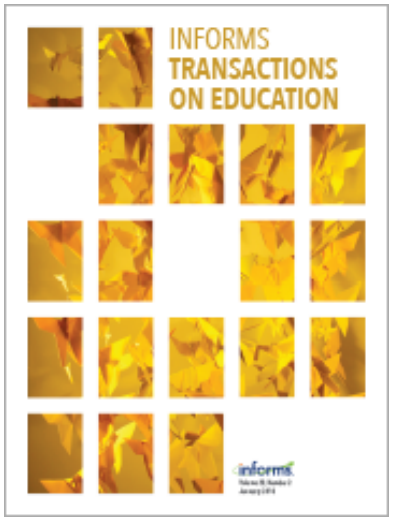


INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Teaching Applied Optimization with Large Language Models

Peter Zhang

To cite this article:
Peter Zhang (2026) Teaching Applied Optimization with Large Language Models. INFORMS Transactions on Education

Published online in Articles in Advance 12 May 2026

. https://doi.org/10.1287/ited.2024.0096

https://pubsonline.informs.org/journal/ited

INFORMS TRANSACTIONS ON EDUCATION
Articles in Advance, pp. 1–26
ISSN 1532-0545 (online)

Teaching Applied Optimization with Large Language Models

Peter Zhang[a]

[a] Carnegie Mellon University, Pittsburgh, Pennsylvania 15213
Contact: pyzhang@cmu.edu, https://orcid.org/0000-0002-0422-834X (PZ)

Received: June 6, 2024
Revised: October 10, 2024; July 31, 2025; October 15, 2025
Accepted: October 17, 2025
Published Online in Articles in Advance: May 12, 2026

https://doi.org/10.1287/ited.2024.0096

Abstract. Teachers face both push and pull to address large language models (LLMs) in the classroom. The pull comes from the increasing market demand for critically assessing or using generative AI (GAI) tools, whereas the pressure arises from students' independent adoption of GAI, regardless of stated classroom policy. A case study at Carnegie Mellon University during fall 2023 with 66 diverse students demonstrated significant shifts when LLMs were used. Notably, office hours for technical Python support dropped significantly, and the learning environment became more equitable, allowing students of varying technical backgrounds to progress at similar rates. Additionally, students showed slight improvements in problem framing but no increase in the time spent analyzing results. Surveys conducted postcourse revealed uniform feedback. Students effectively used ChatGPT for coding tasks such as debugging and learning Python syntax. ChatGPT, combined with OptiGuide, showed useful but unpredictable results in modifying existing models. The survey also highlighted time savings in assignments and projects, especially where clear instructions mimicked the prompts needed for ChatGPT. Interestingly, two distinct student profiles emerged: learners who used ChatGPT to enhance understanding and those who sought to expedite course completion, utilizing any time saved for other activities.

Funding: This project is funded in part by Carnegie Mellon University's Safety21 National University Transportation Center, which is sponsored by the U.S. Department of Transportation. It is also supported by the CMU Block Center for Technology and Society and the CMU Eberly Center for Teaching Excellence and Educational Innovation.

Keywords: optimization stack • large language models • copilot • case study

1. Introduction

1.1. MSPPM:DA Curriculum

The course Decision Analytics for Business and Policy (DABP) is offered mainly for students in the Master of Science in Public Policy and Management (MSPPM) program at Carnegie Mellon University (CMU). In particular, it is a required course for second-year students in the MSPPM: Data Analytics (DA) track.

MSPPM:DA students come from diverse backgrounds. Incoming students, on average, have a few years of work experience, but in recent years, there has been an increasing number of students coming directly out of undergraduate programs. Some come from mathematics, computer science, and engineering programs. Some others come from economics, finance, and political science programs. A portion also come from other social science and language programs. Roughly half of the students are international as of the time of writing. Some go on to take government and nonprofit positions after graduation. Some take management consulting jobs. More and more frequently, we see students interviewing and getting positions at technology companies. An exception is the 2023–2024 academic year, where tech companies' hiring freeze forced some graduating students to change mindsets. The long-term impact of generative AI (GAI) on the tech industry and labor market remains unclear.

With 162 core units and 30 elective units, the MSPPM:DA students lay the groundwork in their first semester by taking courses such as intermediate statistics, intermediate Python, database, exploratory data analysis, applied economics, and writing. In their second semester, students further improve their technical skills via courses in international policy and politics, applied econometrics, machine learning foundations, and management science. In the third and fourth semesters, they take more advanced technical courses and

synthesis courses: accounting and finance, organizational design, decision analytics (subject of this paper), capstone, policy analytics, machine learning for real-world applications, and computing.

1.2. Course Details

From conversations with graduated students, it is not unusual to see them end up in internships and full-time job positions advertised as *data* analytics but, in fact, entail substantial *decision* responsibilities. This focal course, DABP, has a dual focus on scalable decision methods and practice in cases and projects that require a data-to-decisions pipeline.

Prior to DABP, students have already been exposed to management science concepts in their first year, primarily working with Excel. DABP covers three types of data-to-decisions pipelines with practice in Python. The first type is the usual stochastic programming/robust optimization type, where one needs to summarize incomplete/imperfect data into succinct representations and design optimization models to generate decisions in Python and Gurobi. The second type is the (decoupled) predict-then-optimization procedure, where students learn to fit regression, neural network, and structural models and then solve a decision problem with these trained models as part of the objective function or constraints, again in Python, Gurobi, and other nonlinear optimization solvers. The third type is the dynamic optimization/reinforcement learning procedure, where students learn about backward inductions and approximate dynamic programming solution methods. The course emphasizes the application of these methods to large-scale problems in portfolio management, scheduling, transportation, and healthcare management.

A main practical component of the course is a project that starts right before the fall break and goes all the way to the end of the fall semester. The project is worth 40% of their total grades and is done in groups of three or four students. Students are given the opportunity to define their own problem topic, conduct data search, propose a precise decision question, formulate the decision model, solve it in Python, and analyze the results. The project includes several deliverables as checkpoints to keep students on track: initial proposal submission (10%), instructor feedback, presentation (10%), and final report (20%).

In the fall 2023 iteration of the course, projects were proposed under these topics (some include multiple projects):

1. Room assignment for church camps.
2. Personal finance planner.
3. Player acquisition for European football clubs.
4. Life cycle cost and emission optimization model for copper production.
5. Last-mile delivery as capacitated vehicle routing problems.
6. Price optimization in retail.
7. Portfolio optimization.
8. Health clinic scheduler.
9. Renewable energy portfolio optimization.
10. Real-time arbitrage of energy prices with a battery network.
11. Leveraging relative strength index to inform trading decisions.
12. Pittsburgh Regional Transit bus scheduling.
13. Ad placement decisions in mobile apps to maximize long-term. revenue.
14. Optimizing marketing strategies for banks.

