& x_{11} - x_7 - 2x_6 \leq 0; \\
& x_{12} - x_9 - 2x_8 \leq 0; \\
& \sum_{\mathrm{i}=1}^{4} \cos\left(2\pi x_i\right) - y_p \leq 0, \ \forall p; \\
& 0 \leq x_i \leq 3 \ \ (i = 1, \ldots, 4); \quad 0 \leq x_i \leq 1 \ \ (i = 5, \ldots, 9); \\
& 0 \leq x_i \leq 100 \ \ (i = 10, 11, 12); \quad 0 \leq x_{13} \leq 1; \quad -5 \leq y_p \leq 5, \forall p
\end{aligned}
$$

TP9 (Heat Exchanger Design Problem with additional complexities [46]):

$$
\begin{aligned}
\min_{x} \quad & F(x) = x_1 + x_2 + x_3 + \sum_{p=1}^{P} \left(\tan\left(y_p\right) - 15\cos 2\pi\left(x_1 + x_2 + x_3\right)\right)^2 \\
\text{s.t.} \quad & 0.0025x_4 + 0.0025x_6 \leq 1; \\
& 0.0025x_5 + 0.0025x_7 - 0.0025x_4 \leq 1; \\
& 0.01x_8 - 0.01x_5 \leq 1; \\
& -x_1x_6 + 100x_1 + 833.33x_4 \leq 83333.33; \\
& -x_2x_7 + 1250x_5 - 1250x_4 + x_2x_4 \leq 0; \\
& -x_3x_8 - 2500x_5 + x_3x_5 + 1250000 \leq 0; \\
& \tan\left(y_p\right) - \ln\left(x_1 + x_2 + x_3\right) \leq 0 \quad \forall p; \\
& -\tan\left(y_p\right) - \ln\left(x_1 + x_2 + x_3\right) \leq 0 \quad \forall p; \\
& 100 \leq x_1 \leq 10000; \quad 1000 \leq x_i \leq 10000 \ \ (i = 2, 3); \\
& 10 \leq x_i \leq 1000 \ \ (i = 4, \ldots, 8); \quad -\pi/2 \leq y_p \leq \pi/2, \forall p
\end{aligned}
$$

TP10 (Existing problem with additional complexities [72]):

$$
\begin{aligned}
\min_{x} \quad & F(x) = 37.293239x_1 + 0.8356891x_1x_5 + 5.3578547x_3^2 - 40792.14 \\
& \qquad + \sum_{p=1}^{P}(y_p - x_1 - x_3)^2 - 150\sum_{q=1}^{Q}\cos(2\pi z_q) \\
\text{s.t.} \quad & 0.0056858x_2x_5 - 0.0022053x_3x_5 + 0.0006262x_1x_4 \leq 6.665593; \\
& -0.0056858x_2x_5 + 0.0022053x_3x_5 - 0.0006262x_1x_4 \leq 85.334407; \\
& 0.0071317x_2x_5 + 0.0021813x_3^2 + 0.0029955x_1x_2 \leq 29.48751; \\
& -0.0071317x_2x_5 - 0.0021813x_3^2 - 0.0029955x_1x_2 + 9.48751 \leq 0; \\
& 0.0047026x_3x_5 + 0.0019085x_3x_4 + 0.0012547x_1x_3 \leq 15.699039; \\
& -0.0047026x_3x_5 - 0.0019085x_3x_4 - 0.0012547x_1x_3 + 10.699039 \leq 0; \\
& y_p - \ln(x_1 + x_3 + 1) \leq 0, \ \forall p; \\
& z_q^3 - x_1^3 - x_3^3 - x_5^3 \leq 0, \ \forall q; \\
& 78 \leq x_1 \leq 102; \quad 33 \leq x_2 \leq 45; \quad 27 \leq x_3, x_4, x_5 \leq 45; \\
& 0 \leq y_p \leq 5, \forall p; \quad -5 \leq z_q \leq 5, \forall q
\end{aligned}
$$

For computational experiments, we consider the following three scenarios: small-scale ($|y| + |z| = 0$), medium-scale ($|y| + |z| = 20$), and large-scale ($|y| + |z| = 50$) (here, $|y| + |z|$ represents the number of scalable variables and constraints in TP). All test problems are solved in small to large scale scenarios using the Interior Point (IP), Genetic Algorithm (GA), and BOBD methods. For all TP, constraint tolerance is set to 10^{-4}. For genetic algorithm, the details on GA operators and parameters value are provided in Table 10. For GA implemented in BOBD method, all parameters are kept same as mentioned in Table 10, and an improvement based-termination criterion is used. We consider a steady state genetic algorithm [77] in both explicit GA and GA used in BOBD.

