Systems & Control Transactions

Research Article - Peer Reviewed Conference Proceeding
ESCAPE 35 - European Symposium on Computer Aided Process Engineering
Ghent, Belgium. 6-9 July 2025
Jan F.M. Van Impe, Grégoire Léonard, Satyajeet S. Bhonsale,
Monika E. Polańska, Filip Logist (Eds.)

PSE PRESS

Assessing Triviality in Random Mixed-Integer Bilevel Optimization Problems to Improve Problem Generators and Libraries

Meng-Lin Tsai[a], and Styliani Avraamidou[a*]

a Department of Chemical & Biological Engineering, University of Wisconsin-Madison, Madison WI, 53706, USA
* Corresponding Author: avraamidou@wisc.edu.

ABSTRACT

While bilevel optimization is gaining prominence across various domains, the field lacks standardized tools for generating test problems that can effectively evaluate and guide the development of efficient solution algorithms. We define the term "trivial bilevel optimization problem" as a bilevel problem whose high-point relaxation solution is also feasible. These easy-to-solve problems frequently arise in naïve implementations of random bilevel optimization problem generators, significantly impacting the evaluation of bilevel solution algorithms. However, this problem has not been addressed in the literature to our best knowledge. This work introduces a non-trivial mixed-integer bilevel optimization problem generator, NT-BMIPGen, and a problem library, designed to eliminate the generation of trivial bilevel problems. Through analysis of the bilevel problem structure, we identify key factors contributing to problem triviality. Particularly, the upper to lower variable ratios and the number of decoupled lower-level continuous variables increasing two-level competition significantly impact problem triviality. The proposed generator allows users to customize problem structures while ensuring non-triviality by checking the bilevel feasibility of highpoint relaxation. Additionally, we present a library of 105 non-trivial problems to support algorithm development and testing. This work advances the understanding of bilevel problem complexity and provides essential tools for the systematic evaluation of bilevel optimization algorithms.

Keywords: Bilevel Optimization, Random Problem Generator, Algorithm Evaluation

INTRODUCTION

Bilevel optimization, characterized by nested optimization problems, has gained prominence in modeling two-player interactions across various domains, including environmental policy [1], electricity markets [2], hierarchical optimal control [3], supply chain optimization [4], and industrial planning and scheduling [5, 6]. Despite its potential wide applicability, bilevel optimization is not widely utilized as its solution is very challenging, known to be NP-hard [7]. Mixed-integer bilevel optimization problems are even more challenging to solve [8], prompting the development of diverse solution methods, such as Bender's decomposition [9], multiparametric optimization [10, 11], penalty functions [12], and branch-and-bound/cut algorithms [13].

Metaheuristic methods, while not guaranteeing optimality or feasibility in finite time, have gained significant attention for addressing non-linear bilevel problems beyond the reach of deterministic approaches [14]. These problems are often tackled iteratively by alternating between the upper and lower levels, using techniques such as genetic algorithms [15], particle swarm optimization [16], etc. Recently, hybrid approaches have emerged, combining metaheuristics in upper-level with deterministic optimization solvers to handle the lower-level problem within this iterative framework and have guarantees of feasibility [1].

Existing bilevel problem generators and libraries provide valuable resources for testing and evaluating algorithms. For instance, problem generators tailored to specific structures have been developed [17]. A bilevel parametric optimization toolbox (B-POP) is developed with a random problem generator and problem library

[18]. BOBILib [19] offers a comprehensive mixed-integer bilevel problem library containing 2,500 problems. However, due to the diversity in problem types—encompassing different variable types, constraints, and objective functions—the field lacks standardized problem generators or libraries that are adaptable to user-defined problem structures and sizes. While random problem generators are widely used for algorithm evaluation [11], they often result to a significant number of very short solving time bilevel problems [19], limiting their utility for rigorous algorithm testing.

This article presents a non-trivial mixed-integer bilevel optimization problem generator (NT-BMIPGen) and accompanying problem library designed to support the understanding, analysis, and evaluation of bilevel optimization problems. We first outline the problem structure (number of upper/lower variables, binary/continuous, coupled/decoupled variables, and constraints). Next, we examine the concept of problem triviality and its underlying causes. We then present the bilevel problem generator and a bilevel problem library containing 105 problems, highlighting their utility in evaluating diverse non-trivial bilevel problem structures. Finally, we conclude with a discussion of future research opportunities in bilevel optimization.