1.3. Contributions

This paper contributes to the pedagogical integration of large language models (LLMs), specifically ChatGPT and OptiGuide, into a graduate-level applied optimization course. We summarize four main contributions below.

Contribution 1: Teaching strategy redesign. We restructured our teaching and assessment methods to encourage responsible AI use while mitigating plagiarism. This included critical evaluation of AI outputs in student projects and a greater emphasis on in-class, closed-book assessments.

Contribution 2: Refocusing instructional priorities. LLMs were reframed as tutoring tools rather than shortcut devices. Some teaching hours could then be reallocated away from routine technical debugging to more advanced conceptual skills such as dissecting a messy problem, model formulation, and interpreting results.

Contribution 3: Promoting fair learning. LLMs helped level the playing field across students with diverse technical backgrounds by offering accessible support in coding and syntax, thus reducing reliance on office hours and teaching assistant (TA) help. This is important to our course in particular because students come from many different quantitative and qualitative backgrounds.

Contribution 4: Empirical evidence from practice. We document survey and observational evidence showing reduced technical support demand, differentiated learning modes among students, and the strengths and limitations of LLM use in educational contexts.

2. Literature Review

Recent research on the use of large language models and generative AI in operations and business management provides interesting insights that are relevant to our applied optimization course. These studies provide anecdotal (and sometimes more formal) experiences

about the opportunities and challenges of integrating AI tools into education.

Terwiesch (2023) was among the first to propose the question: would LLMs be able to obtain a degree in higher education? The author specifically focused on an MBA operations management class at Wharton, a close counterpart of our applied optimization/operations class for graduate students in analytics at CMU. The author found that whereas AI performed well on basic tasks, it struggled with more complex problems involving variability and process flows. This finding is in line with some of our findings that AI was rated highly for more technical tasks while performing worse for more complex tasks in optimization model formulation. In our paper, we emphasize more on the consequences of such observations: the reduced technical debugging need from the teaching team, a more equitable learning environment for a diverse classroom that we have in this program, and more time for potentially engaging students in more critical-thinking tasks.

Bernabei et al. (2023) examine the use of LLMs in engineering education, with a particular focus on essay generation and assignment completion. Their findings suggest that whereas LLMs such as ChatGPT reduce the time required for routine tasks, they pose challenges regarding academic integrity and critical thinking. In the context of teaching optimization, we again share similar observations that the technical/routine tasks can be completed by AI relatively easily. This reinforces our proposal that students should use LLMs not simply as tools for completing assignments but as aids for deeper understanding of the course materials. In this current paper as well as future pedagogical designs, we hope to emphasize strategies to guide students to critically assess and refine LLM-generated outputs, ensuring that they comprehend the underlying optimization processes rather than relying solely on AI solutions.

Ibrahim et al. (2023) address another crucial aspect: the detectability of AI-generated content across various academic disciplines. Their research shows that current methods for detecting AI involvement, including tools such as GPTZero, are largely ineffective because it would overly label human-generated responses as AI generated and also miss the AI output that was edited by humans. In the context of education, this poses significant challenges for educators in coping with this unavoidable trend. In optimization courses where nuances in problem formulation can lead to different formulations and thus outcomes, it becomes even more important to emphasize transparency and critical engagement with AI outputs. Given that there is no way to prevent students from using LLMs outside of the classroom, students should be incentivized (by both assignment design and by more controlled examination time) to use LLMs not as a tool to merely complete assignments but to improve their learning process.

Svanberg et al. (2024) present an interesting work on the impact of automation on jobs, focusing on AI tools from computer vision. Although a different type of tool, their paper provides an interesting structure to think about how new tools would impact different parts of a job, that is, different tasks. In the context of classroom teaching, this brings up two lines of inquiry. First, if we consider "teaching" as the job being partially automated, then which parts of teaching would be likely automated, and how would that improve the teaching productivity (consequently, student learning outcomes) and teachers' job displacement? If we consider "learning" as the job being partially automated, then which part of learning can be automated in order to increase the productivity, that is, speeding up the learning process? Would the automation reduce the demand for teaching a particular subject to large classes because we will eventually see fewer, better students who can master a certain subject? Our current paper provides some partial answers by breaking down the job of an optimizer into model formulation and technical Python implementation.

These studies highlight a few things that educators are facing at the moment. First and foremost, it is, right now, a consensus at Heinz College and in the literature that there is no realistic way of preventing students from using AI tools outside of the classroom because detectors do not perform well, and we cannot control what students use. Consequently, we need to incentivize *how* students use these tools. Our paper provides some initial solutions that could potentially become long-term changes: asking students to critically assess AI output and increasing in-class, controlled test times. Assuming such incentives work well, then the educators in fact would have *more control* over how students should focus on the course materials. If we want students to focus on coding, then we can ask them to evaluate AI-generated code in assignments and administer more in-class tests about coding. If we want them to focus on more mathematical problems such as optimization model formulation, then we can allow them to use AI to speed up the debugging process of technical issues and focus on the mathematical formulation part. What is unclear is the impact of this kind of pedagogical changes on university-level curriculum and education as a whole. This is an important question but out of the scope of the current paper. We defer it to future studies.

3. Including Large Language Models

3.1. General Use of Large Language Models in the Course

The application of ChatGPT to operations problems has been done before (Terwiesch 2023). Compared

with prior work, we focus on a decentralized approach where 66 students take the lead in using the tool. The teaching team provided initial scaffolding and ongoing support and at times had to adjust our stance slightly to accommodate. We document this process in the section and our findings in the next few sections.

In the same fall 2023 iteration of the course, we encouraged the adoption of large language models. The decision was made after observing that students were heavily using these tools whether instructors allowed or disallowed it in other courses, and also considering the potential value of the tool in the job market. CMU course instructors had complete freedom to choose the degree to which LLMs can/cannot be used in a course, and our view does not reflect the views or contexts of other courses at Heinz College or CMU. In our case, we took and edited language from the college and included it in our syllabus.