Table 10: Genetic algorithm operators and parameter values for computational experiments

GA operators	**Mechanisms**	**Parameters value**
Crossover	simulated binary crossover (SBX) [20]	0.90
Mutation	polynomial mutation [21]	0.10
Selection	tournament selection [54]	–
Population size	—	200
No. of offsprings	—	2

Each test problem (TP1-TP10) is solved 11 times using the IP, GA, and BOBD methods and the corresponding objective function values are recorded. For each TP, the best feasible objective function value (obtained or known from the literature) is recorded. Every time a method is executed, we evaluate the quality of solution in terms of absolute deviation, which is the absolute difference in the solution obtained from the method and the best known solution. These deviations are illustrated using box-plots in Figure 10 and Figure 11. The figures indicate that BOBD method consistently yields the best solutions across all 11 runs for every test problem. In contrast, IP and GA frequently converge to local optima and, in several cases, even provide infeasible solutions. The BOBD method either matches or outperforms the solutions obtained by IP and GA in all instances. In the case of BOBD, the solutions are at least the same or better compared to the results obtained from IP and GA. For certain problems in figures, BOBD performance appears to be slightly worse than IP. This is because, in such cases, both BOBD and IP have converged close to the optimum and the difference in solution quality is marginal.

We also record the average computational time for solving all instances using BOBD. The IP method typically terminates within 1–10 seconds for most of the test problems. Computational times for BOBD method are provided in Table 11. For a fair comparison, GA is allowed to run for twice the time taken by the BOBD method for each instance. Overall, the data in Figure 10, Figure 11, and Table 11 convey that IP method provides the solutions quickly but it often converges to suboptimal solutions. GA frequently struggles to find even a feasible solution, particularly in medium and large scale scenarios. BOBD method took more computational time than IP, but it consistently delivers high-quality solutions during all 11 runs, which demonstrates better accuracy and repeatability of BOBD compared to IP and GA methods.

Table 11: Average computational time for BOBD method (in seconds)

Test Problem	$\|y\| + \|z\| = 0$	$\|y\| + \|z\| = 20$	$\|y\| + \|z\| = 50$
TP1	1.90	-	-
TP2	4.00	-	-
TP3	3.34	-	-
TP4	3.86	28.62	32.49
TP5	4.34	8.77	12.33
TP6	7.72	8.26	21.00
TP7	8.33	11.26	51.21
TP8	16.16	4.83	5.79
TP9	14.83	291.26	496.70
TP10	27.23	28.67	32.42