GENERAL BILEVEL PROBLEM STRUCTURE

A general Mixed-Integer Bilevel programming problem (B-MIP) can be written in the form of Equation (1). The problem contains two objective functions, an upper-level objective function F and a lower-level objective function f. The problem includes both continuous and integer variables, where $\boldsymbol{x}$ is a vector of all continuous variables, $\boldsymbol{y}$ is a vector of all integer variables, subscripts u and l indicate upper- and lower-level variables respectively, and function vectors $\boldsymbol{G}$ and $\boldsymbol{g}$ represent upper- and lower-level constraints respectively.

$$\min_{\boldsymbol{x},\boldsymbol{y}} F(\boldsymbol{x},\boldsymbol{y})$$

$$s.t.\, G(\boldsymbol{x},\boldsymbol{y}) \le 0,$$

$$\boldsymbol{x_l},\boldsymbol{y_l} \in arg\min_{\boldsymbol{x_l},\boldsymbol{y_l}}\{f(\boldsymbol{x},\boldsymbol{y}): g(\boldsymbol{x},\boldsymbol{y}) \le 0\}$$

$$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x_u}\\ \boldsymbol{x_l}\end{bmatrix}^T, \boldsymbol{y} = \begin{bmatrix}\boldsymbol{y_u}\\ \boldsymbol{y_l}\end{bmatrix}^T,$$

$$\boldsymbol{x} \in R^n, \boldsymbol{y} \in Z^m \qquad (1)$$

Bilevel programming problems couples two distinct optimization problems by embedding the lower-level optimization problem as a constraint of the upper-level optimization problem. A bilevel problem is feasible if both upper- and lower-level constraints (G, g) are satisfied, and the lower-level problem is solved to global optimality at the selected optimal upper-level variables.

Coupling bilevel optimization structure

Couplings between the two optimization levels appear linked when upper-level and lower-level variables emerge together in either the constraints and/or objective functions of the problem. Coupled variables, defined as variables that appear in constraint or objective functions that doesn't belongs to their own level, enhance the interaction between the two optimization levels. Coupled variables can manifest as either collaborative or competitive dynamics. In contrast, decoupled variables lack direct interaction with the other level and only exist in their own level but can still increase the overall problem complexity by increasing the complexity of the problem within each level.

To explore the coupling behavior of the bilevel optimization problems, general B-MIPs can be reformulated, without loss of generality, as Equation (2). This form is modified from [16] and can help introduce different B-MIP structures to explore coupling behavior.

$$\min_{\boldsymbol{x},\boldsymbol{y}} F = F_u(\boldsymbol{x_{ud}},\boldsymbol{y_{ud}}) + F_l(\boldsymbol{x_{ld}},\boldsymbol{y_{ld}}) + F_c(\boldsymbol{x_{uc}},\boldsymbol{x_{lc}},\boldsymbol{y_{uc}},\boldsymbol{y_{lc}})$$

$$s.t.\, G_{ud}(\boldsymbol{x_{ud}},\boldsymbol{y_{ud}}) \le 0, G_{uc}(\boldsymbol{x_{ud}},\boldsymbol{x_{uc}},\boldsymbol{y_{ud}},\boldsymbol{y_{uc}}) \le 0$$

$$\boldsymbol{x_{ld}},\boldsymbol{x_{lc}},\boldsymbol{y_{ld}},\boldsymbol{y_{lc}} \in arg\min_{x_{ld},x_{lc},y_{ld},y_{lc}}\{f(\boldsymbol{x},\boldsymbol{y}) = f_l(\boldsymbol{x_{ld}},\boldsymbol{y_{ld}}) + f_c(\boldsymbol{x_{uc}},\boldsymbol{x_{lc}},\boldsymbol{y_{uc}},\boldsymbol{y_{lc}}): g_{ld}(\boldsymbol{x_{ld}},\boldsymbol{y_{ld}}) \le 0, g_{lc}(\boldsymbol{x_l},\boldsymbol{y_l}) \le 0, g_g(\boldsymbol{x_{uc}},\boldsymbol{x_{lc}},\boldsymbol{y_{uc}},\boldsymbol{y_{lc}}) \le 0\}$$

$$\boldsymbol{x} = \begin{bmatrix}\boldsymbol{x_{ud}}\\ \boldsymbol{x_{uc}}\\ \boldsymbol{x_{ld}}\\ \boldsymbol{x_{lc}}\end{bmatrix}^T, y = \begin{bmatrix}\boldsymbol{y_{ud}}\\ \boldsymbol{y_{uc}}\\ \boldsymbol{y_{ld}}\\ \boldsymbol{y_{lc}}\end{bmatrix}^T,$$

$$\boldsymbol{x} \in R^n, \boldsymbol{y} \in Z^m \qquad (2)$$

,where $\boldsymbol{x}$ is a vector of all continuous variables, $\boldsymbol{y}$ is a vector of all integer variables, subscripts u and l indicate upper and lower levels respectively; and subscripts c and d indicates coupled and decoupled respectively as shown in Table 1

Table 1: Variable Classification by Type and Coupling.