> About generative AI tools. In this course, you are encouraged to explore the use of generative artificial intelligence (GenAI) tools, such as ChatGPT. From time to time we may explicitly ask you to explore ChatGPT and analyze its output for certain problems. In any case, the use of GenAI tools must be appropriately acknowledged and cited, including the specific version of the tool used. Submitted work should include the exact prompt used to generate the content as well as the AI's full response in an Appendix. Because AI generated content is not necessarily accurate or appropriate, it is each student's responsibility to assess the validity and applicability of any generative AI output that is submitted. You may not earn full credit if inaccurate, invalid, or inappropriate information is found in your work. Deviations from these guidelines will be considered violations of CMU's academic integrity policy. Note that expectations for "plagiarism, cheating, and acceptable assistance" on student work may vary across your courses and instructors. Please email me if you have questions regarding what is permissible and not for a particular course or assignment.

The purpose of this language is twofold. First, we wanted students to practice with LLMs. Second, we wanted to encourage honesty through transparency and relaxed rules. Throughout the course, students repeatedly verified with the teaching team whether ChatGPT was allowed and how they should include the chat history in their submissions. We had to slightly modify our stance to not require a full transcript in the appendix of the assignment submissions. A link to the transcript history was sufficient as long as it accurately reflected the interactions between the students and the machines. We recognized that there was no way to observe actions outside of the quoted transcript. For example, students could, in theory, "practice" with the LLM a few times before actually producing a "real" transcript. It was not clear how the boundaries should be drawn and what the value was in precisely monitoring students' actions. So we did not try to address this possibility and acknowledged it as a hidden action part of the setup.

3.2. Specific Use of ChatGPT and OptiGuide for Course Project

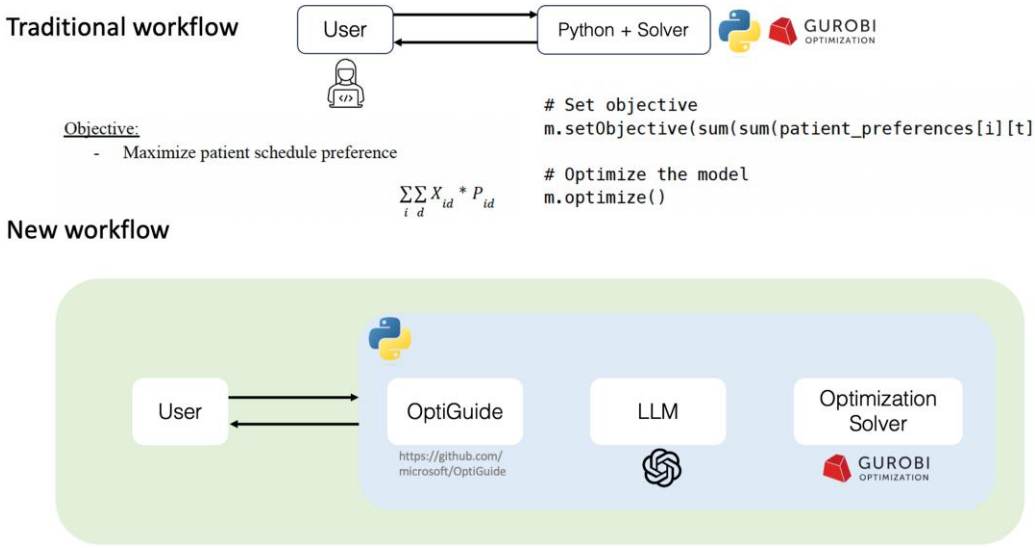
Within the scope of the course project, students are expected to integrate ChatGPT with OptiGuide (Li et al. 2023) and the Gurobi optimizer in a Python-based analytical pipeline (Figure 1). This integration is suggested for model analysis and iterative refinement (although not surprisingly, students did try to ask ChatGPT to directly come up with the formulation from scratch, as we learned from later conversations).

OptiGuide is an open-source tool that assists optimization modelers with the iterative refinement of optimization models in Python after a human has provided the initial model formulation. Without OptiGuide, LLMs such as GPT do not have precise control over the model adjustment when one needs to change the objective function, constraints, data, or decision variables. OptiGuide interfaces with ChatGPT to provide more precise control. In this course, OptiGuide is used alongside ChatGPT to help students modify parameters, constraints, and objective functions within Python-based models, enabling what-if analyses and scenario planning. This integration allows students to explore and refine optimization problems quickly, simulating real-world business environments where rapid model iteration is critical.

The project emphasizes conducting what-if analyses and sensitivity tests to explore the behavior of optimization models and solutions under varying scenarios. By using ChatGPT alongside OptiGuide, students could quickly modify the model design (parameter values, constraints, objective functions) and reinterpret solutions. This setup aims to simulate real-world situations where business analysts must adapt their strategies to rapidly changing market conditions or operational constraints, sometimes within the span of a meeting.

The motivation behind this design comes from personal experiences of doing sponsored projects and consulting with industry partners. On the one hand, for instance, weights in objective function and constraints need to be specified precisely to produce solutions for discussion with clients. On the other hand, the precise parameter values often deviate from what the clients believe should be true and sometimes generate a disproportionately negative response. Even though model designers are not attached to these values and they can be changed relatively easily in most cases, it usually takes hours to rerun the full analysis, that is, not within the same client meeting. If clients are not convinced

Figure 1. Traditional vs. New Workflows for Python and Gurobi Programming



Note. Students used the new workflow in their projects.

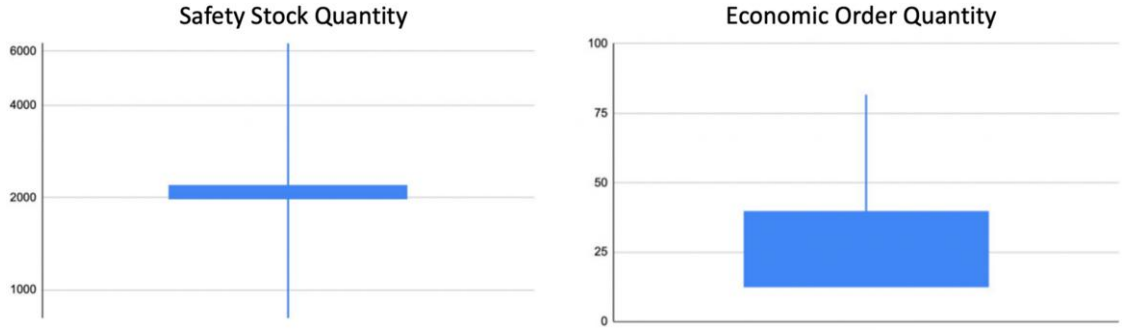
after three meetings, they usually lose confidence. We believe ChatGPT plus OptiGuide types of tools can partially alleviate this issue by automating the redesign process, where model designers can change parameter values, constraints, and objective functions on the fly and do not lose the audience.