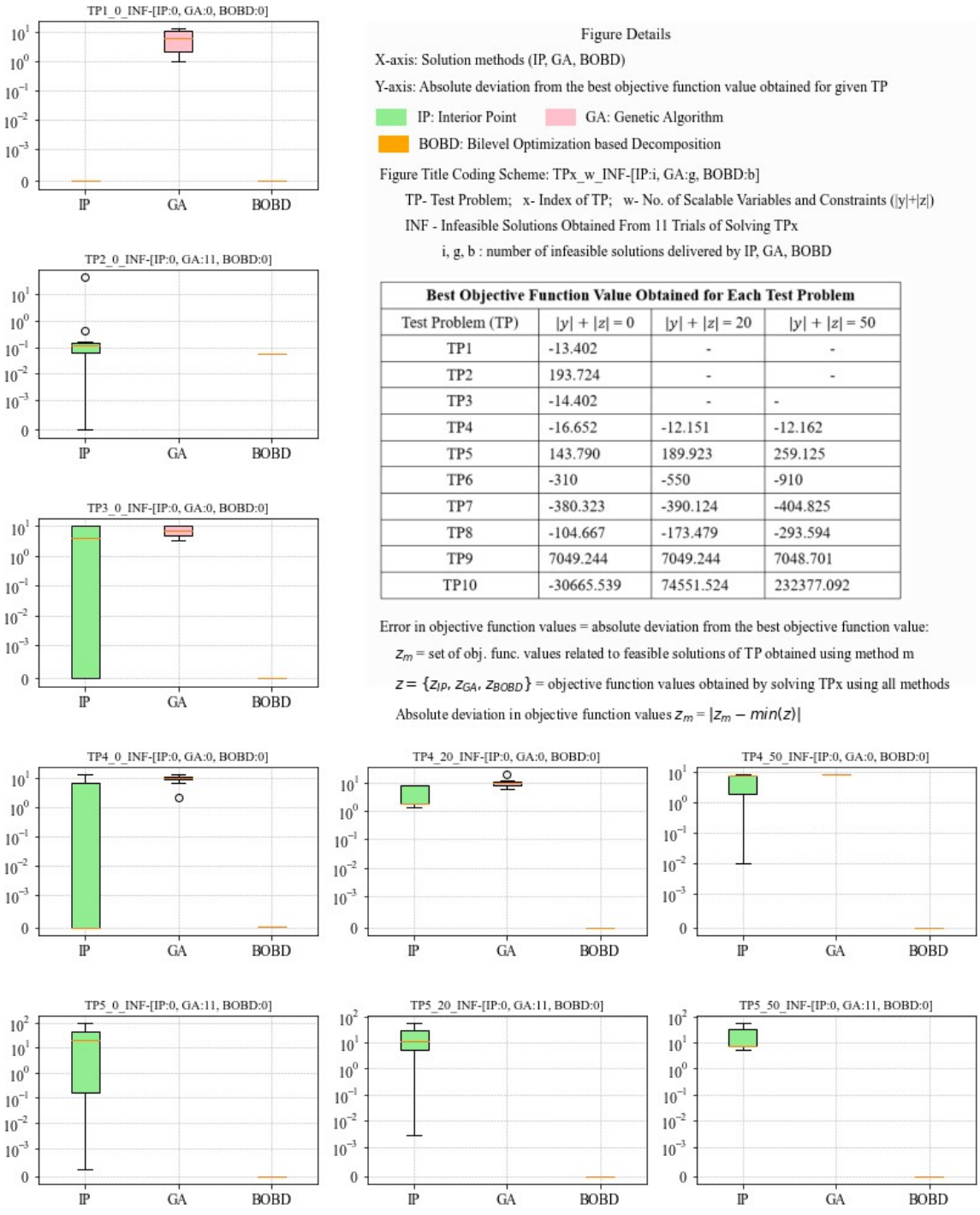


Figure Details

X-axis: Solution methods (IP, GA, BOBD)

Y-axis: Absolute deviation from the best objective function value obtained for given TP

IP: Interior Point GA: Genetic Algorithm

BOBD: Bilevel Optimization based Decomposition

Figure Title Coding Scheme: TPx_w_INF-[IP:i, GA:g, BOBD:b]

TP- Test Problem; x- Index of TP; w- No. of Scalable Variables and Constraints (|y|+|z|)

INF - Infeasible Solutions Obtained From 11 Trials of Solving TPx

i, g, b : number of infeasible solutions delivered by IP, GA, BOBD

Best Objective Function Value Obtained for Each Test Problem

Test Problem (TP)	$\|y\| + \|z\| = 0$	$\|y\| + \|z\| = 20$	$\|y\| + \|z\| = 50$
TP1	-13.402	-	-
TP2	193.724	-	-
TP3	-14.402	-	-
TP4	-16.652	-12.151	-12.162
TP5	143.790	189.923	259.125
TP6	-310	-550	-910
TP7	-380.323	-390.124	-404.825
TP8	-104.667	-173.479	-293.594
TP9	7049.244	7049.244	7048.701
TP10	-30665.539	74551.524	232377.092

Error in objective function values = absolute deviation from the best objective function value:

z_m = set of obj. func. values related to feasible solutions of TP obtained using method m

$z = \{z_{IP}, z_{GA}, z_{BOBD}\}$ = objective function values obtained by solving TPx using all methods