	Continuous ($\boldsymbol{x}$)				Binary ($\boldsymbol{y}$)			
	$\boldsymbol{x_{ud}}$	$\boldsymbol{x_{uc}}$	$\boldsymbol{x_{ld}}$	$\boldsymbol{x_{lc}}$	$\boldsymbol{y_{ud}}$	$\boldsymbol{y_{uc}}$	$\boldsymbol{y_{ld}}$	$\boldsymbol{y_{lc}}$
Upper	✓	✓			✓	✓		
Lower			✓	✓			✓	✓
Coupled		✓		✓		✓		✓
Decoupled	✓		✓		✓		✓	

The constraints G and g correspond to the upper and lower constraints respectively, which can be categorized into 5 types based on the type of variables that can appear within them (Table 2). Functions F and f are the

upper and lower-level objective functions, which can also be decomposed into five subtypes based on their structure. A detailed classification of the five types of constraints and objective functions is provided in Table 2.

Table 2: Constraints and Objective Function Classification by Type and Coupling.

	Name	Description
Constraints		
1	$G_{ud}(\boldsymbol{x_{ud}}, \boldsymbol{y_{ud}})$	Upper decoupled
2	$G_{uc}(\boldsymbol{x_{ud}}, \boldsymbol{x_{uc}}, \boldsymbol{y_{ud}}, \boldsymbol{y_{uc}})$	Upper coupled
3	$g_{ld}(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}})$	Lower decoupled
4	$g_{lc}(\boldsymbol{x_{ld}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{ld}}, \boldsymbol{y_{lc}})$	Lower coupled
5	$g_g(\boldsymbol{x_{uc}}, \boldsymbol{y_{uc}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{lc}})$	General
Objective Functions		
1	$F_u(\boldsymbol{x_{ud}}, \boldsymbol{y_{ud}})$	Upper decoupled upper-level
2	$F_l(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}})$	Lower decoupled upper-level
3	$F_c(\boldsymbol{x_{uc}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{ud}}, \boldsymbol{y_{lc}})$	Coupled upper-level
$F = F_u + F_l + F_c$		Full upper-level
4	$f_l(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}})$	Lower decoupled lower-level
5	$f_c(\boldsymbol{x_{uc}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{ud}}, \boldsymbol{y_{lc}})$	Coupled lower-level
$f = f_l + f_c$		Full lower-level

BILEVEL PROBLEM TRIVIALITY

Even though bilevel optimization problems have been proved to be hard to solve, randomly generated bilevel optimization problems are not always hard to solve. As shown in [19], randomly generated problems of the same size and structure have a large range of solving time, some of them can be solved within seconds while some need hours to be solved. Some of the quickly and easily solvable problems are likely trivial and should not be used for algorithm development or evaluation, as they fail to represent the true complexity of bilevel optimization problems. To determine triviality, we solve two single-level problems: the high point relaxation problem (Equation (3)) and the bilevel feasibility check problem (Equation (4)).

Triviality Check process

High point relaxation problem is a single-level optimization problem optimizing the upper-level objective function over the set of all constraints in the bilevel problem, including both the upper and lower-level constraints, as shown in Equation (3). A bilevel problem is considered trivial if its high-point relaxation is bilevel feasible. To verify bilevel feasibility, we solve the bilevel feasibility check problem, where the upper-level variables' solution is fixed (denoted as $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}}$) and the lower-level problem is solved, as shown in Equation (4). If the upper-level objective F calculated by solutions in Equation (4) equals that in Equation (3), the problem is bilevel feasible.