One limitation of the tool chain is that it only applies to the first two types of decision analytics pipelines in this course: stochastic/robust optimization and predict-then-optimize. It does not apply to the dynamic optimization/reinforcement learning type of problems.

Including OptiGuide in the tool chain was an obvious choice. Without OptiGuide, ChatGPT (versions 3.5 and 4 in fall 2023) was not able to perform precise mathematical or formulation tasks (Figure 2).

When OptiGuide is integrated into the workflow, the quantitative analysis and consistency improved, especially with the well-designed prompts and well-documented Python Gurobi formulations (Tables 1 and 2). Note that OptiGuide is used to revise optimization formulations in Python, not to come up with new formulations. These numerical tests are done in another course we taught, prior to the commencement of DABP projects. A sample prompt that includes in-context learning (ICL) is listed below.

Figure 2. ChatGPT (v3.5 and 4 in Fall 2023) Alone Produces Inconsistent Results for the Same Instances of Simple Problems Such as Safety Stock and Economic Order Quantity Computations



Notes. A specific instance of safety stock and a specific instance of economic order quantity are solved multiple times (by directly asking ChatGPT to give solutions). The two box plots show the distribution of the optimal quantities for the two instances, respectively. There is only one correct answer in each instance, yet the ChatGPT-computed solutions are inconsistent over multiple runs.

Listing 1: Code snippet for in-context learning.

```
# In-context learning examples.
example_qa = """
----------
Question: What if a new room with capacity 4 is added?
Answer Code:
```python
number_of_rooms += 1
capacity_of_room = {f"room{j}" : maximum_number_of_students_per_room for j in range(1, number_of_rooms + 1)}
minimum_number_of_students_in_room = {f"room{j}" : minimum_number_of_students_per_room for j in range(1, number_of_rooms + 1)}
assignment_decision_variables = [(f"student{ind}", f"room{j}") for ind in student_data_df.index for j in range(1, number_of_rooms + 1)]
rooms = list(capacity_of_room.keys())

x = model.addVars(assignment_decision_variables,
                  vtype=GRB.BINARY,
                  name="x")

y = model.addVars(rooms,
                  vtype=GRB.BINARY,
                  name="y")

# Set objective
model.setObjective(sum(sum(overall_similarity(s1, s2) * x[s1, r1] * x[s2, r2] for s1, r1 in assignment_decision_variables if r1 == r2 and s1 < s2) for s2, r2 in assignment_decision_variables), GRB.MAXIMIZE)

# Constraints
## Each student should be assigned to one and only one room
for s in students:
```
```

```
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[0] == s) == 1, f"student_assignment_constraint_for_{s}")

## Room capacity constraints
for r in rooms:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[1] == r) <= capacity_of_room[r], f"room_capacity_constraint_for_{r}")

## Minimum number of students per room constraints
for r in rooms:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[1] == r) >= minimum_number_of_students_in_room[r], f"min_num_students_constraint_for_{r}")

## Same gender per room constraints
for r in rooms:
    model.addConstr(sum(student_gender[i[0]] * x[i] for i in assignment_decision_variables if i[1] == r) - y[r] * sum(x[i] for i in assignment_decision_variables if i[1] == r) == 0, f"single_gender_constraint_for_{r}")

```

Question: What if we require that students with ID numbers 115 and 155 are in the same room?
Answer Code:
```python
s1 = get_ind_by_id(115)
s2 = get_ind_by_id(155)

for r in rooms:
    model.addConstr(x[s1, r] - x[s2, r] == 0)

"""
```

Table 1. Model Stability: OptiGuide's Ability to Produce a Response

Test	Number of successful responses	Percentage of successful responses
Gurobi example code alone	24/50	48
With comments	40/50	80
With ICL	50/50	100

Notes. ICL refers to in-context learning, where the prompts contain example query-answer pairs to guide how the LLM should respond. The base formulation is the supply chain network design example from the Gurobi website (Gurobi 2024).

3.3. Emphasizing Critical Assessment of LLMs Halfway Through the Course

Roughly halfway through the course, students became visibly frustrated that ChatGPT and other LLMs could not produce consistent or meaningful results. From what we observed, the primary issues come from the quality of their formulations and comments and the quality of in-context learning prompts. As seen in Tables 1 and 2, the quality of Python code comments and ICL prompts significantly impacts the quality of ChatGPT with OptiGuide responses.

Whereas we could ask students to improve the quality of their code and prompts, we recognized that it was impossible to guarantee perfectly accurate responses from LLMs in general. Therefore, we emphasized a key point from the syllabus that students should not expect perfect responses from ChatGPT. In particular, they are not graded on how accurate the ChatGPT-assisted responses are, but graded on their critical assessment of the responses. And ideally, they should show both positive and negative examples of machine-assisted output. Students must report on the accuracy, reliability, and relevance of the responses provided by ChatGPT in the context of their specific projects.

Emphasizing this evaluation criterion alleviated the pressure of finding flawless technical performance, which can be unpredictable with current AI technologies. Students' stress levels dropped significantly. It encouraged students to develop a better understanding of the strengths and limitations of existing language models (Figure 3). Most importantly, it nudged students to look deeper into the formulations themselves instead of blindly relying on tools.

4. Observations by the Teaching Team

4.1. Reduction in Technical Office Hours

In the 2019–2022 versions of the same course, the instructor spent 5 to 10 hours per semester outside of the classroom to answer technical questions related to Python and Gurobi syntax, and teaching assistants spent more time. It generally took four to six weeks and two assignments for the students to feel comfortable using the tool.

In fall 2023, the introduction of LLMs in the course significantly reduced the office hours spent on troubleshooting technical issues related to Python and Gurobi syntax. The instructor received no technical Python/Gurobi questions at all. The TAs spent a few hours on it. The students were even able to utilize some Python and Gurobi functions that were previously unknown to the teaching team.

4.2. Equitable Learning Environment

Related to the previous observation, perhaps the most encouraging effect of LLMs is the leveling of the playing field.

From 2019 to 2022, the instructor repeatedly saw students who were hardworking, thoughtful, and could produce rigorous arguments, but were not familiar with the language of mathematics (notations) in general. They had a hard time catching up with the rest of the class because of the extra learning step they had to take to familiarize themselves with the language. That resulted in a missed opportunity for these students to learn.