Absolute deviation in objective function values $z_m = |z_m - min(z)|$

Figure 10: Absolute deviation in objective function values from 11 runs: TP1-TP5

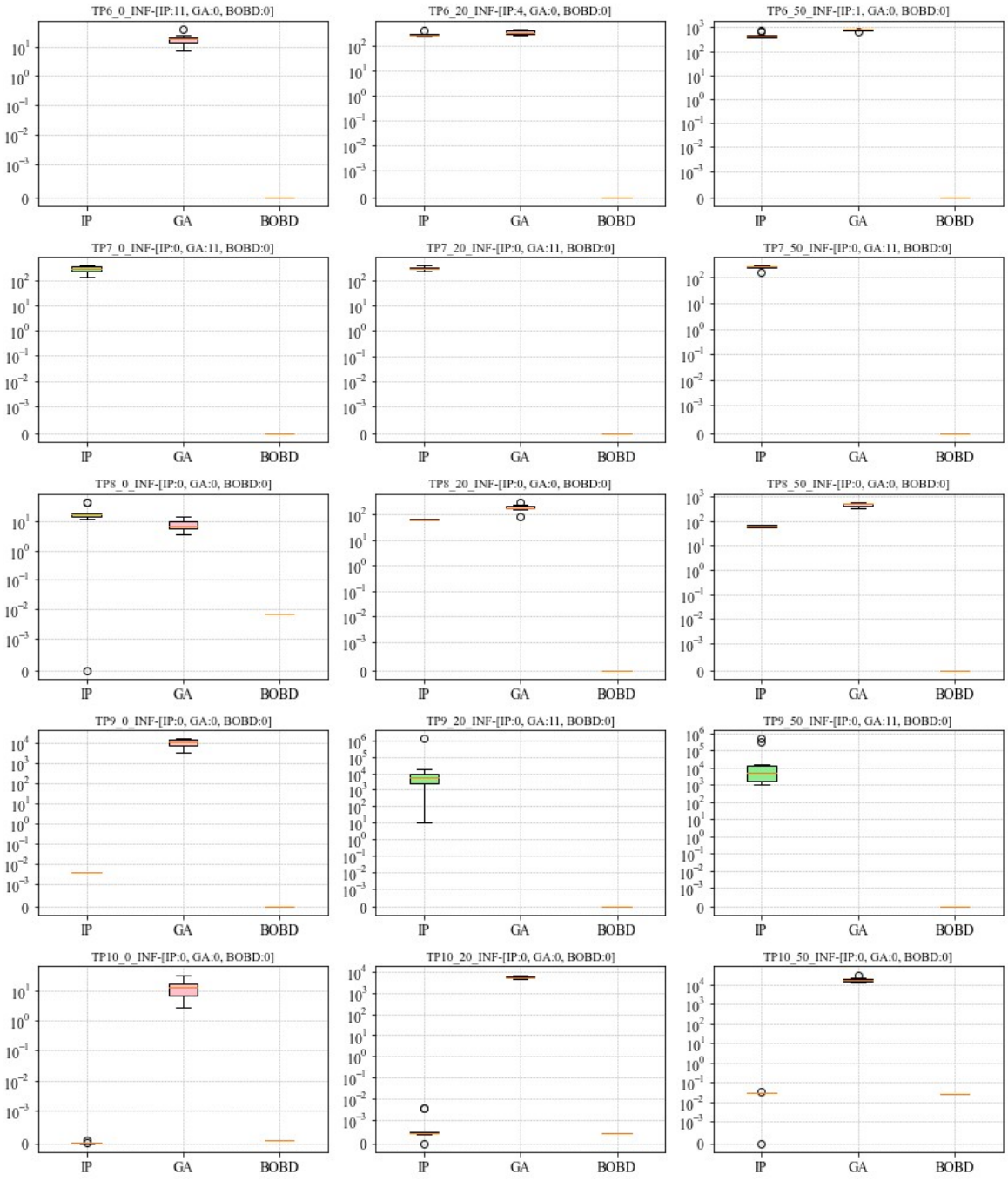


Figure 11: Absolute deviation in objective function values from 11 runs: TP6-TP10