A bilevel optimization problem is termed trivial if its solution can be obtained by solving these two single-level problems: the high-point relaxation (3), and the bilevel feasibility check problem (4). Both single-level problems are solvable using standard optimization solvers. The high-point relaxation provides the lower bound, and the bilevel feasibility check provides the upper bound to the original bilevel problem. When these bounds are equal, the solution corresponds to the original bilevel problem, and the trivial bilevel problem is solved.

$$\min_{\boldsymbol{x},\boldsymbol{y}} F = F_u(\boldsymbol{x_{ud}}, \boldsymbol{y_{ud}}) + F_l(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}}) + F_c(\boldsymbol{x_{uc}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{uc}}, \boldsymbol{y_{lc}})$$

$$s.t.\, G_{ud}(\boldsymbol{x_{ud}}, \boldsymbol{y_{ud}}) \le 0, G_{uc}(\boldsymbol{x_{ud}}, \boldsymbol{x_{uc}}, \boldsymbol{y_{ud}}, \boldsymbol{y_{uc}}) \le 0,$$
$$g_{ld}(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}}) \le 0, g_{lc}(\boldsymbol{x_l}, \boldsymbol{y_l}) \le 0,$$
$$g_g(\boldsymbol{x_{uc}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{uc}}, \boldsymbol{y_{lc}}) \le 0\}$$

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x_{ud}} \\ \boldsymbol{x_{uc}} \\ \boldsymbol{x_{ld}} \\ \boldsymbol{x_{lc}} \end{bmatrix}^T, \boldsymbol{y} = \begin{bmatrix} \boldsymbol{y_{ud}} \\ \boldsymbol{y_{uc}} \\ \boldsymbol{y_{ld}} \\ \boldsymbol{y_{lc}} \end{bmatrix}^T,$$

$$\boldsymbol{x} \in R^n, \boldsymbol{y} \in Z^m \tag{3}$$

$$\min_{\boldsymbol{x_{ld}}, \boldsymbol{x_{lc}}, \boldsymbol{y_{ld}}, \boldsymbol{y_{lc}}} f(\boldsymbol{x}, \boldsymbol{y}) = f_l(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}}) + f_c(\tilde{\boldsymbol{x}}_{\boldsymbol{uc}}, \boldsymbol{x_{lc}}, \tilde{\boldsymbol{y}}_{\boldsymbol{uc}}, \boldsymbol{y_{lc}}):$$
$$st.\, G_{uc}(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}}) \le 0, g_{lc}(\boldsymbol{x_l}, \boldsymbol{y_l}) \le 0, g_g(\tilde{\boldsymbol{x}}_{\boldsymbol{uc}}, \boldsymbol{x_{lc}}, \tilde{\boldsymbol{y}}_{\boldsymbol{uc}}, \boldsymbol{y_{lc}}) \le 0,$$
$$\boldsymbol{x} = \begin{bmatrix} \tilde{\boldsymbol{x}}_{\boldsymbol{ud}} \\ \tilde{\boldsymbol{x}}_{\boldsymbol{uc}} \\ \boldsymbol{x_{ld}} \\ \boldsymbol{x_{lc}} \end{bmatrix}^T, \boldsymbol{y} = \begin{bmatrix} \tilde{\boldsymbol{y}}_{\boldsymbol{ud}} \\ \tilde{\boldsymbol{y}}_{\boldsymbol{uc}} \\ \boldsymbol{y_{ld}} \\ \boldsymbol{y_{lc}} \end{bmatrix}^T,$$
$$F = F_u(\tilde{\boldsymbol{x}}_{\boldsymbol{ud}}, \boldsymbol{y_{ud}}) + F_l(\boldsymbol{x_{ld}}, \boldsymbol{y_{ld}}) + F_c(\tilde{\boldsymbol{x}}_{\boldsymbol{uc}}, \boldsymbol{x_{lc}}, \tilde{\boldsymbol{y}}_{\boldsymbol{uc}}, \boldsymbol{y_{lc}})$$

$$\boldsymbol{x} \in R^n, \boldsymbol{y} \in Z^m \tag{4}$$

Exploring Factors Contributing to Triviality

Bilevel optimization problems with different structures have different implication in the real world. Looking into what is causing the bilevel optimization problem triviality can help us identify what is the critical factors that create complex interactions of the problem. We analyzed various problem structures to identify factors contributing to solution triviality. The problems are generated by using naïve random bilevel problem generation process, which we generate the objectives and constraints using random vectors, as detailed in the problem generator and library section.

Continuous bilevel optimization problem

In Figure 3, we varied the number of continuous coupled variables $\boldsymbol{x_{uc}}$ and $\boldsymbol{x_{lc}}$ within the range of 2 to 20

while fixing the general constraints $g_g = 10$. The resulting heatmap illustrates the trivial percentage calculated across 30 test instances, where the trivial percentage is defined as the ratio of number of trivial problems to the total number of tests.