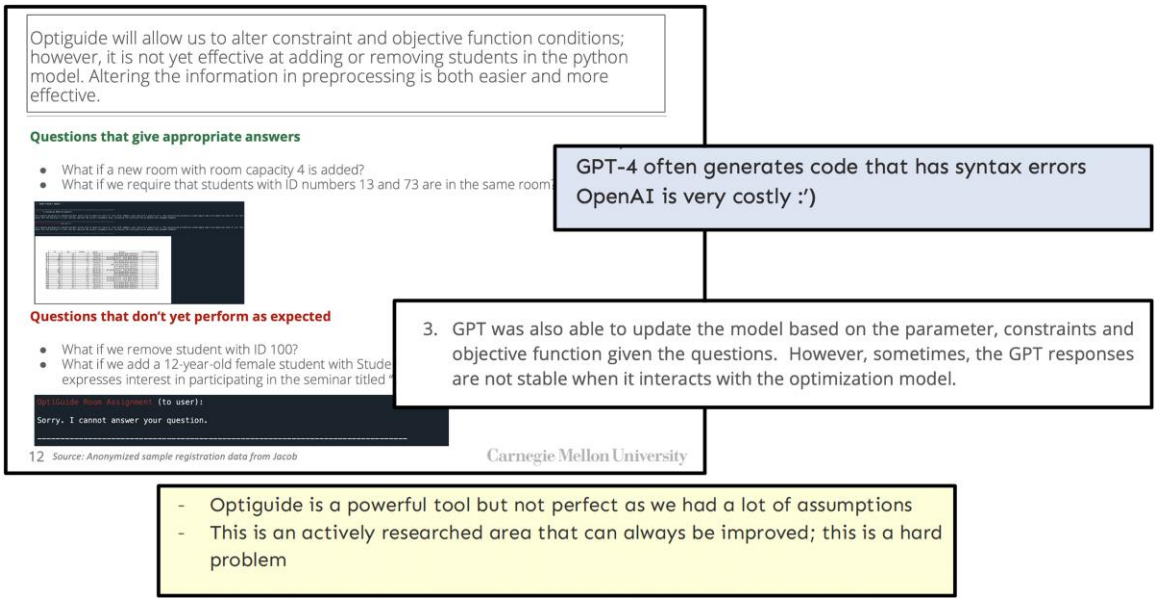
In fall 2023, we observed that students who may not have had a strong technical background but had grit and rigor were able to get over their technical questions

Table 2. Model Accuracy: OptiGuide's Ability to Produce a *Correct* Response

Test	Number of correct responses	Percentage of correct responses
Gurobi example code alone	15/24	62.5
With comments	20/40	50
With ICL	50/50	100

Notes. ICL refers to in-context learning, where the prompts contain example query-answer pairs to guide how the LLM should respond. The base formulation is the supply chain network design example from Gurobi website (Gurobi 2024).

Figure 3. Students Had Critical and Mixed Reactions About the Tool Chain by the End of the Project



quickly and dive into the actual critical-thinking part of assignments and projects.

4.3. Marginally Improved Problem Framing

By problem framing, we looked at both identifying/framing of questions in project proposals and the designing of optimization formulations.

There was a marginal improvement in how students identified project questions. This was apparent from the better diversity of project topics compared with previous years. But the effect was not significant.

There was also a marginal improvement in the mathematical formulations, especially when it came to logical constraints related to integer variables. At times, the students presented clean and clever integer formulation techniques that took the teaching team a minute to recognize as accurate. Overall, this effect was not significant.

Under this generally positive trend, we also observed that students sometimes directly presented formulation details that they were not able to fully comprehend. This is a potential risk.

4.4. Optimization Analysis Time

To our surprise, despite the advancements in tool integration and automation, it seemed that the quality of students' analysis of the optimization results did not improve. This observation indicated that whereas students were getting better at using tools to perform and manage optimization tasks, the time saved was not directed to refining their analysis. We later found this to be consistent with direct student feedback, discussed in the next section.

5. Student Feedback

5.1. Survey Design

Four months after the completion of the Decision Analytics for Business and Policy course, a reflective and anonymized survey was given to 10 students; all of them responded to all the questions. The survey was not randomized. First, the instructor reached out to the top student from fall 2023 and asked her to reach out to her classmates/friends. She was encouraged to reach out to a diverse group of people. The focal student had a concrete list of questions but conducted the survey mainly via casual conversations, asking questions about how each student felt about ChatGPT in general, for the course DABP, how they spent their time, and so on. In addition, the students surveyed were asked to rate ChatGPT's ability and their own ability on the following four types of tasks, from one to five.

1. Formulation of mathematical optimization models.
2. Coding and implementation of formulations.
3. Debugging.
4. Learning new syntax.

Nine months after the course finished, we conducted another round of survey via Google Form, sent to students' alumni email addresses (all of them graduated by May 2024). The same questions were asked. We had some additional considerations when distributing this new round of the survey.

Consideration 1. Inclusion of a survey incentive: Offering a modest incentive is a proven strategy to boost participation rates, particularly in student populations. Our goal was to reach a more representative sample of the class, and the incentive helped ensure we heard from a diverse group of students (instead of the direct hub-and-spoke outreach conducted by a focal

Table 3. Old Survey Responses: Skills in Python Coding and GenAI Capabilities

	Your skill				GenAI skill			
	Formulation	Coding	Debugging	Learn	Formulation	Coding	Debugging	Learn with GenAI
	4	4	4	3	3	5	5	5
	4	3	2	3	3	5	4	4
	4	5	5	4	2	5	3	5
	3	4	4	3	3	5	2	3
	3	4	4	3	3	4	3	5
	5	3	2	2	3	5	4	4
	5	3	2	3	4	4	3	4
	4	2	1	3	5	5	4	3
	4	3	2	2	3	4	3	4
μ	4.00	3.44	3.11	2.89	3.22	4.67	3.44	4.11
σ	0.71	0.96	1.17	0.60	0.83	0.50	0.95	0.78

student in the previous iteration of the paper). We believe a $25 gift card served as an attractive but not excessive incentive, which is commonly used in academic settings to encourage participation. The amount was sufficient to motivate students without exerting undue influence over their responses. Instead of offering the gift card to every participant, we opted for a random drawing. This method still encourages participation but reduces the potential for respondents to focus solely on the reward.

Consideration 2. Potential bias concerns and mitigation strategies: To reduce the risk of students rushing through the survey, we included one attention-check question within the survey to ensure that participants were paying attention and providing meaningful responses. We clearly communicated to students that participation was entirely voluntary and that their honest feedback was valued, whether positive or negative. This was emphasized in the email sent alongside the survey, ensuring transparency about the purpose and the use of responses.

5.2. Survey Results

Except for the first task (formulation), students' ratings of ChatGPT were higher than ratings of their own ability. We use this as a proxy to gauge how much they utilized ChatGPT for different tasks in their learning process.

Observations from the two rounds of surveys were consistent (Tables 3 and 4). For coding, students rated ChatGPT's ability between four and five on a five-point scale. Their self-assessment in coding skills varied more widely, ranging from three to five. This aligns with the teaching team's observation about the decreased need for technical office hours, suggesting that ChatGPT was a useful tool for reducing the

Table 4. New Survey Responses: Skills in Python Coding and GenAI Capabilities

	Your skill				GenAI skill			
	Formulation	Coding	Debugging	Learn	Formulation	Coding	Debugging	Learn with GenAI
	5	5	5	4	3	5	5	4
	4	4	4	3	3	3	5	4
	4	3	3	3	4	5	3	4
	4	4	4	4	3	3	4	4
	3	3	3	2	2	4	4	4
	4	4	4	3	5	5	5	4
	4	4	3	4	3	5	4	5
	2	4	4	3	3	4	4	4
	4	4	4	3	2	4	2	4
	3	3	3	3	2	4	3	4
	4	4	4	4	4	3	3	5
	3	5	4	3	3	5	4	5
	3	4	4	2	4	5	4	5
	4	3	3	3	1	3	3	4
	4	4	4	5	3	5	5	4
μ	3.67	3.87	3.73	3.27	3.00	4.20	3.87	4.27
σ	0.72	0.64	0.59	0.80	1.00	0.86	0.92	0.46

technical effort. This may be both good and bad at the same time because students can use the tool to reduce effort, but not necessarily improve learning outcomes.

For the task of generating optimization formulations, students rated ChatGPT slightly lower than themselves, indicating a recognition of the limitations of current AI capabilities in developing complex mathematical models from scratch. This aligns with qualitative feedback that students were mostly unsuccessful when asking ChatGPT to independently generate complete optimization formulations.

ChatGPT was rated highly for its ability to debug. Students found that ChatGPT performed better than themselves in identifying and fixing errors in their code. This high rating for debugging support from ChatGPT corroborates the reduced demand for office hours focused on technical issues, as ChatGPT provided an effective first line of support for troubleshooting.

Students felt that ChatGPT provided a better learning experience when it came to understanding new programming syntax, especially like the one presented by specialized packages like Gurobi. This indicates that ChatGPT was not only a tool for immediate problem solving but also *potentially* an educational resource that enhanced their learning process for some students.

5.3. Time Saving

Reflecting on the efficiency of their coursework, students acknowledged significant time savings in completing assignments and projects. This efficiency was attributed to the direct assistance from ChatGPT, particularly in well-defined technical assignments where ChatGPT's responses could be leveraged effectively. This is an interesting observation because traditionally, technical assignment questions are supposed to be given precisely to reduce ambiguity in human interpretation. This, in fact, leads to the assignment PDFs becoming well-designed prompts for machines that can be directly uploaded to ChatGPT for a full answer.

5.4. Divergent Learning Modes

A somewhat unexpected but unsurprising phenomenon is that two learning modes emerged, and students' learning paths followed a "K"-shaped pattern. In general, studious students became better, and nonstudious students learned even less. Through a combination of survey results, follow-up conversations with students, and classroom observations, we identified a strong association between students' learning modes and their underlying motivation in the course. Because this is a required course in MSPPM:DA students' curriculum, some of them have to take this course not by choice, but by program requirement. This naturally leads to two types of students: those who want to learn and those who want to just complete the course. One group viewed ChatGPT as a valuable educational tool similar to traditional resources such as StackOverflow but more efficient than traditional sources, using it to fill gaps in knowledge and deepen their understanding. The other group used ChatGPT to expedite their course requirements with minimal effort, often reallocating saved time to other priorities such as personal leisure, job search, or other more demanding courses. This distinction highlights the potential risks of AI tools in learning. It also poses an interesting question for educators. It seems that whereas the new technology tools are creating a more equitable learning environment for students with different *backgrounds*, it creates issues in terms of the effectiveness of teaching for students with different *motivations*.

6. Discussion

6.1. ChatGPT

First, ChatGPT alone does not give reliable answers for even simple calculations such as safety stock or economic order quantities. This corroborates previous observations, for example, pointed out by Terwiesch (2023).

Second, in an environment where humans, ChatGPT, and additional wrapper tools are used (Figure 1), the tool chain's performance ranged from excellent (coding and debugging) to mediocre (revising formulations that have been implemented in Python) to poor (coming up with new formulations for complex optimization problems).

Third, if we look deeper into how the tool chain performs under different setups (Tables 1 and 2), we can see that thoroughly commenting code and using in-context learning can significantly improve the stability and accuracy of ChatGPT's responses. In-context learning refers to prefixing a prompt with "sample query 1 – sample answer 1; sample query 2 – sample answer 2; ⋯ actual query –".

Fourth, focusing on in-context learning: In our experience, including as few as two examples suffices when students asked ChatGPT to revise a formulation. This is a remarkable ability. Not only are the problem formulations complex, but the problems are *novel*—created by the students in this course. This provides some arguments to support the existence of LLMs' ICL ability. We note that GPT4 was significantly better than GPT3.5, thus ICL may also be an emerging ability. ICL ability would potentially alleviate (1) the cost of context-specific fine-tuning, and (2) some privacy concerns because LLM model parameters are not necessarily updated with in-context data.

6.2. Syllabus

Language in the syllabus will be modified slightly to address the inevitable use of LLMs in assignments, take-home exams, and projects.

> About Generative AI Tools. In this course, you are encouraged to explore the use of generative artificial intelligence (GenAI) tools, such as ChatGPT. From time to time, we may explicitly ask you to explore ChatGPT and analyze its output for certain problems. In any case, the use of GenAI tools must be appropriately acknowledged and cited, including the specific version of the tool used. Submitted work should include a link to the chat history. Because AI-generated content is not necessarily accurate or appropriate, it is each student's responsibility to assess the validity and applicability of any generative AI output that is submitted. If there are mistakes in the LLM-generated responses, you can still receive grades by correctly identifying all the inaccuracies generated by machines. Deviations from these guidelines will be considered violations of CMU's academic integrity policy. Note that expectations for "plagiarism, cheating, and acceptable assistance" on student work may vary across your courses and instructors. Please email me if you have questions regarding what is permissible and not for a particular course or assignment.