Defining the upper-to-lower variable ratio (ULVR) as the number of upper-level variables divided by the number of lower-level variables, we observed the following trends: when $ULVR > 2$, the trivial percentage is high (above 80%); when $2 > ULVR > 0.5$, the trivial percentage is moderate (20–80%); and when $ULVR < 0.5$, the trivial percentage is low (below 20%).

These trends indicate that the lack of lower-level variables in the problem prevents the lower-level objective from competing with the upper-level objective, resulting in triviality. This finding suggests that increasing lower-level competition with the upper-level can help reduce triviality. To test this hypothesis, we modified the lower-level decoupled continuous variable x_{ld}, as introducing such variables complicates the lower-level objective but has no direct impact on the upper-level objective. This creates divergences between the two objectives, causing them to compete, resulting in triviality decreases.

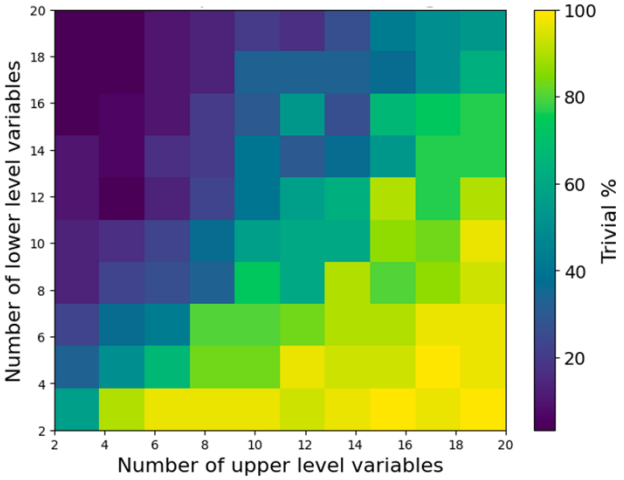

Figure 3. Trivial percentage heatmap by varying $\boldsymbol{x_{uc}}$ (2-20) and $\boldsymbol{x_{lc}}$ (2-20) of the continuous bi-level optimization problems. Each structure is tested for 30 times, and the $g_g = 10$ for all instances.

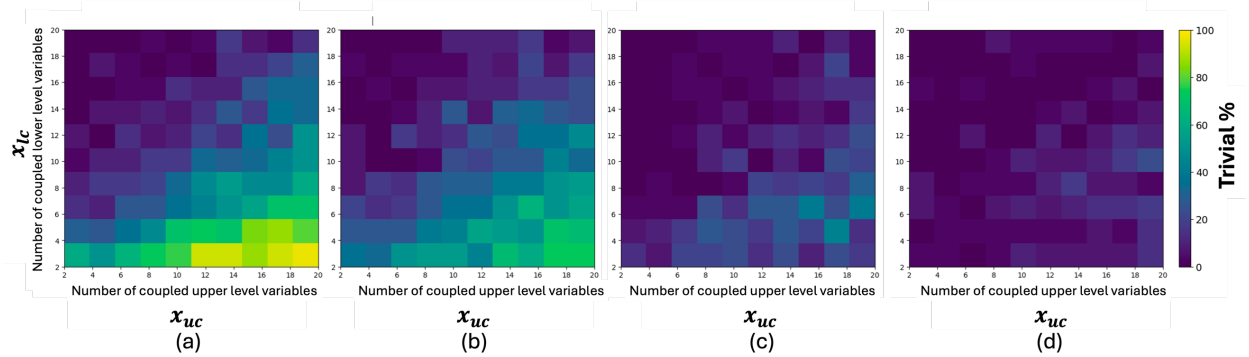

Figure 1: Changing $\boldsymbol{x_{uc}}$, $\boldsymbol{x_{lc}}$ from 2 to 20, fixing Number of decoupled lower level continuous variable $\boldsymbol{x_{ld}}$ at a: 0, b: 1, c: 5, d: 10; $G_{uc} = g_{lc} = g_g = 10$. Each structure is tested for 30 times. As shown by the graph, by changing $\boldsymbol{x_{ld}}$, the triviality decreases. Suggesting Number of decoupled lower level continuous variable is a key factor in triviality.