6.3. Plagiarism Concerns

Plagiarism is an important aspect of incorporating foundational models that could provide a response to almost any question. At Heinz College, we (instructors of all courses) asked ourselves a question: if a hypothetical student were to use ChatGPT to answer all take-home assignments and projects, would they be able to pass the courses and obtain a degree? Traditionally, well-designed assignment questions helped to minimize ambiguity, but these same designs now serve as effective prompts for GenAI tools, which raises new concerns for academic integrity. To address this issue, we implemented two significant changes to our course design.

We required students to provide a critical assessment of the responses generated by ChatGPT, especially in the course project. This approach placed students in the role of an evaluator, assessing the accuracy of the AI's outputs. For an analytical course like ours that requires precision in logic and notations, this strategy made sure students actually understood the material (e.g., what certain integer variables and logical constraints did). Their grades were then based on their ability to critically assess the AI's output. This encouraged students to understand the material thoroughly rather than passively relying on ChatGPT's answers.

Second, we incorporated more time for closed-book, in-class tests to check students' understanding independently of GenAI tools. And importantly, we gave students advanced notice about the increased test time in the course so that they had time to properly adapt to this incentive. This allowed us to evaluate their understanding of the course materials in a controlled environment.

We observed positive results from both strategies, and we believe that these methods will remain effective as LLMs become increasingly integrated into educational contexts.

6.4. Conclusion

We make the following observations and extrapolations.

Having implemented and studied the integration of LLMs into a graduate applied optimization course, we conclude that effective educational use of AI tools hinges not on technical enforcement but on intentional pedagogical design. The pivot from prohibition to structured engagement proved fruitful across teaching effectiveness, assessment strategy, and student equity.

One key insight is that simply allowing LLMs is not sufficient. Students need structured incentives to engage with these tools critically. Without guided expectations, the risk is both misuse and missed opportunities for deeper learning. Our course design required students to assess AI-generated outputs analytically, positioning them as active judges of machine-generated work rather than passive users.

Another important observation is that whereas LLMs can democratize access to technical support, they also exacerbate motivational disparities. Students who were intrinsically motivated used AI tools to deepen understanding, whereas others used them primarily to minimize effort. This K-shaped divergence highlights a need for ongoing adaptation in course design and the instructor's role in order to address the *human* aspect of learning: learning intentions.

The empirical reduction in technical support demand allowed instructors to redirect efforts toward fostering higher-order thinking. This shift allowed us to align instructional effort with what cannot be automated. The design choices we made (such as emphasizing in-class testing and focusing grading rubrics on critical assessment) can serve as a model for courses grappling with similar technological disruptions.

Finally, our findings suggest that responsible LLM integration can enhance rigorous education. But this outcome is not automatic. It demands instructors to examine students' learning intentions, deliberately scaffold, clearly communicate, and adapt to learning feedback.