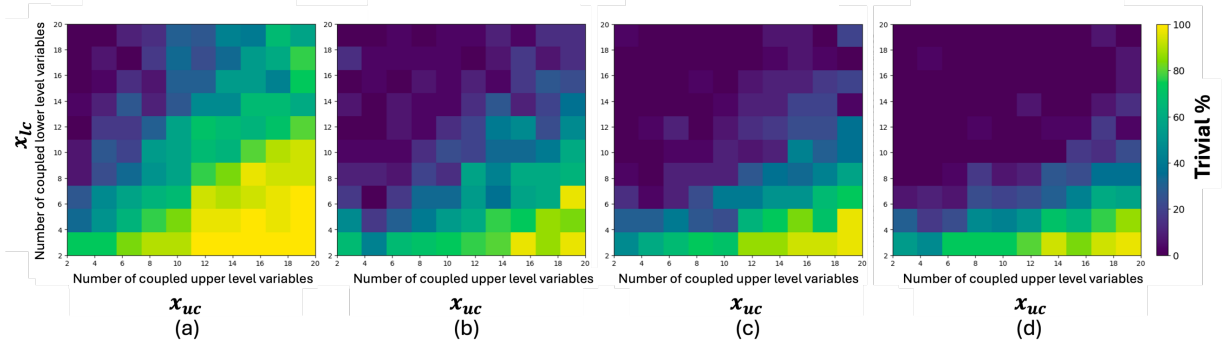

Figure 2: Changing $\boldsymbol{x_{uc}}$, $\boldsymbol{x_{lc}}$ from 2 to 20, fixing Number of binary coupled upper- and binary coupled lower-level variable ($\boldsymbol{y_{uc}}$, $\boldsymbol{y_{lc}}$) at a: (0,0), b: (2,2), c: (10,10), d: (20,20), g_g is fixed at 20 for each type of constraints. Each structure is tested for 30 times. As shown by the graph, by changing y_{uc}, y_{lc}, the triviality decreases. Suggesting

Continuous bilevel optimization problem with decoupled structure

Figure 2 illustrates the effect of varying the number of pure lower-level decoupled continuous variables ($\boldsymbol{x_{ld}}$) while fixing the values of the coupled variables $\boldsymbol{x_{uc}}$ and $\boldsymbol{x_{lc}}$, and setting the constraints $G_{uc} = g_{lc} = g_g = 10$. Subfigures 2 (a), (b), (c), and (d) represent cases where $\boldsymbol{x_{ld}}$ is set to 0, 1, 5, and 10, respectively. The resulting heatmaps illustrate the trivial percentage across 30 test instances The heatmaps clearly demonstrate that increasing the number of $\boldsymbol{x_{ld}}$ reduces the triviality of the problem. With the addition of just 1 $\boldsymbol{x_{ld}}$, the heatmap results show clear reduction of triviality percentage, as shown in Figure 2 (b). When x_{ld} continuous increases, the percentage of trivial solutions decreases significantly. The triviality percentage goes to near zero in most cases as x_{ld} goes to 10, as shown in Figure 2 (d).

This observation highlights that the number of decoupled lower-level continuous variables plays a crucial role in determining triviality. The results suggest that introducing or increasing $\boldsymbol{x_{ld}}$ enhances the complexity of interactions between levels, reducing the likelihood of trivial solutions.

Mixed-Integer bilevel optimization problem

Introducing more binary variables also fosters competition between the upper and lower levels. As shown in Figure 3, increasing the number of coupled binary variables reduces triviality; however, the effect is less pronounced compared to the influence of x_{ld} as shown in Figure 2. This effect is less pronounced in the region where the number of continuous coupled lower-level variables x_{lc} is small. We suspect that by only increasing the number of the binary coupled lower-level variables cannot effectively affect the lower-level objective function competence with the upper-level objective function. A minimum amount of the $\boldsymbol{x_{lc}}$ is required to create competence synergetic with the $\boldsymbol{y_{uc}}$, and $\boldsymbol{y_{lc}}$.

Decoupled binary variables also exhibit a similar trend as coupled binary variables, though these results are omitted here for brevity.

In summary, altering the number of variables across different classes significantly impacts the structure of bilevel problems.

PROBLEM GENERATOR AND LIBRARY

Based on the results in the previous section, we found that naïve implementations of a random bilevel problem generator produce varying levels of triviality depending on the problem's structure. If a problem set with high triviality is used for algorithm testing, it may lead to an overestimation of the solver's capability. To the best of our knowledge, no prior literature has addressed the issue of triviality in generating bilevel optimization problems. Therefore, we developed the random bilevel problem generator, NT-BMIPGen, which can generate non-trivial mixed integer bilevel test problems. The generator cand help the field to generate problems that are non-trivial and truly reflect the ability of the tested algorithm. The generation process is shown in Figure 4.

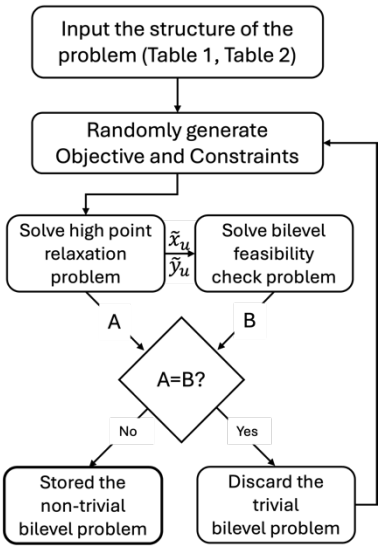

Figure 4. Flow diagram showing how random non-trivial bilevel optimization problem generator works.

The user can customize problem structures to explore different forms of two-level interaction by specifying the parameters listed in Table 1 and Table 2. To determine the parameters for the constraints and the objective function, the generator generates a matrix and a vector for each constraint type, along with random vectors for the upper- and lower-level objective subfunctions, based on the number of corresponding variables and constraints number specified.

By default, random parameter values are sampled from a uniform distribution in the range [-10, 10], except for the constraint right hand side vectors, which are restricted to non-negative values in the range [0, 10] to ensure the origin is always feasible. This design ensures the generated problems' feasibility. The continuous variables have a default upper and lower bound of -10 and 10, respectively.

Each problem undergoes a triviality check. The single level subproblems (high point relaxation problem (3) and the bilevel feasibility check problem (4)) are formulated in Pyomo [20] and solved using the Gurobi solver (v11.0.0) [21]. Only non-trivial problems, as verified through this process, are retained and stored with their upper and lower bounds obtained by solving the two subproblems. While the trivial problem is discarded, and the generator will generate another random problem. Each generated problem is stored in a folder, where each matrix and vector are saved in separate CSV files, and

associated parameters and upper and lower bounds solved by the two subproblems are stored in a JSON file. The non-trivial mixed-integer bilevel generator is available at https://test.pypi.org/project/NT-BMIPGen/.

A library with 105 different structure non-trivial B-MIP problems generated from the generator is stored at https://drive.google.com/file/d/185xnM-IG_vziXSDqGIyhSU-HDFgRDr51/view.

CONCLUSION AND DISCUSSION

This work emphasizes the importance of problem triviality in designing problem generators and using a problem library for bilevel optimization problems, which can lead to the meaningless comparison and evaluation of different solvers, an issue that has been largely overlooked in previous research.

It is worth noting that while the effects of various constraints were also examined, the interaction is more complicated, and no clear trends were observed. A more detailed analysis represents a compelling topic for future research. We acknowledge that our study examines only a limited subset of bilevel problem structures. Future studies will delve into a broader range of bilevel problem structures to gain a deeper understanding of how various parameters influence triviality and its effect on algorithm benchmarking and development.

The proposed bilevel problem generator and library aims to enhance the robustness of continuous and mixed-integer bilevel optimization algorithm testing by ensuring that the generated problems provide a meaningful challenge to the algorithms, and accelerate the development of efficient solvers for complex, real-world bilevel optimization problems.

ACKNOWLEDGEMENTS

This material is based upon work supported by the Wisconsin Alumni Research Foundation and the Department of Chemical and Biological Engineering at the University of Wisconsin-Madison. Furthermore, we would like to acknowledge the assistance provided by large language models (LLMs), including OpenAI's ChatGPT-4o and Anthropic's Claude 3.5, in the preparation of this manuscript. These tools were used for improving the grammar and fluency of the text, as well as for assistance in coding and commenting.

REFERENCES

1. Beykal B, Avraamidou S, Pistikopoulos IP, Onel M, Pistikopoulos EN. Domino: Data-driven optimization of bi-level mixed-integer nonlinear problems. J Glob Optim 78:1–36 (2020). https://doi.org/10.1007/s10898-020-00890-3
2. Fampa M, Barroso L, Candal D, Simonetti L. Bilevel optimization applied to strategic pricing in competitive electricity markets. *Comput Optim Appl* 39:121–142 (2008). https://doi.org/10.1007/s10589-007-9066-4
3. Avraamidou S, Pistikopoulos EN. A multi-parametric bi-level optimization strategy for hierarchical model predictive control. *Comput Aided Chem Eng* 40:1591–1596 (2017a). https://doi.org/10.1016/B978-0-444-63965-3.50267-1
4. Yue D, You F. Stackelberg-game-based modeling and optimization for supply chain design and operations: A mixed integer bilevel programming framework. Comput. Chem. Eng. 102:81-95 (2017). https://doi.org/10.1016/j.compchemeng.2016.07.026
5. Avraamidou S, Pistikopoulos EN. A multiparametric mixed-integer bi-level optimization strategy for supply chain planning under demand uncertainty. *IFAC-PapersOnLine* 50(1):10178–10183 (2017b). https://doi.org/10.1016/j.ifacol.2017.08.1766
6. Nikkhah H, Aghayev Z, Shahbazi A, Charitopoulos VM, Avraamidou S, Beykal B. Bi-level data-driven enterprise-wide optimization with mixed-integer nonlinear scheduling problems. Digit Chem Eng 100218 (2025). https://doi.org/10.1016/j.dche.2025.100218
7. Hansen P, Jaumard B, Savard G. New branch-and-bound rules for linear bilevel programming. SIAM J Sci Stat Comput 13(5):1194–1217 (1992). https://doi.org/10.1137/0913069
8. Kleinert T, Labbé M, Ljubić I, Schmidt M. A survey on mixed-integer programming techniques in bilevel optimization. *EURO J Comput* Optim 9:100007 (2021). https://doi.org/10.1016/j.ejco.2021.100007
9. Saharidis GK, Ierapetritou MG. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *J Glob O*ptim 44:29–51 (2009). https://doi.org/10.1007/s10898-008-9291-0
10. Köppe M, Queyranne M, Ryan CT. Parametric integer programming algorithm for bilevel mixed-integer programs. *J Optim Theory* Appl 146(1):137–150 (2010). https://doi.org/10.1007/s10957-010-9668-3
11. Avraamidou S, Pistikopoulos EN. A multi-parametric optimization approach for bilevel mixed-integer linear and quadratic programming problems. *Comput Chem Eng* 125:98–113 (2019). https://doi.org/10.1016/j.compchemeng.2019.01.021
12. Dempe S, Kalashnikov V, Ríos-Mercado RZ. Discrete bilevel programming: Application to a

natural gas cash-out problem. *Eur J Oper* Res 166(2):469–488 (2005). https://doi.org/10.1016/j.ejor.2004.01.047
13. Fischetti M, Ljubić I, Monaci M, Sinnl M. On the use of intersection cuts for bilevel optimization. Math Program 172(1):77–103 (2018). https://doi.org/10.1007/s10107-017-1189-5
14. Camacho-Vallejo J-F, Corpus C, Villegas JG. Metaheuristics for bilevel optimization: A comprehensive review. Comput Oper Res 161:106410 (2024). https://doi.org/10.1016/j.cor.2023.106410
15. Oduguwa V, Roy R. Bi-level optimization using genetic algorithm. Proc IEEE Int Conf Artif Intell Syst (ICAIS) 2002:322–327 (2002). https://doi.org/10.1109/ICAIS.2002.1048121
16. Li X, Tian P, Min X. A hierarchical particle swarm optimization for solving bilevel programming problems. Int Conf Artif Intell Soft Comput 2006:1169–1178. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11785231_122
17. Sinha A, Malo P, Deb K. Test problem construction for single-objective bilevel optimization. *Evol Comput* 22(3):439–477 (2014). https://doi.org/10.1162/EVCO_a_00116
18. Avraamidou S, Pistikopoulos EN. B-POP: Bi-level parametric optimization toolbox. Comput Chem Eng 122:193-202 (2019). https://doi.org/10.1016/j.compchemeng.2018.07.007
19. Thürauf J, Kleinert T, Ljubić I, Ralphs T, Schmidt M. Bobilib: Bilevel optimization (benchmark) Instance Library. Book Abstr PGMO DAYS 2024:121 (2024).
20. Hart, W. E., Laird, C. D., Watson, J. P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L., & Siirola, J. D. Pyomo-optimization Modeling in Python (Vol. 67, p. 277). Berlin: Springer. (2017)
21. Gurobi Optimization, LLC. Gurobi optimizer reference manual. (2024). https://www.gurobi.com