Appendix A. Sample Project Code

Listing 2: Python-Gurobi formulation for a room assignment problem.

```
import string
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from tabulate import tabulate
from gurobipy import *
import time
import matplotlib.pyplot as plt
from pandas.plotting import table

# Number of rooms
number_of_rooms = 7

# Maximum number of students per room
maximum_number_of_students_per_room = 4

# Minimum number of students per room
minimum_number_of_students_per_room = 2

# A dictionary mapping each room to its capacity
capacity_of_room = {f"room{j}" : maximum_number_of_students_per_room for j in range(1,
    number_of_rooms + 1)}

# A mapping that associates each room with the minimum required number of students it
   should accommodate.
minimum_number_of_students_in_room = {f"room{j}" : minimum_number_of_students_per_room
    for j in range(1, number_of_rooms + 1)}

# Load student data from csv file into a data frame
student_data_df = pd.read_csv("sample_data.csv")
student_data_df.index = range(1, len(student_data_df) + 1)

# Remove or add students here

# Number of students
number_of_students = len(student_data_df)

# Encode the gender variable (Female = 0 and Male = 1)
le = LabelEncoder()
coded_genders = le.fit_transform(student_data_df["Gender"])
student_data_df["Coded Gender"] = coded_genders
```

```
39
40 # A dictionary mapping each student to their encoded gender
41 student_gender = {f"student{ind}" : student_data_df.loc[ind, "Coded Gender"] for ind
       in student_data_df.index}
42
43 # A dictionary mapping each student to their age
44 student_age = {f"student{ind}" : student_data_df.loc[ind, "Estimated Age"] for ind in
       student_data_df.index}
45
46 # A dictionary mapping each student to their student ID
47 student_id = {f"student{ind}" : student_data_df.loc[ind, "Student ID"] for ind in
       student_data_df.index}
48
49 # A dictionary mapping each student to their church
50 student_church = {f"student{ind}" : student_data_df.loc[ind, "Church"] for ind in
       student_data_df.index}
51
52 # A dictionary mapping each student to their chosen seminar
53 student_seminar = {f"student{ind}" : student_data_df.loc[ind, "Chosen Seminar"] for
       ind in student_data_df.index}
54
55 # Establish connections between each student and all available rooms
56 assignment_decision_variables = [(f"student{ind}", f"room{j}") for ind in
       student_data_df.index for j in range(1, number_of_rooms + 1)]
57
58 # Names of the z decision variables
59 z_keys = [(f"student{ind}", f"student{ind1}", f"room{j}") for j in range(1,
       number_of_rooms + 1) for ind in student_data_df.index for ind1 in student_data_df.
       index if ind < ind1]
60
61 # A list of rooms
62 rooms = list(capacity_of_room.keys())
63
64 # A list of students
65 students = list(set(i[0] for i in assignment_decision_variables))
66
67 # A function to compute the age difference between 2 students
68 def compute_age_diff(s1, s2):
69     return(np.abs(student_age[s1] - student_age[s2]))
70
71 # A function to compute the church similarity between 2 students
72 def church_similarity(s1, s2):
```

```
    if student_church[s1] == student_church[s2]:
        return(1)
    else:
        return(0)

# A function to compute the seminar similarity between 2 students
def seminar_similarity(s1, s2):
    if student_seminar[s1] == student_seminar[s2]:
        return(1)
    else:
        return(0)

# Overall similarity score
def overall_similarity(s1, s2, ws = 1, wc = 1, wa = 1):
    """
    Parameters
    ----------
    ws : weight to place on seminar similarity (float/int)
    wc : Weight to place on church similarity (float/int)
    wa : Weight to place on age similarity (float/int)

    Returns
    -------
    Overall similarity score (float/int)

    """
    return(ws * seminar_similarity(s1, s2) + wc * church_similarity(s1, s2) - wa *
    compute_age_diff(s1, s2))

# A function to generate a table visualizing the room assignments for students and
    save it as an image file.
def make_table_of_student_assignment_plan():
    room_assignment_df = pd.DataFrame()
    room_numbers = [int(''.join([ch for ch in r if ch in string.digits])) for r in
    rooms]
    room_numbers.sort()
    for room_num in room_numbers:
        room_assignment_df[room_num] = [np.abs(round(x[s, f"room{room_num}"].x)) for s
     in students]

    student_assignments_df = pd.DataFrame([student_id, student_age, student_gender,
    student_church, student_seminar], index = ["ID", "Age", "Gender", "Chuch", "Seminar
    "]).T
```

```
    student_assignments_df = student_assignments_df.loc[students].reset_index(drop =
    True)
    room_numbers_df = pd.DataFrame(data = np.array([room_numbers] * len(students)))
    student_assignments_df["Room Assignment"] = np.sum((room_numbers_df.values *
    room_assignment_df.values), axis = 1)
    student_assignments_df.sort_values("Room Assignment", inplace = True)
    student_assignments_df.index = range(len(student_assignments_df))
    # Create a figure and axis
    fig, ax = plt.subplots(figsize=(16, 8))

    # Hide the axes
    ax.axis('off')

    # Plot the DataFrame as a table
    tbl = table(ax, student_assignments_df, loc='center', colWidths=[0.1, 0.1, 0.1,
    0.1, 0.3, 0.12])

    # Set the font size of the table
    tbl.auto_set_font_size(False)
    tbl.set_fontsize(10)

    # Save the figure as an image
    file_name = 'student_assignment.png'
    plt.savefig(file_name, bbox_inches='tight', pad_inches=0.05)
    current_dir = os.getcwd()
    file_path = os.path.join(current_dir, file_name)
    return(file_path)

def get_ind_by_id(sid):
    return next((key for key, val in student_id.items() if val == sid), None)

# Create a new model
model = Model(name = "Room assignment")

# OPTIGUIDE DATA CODE GOES HERE

# Create variables
x = model.addVars(assignment_decision_variables,
                  vtype=GRB.BINARY,
                  name="x")

z = model.addVars(z_keys,
```

```
                    lb = 0,
                    ub = 1,
                    name = "z")

# Set objective function to maximize
model.setObjective(sum(overall_similarity(i[0], i[1]) * z[i] for i in z_keys), GRB.
    MAXIMIZE)

# Constraints

## Each student should be assigned to one and only one room
for s in students:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[0] == s) ==
    1, f"student_assignment_constraint_for_{s}")

## Room capacity constraints
for r in rooms:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[1] == r) <=
    capacity_of_room[r], f"room_capacity_constraint_for_{r}")
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[1] == r) >=
    minimum_number_of_students_in_room[r], f"min_num_students_constraint_for_{r}")

## Constraints on the z variables
for i in z_keys:
    model.addConstr(z[i] - x[i[0], i[2]] <= 0)
    model.addConstr(z[i] - x[i[1], i[2]] <= 0)
    model.addConstr(z[i] - x[i[0], i[2]] - x[i[1], i[2]] + 1 >= 0)

for r in rooms:
    model.addConstr(sum(z[i] for i in z_keys if i[2] == r) <= capacity_of_room[r] * (
    capacity_of_room[r] - 1)/2)
    model.addConstr(sum(z[i] for i in z_keys if i[2] == r) >=
    minimum_number_of_students_in_room[r] * (minimum_number_of_students_in_room[r] - 1)
    /2)


## Gender constraints
for i in z_keys:
    if (student_gender[i[0]] != student_gender[i[1]]):
        model.addConstr(z[i] == 0)
```

```

# Students with IDs 115 and 155 must be in the same room (Insert code here)


# Optimize model
model.Params.TimeLimit = 2 * 60
model.Params.NonConvex = 2
model.optimize()
m = model

# OPTIGUIDE CONSTRAINT CODE GOES HERE

# Solve
m.update()
model.optimize()
# make_table_of_student_assignment_plan()


print(time.ctime())
if m.status == GRB.OPTIMAL:
    print(f'Optimal fit score: {m.objVal}')
    make_table_of_student_assignment_plan()
else:
    print("Not solved to optimality. Optimization status:", m.status)
```

Listing 3: Python code for using ChatGPT and OptiGuide to revise formulations.

```
# test Gurobi installation
import gurobipy as gp
from gurobipy import GRB
from eventlet.timeout import Timeout

# import auxillary packages
import re
import requests  # for loading the example source code
import openai
import os

# import flaml and autogen
from flaml import autogen
from flaml.autogen.agentchat import Agent, UserProxyAgent
from optiguide.optiguide import OptiGuideAgent
```

```
#%%

config_list = [
    {
            "model": "gpt-3.5-turbo",
            "api_key": "Enter your OpenAI API key here",
            "api_type": "open_ai",
            "api_base": "https://api.openai.com/v1",
        }
]


#%%

file_path = './room_assignment.py'

with open(file_path, 'r') as file:
    python_code = file.read()

code = python_code

# show the first head and tail of the source code
print("\n".join(code.split("\n")[:10]))
print(".\n" * 3)
print("\n".join(code.split("\n")[-10:]))

#%%

# In-context learning examples.
example_qa = """
----------
Question: What if a new room with capacity 4 is added?
Answer Code:
```python
number_of_rooms += 1
capacity_of_room = {f"room{j}" : maximum_number_of_students_per_room for j in range(1,
    number_of_rooms + 1)}
minimum_number_of_students_in_room = {f"room{j}" : minimum_number_of_students_per_room
    for j in range(1, number_of_rooms + 1)}
assignment_decision_variables = [(f"student{ind}", f"room{j}") for ind in
   student_data_df.index for j in range(1, number_of_rooms + 1)]
rooms = list(capacity_of_room.keys())
```
```

```
x = model.addVars(assignment_decision_variables,
                  vtype=GRB.BINARY,
                  name="x")

y = model.addVars(rooms,
                  vtype=GRB.BINARY,
                  name="y")

# Set objective
model.setObjective(sum(sum(overall_similarity(s1, s2) * x[s1, r1] * x[s2, r2] for s1,
    r1 in assignment_decision_variables if r1 == r2 and s1 < s2) for s2, r2 in
    assignment_decision_variables), GRB.MAXIMIZE)

# Constraints
## Each student should be assigned to one and only one room
for s in students:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[0] == s) ==
    1, f"student_assignment_constraint_for_{s}")

## Room capacity constraints
for r in rooms:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[1] == r) <=
    capacity_of_room[r], f"room_capacity_constraint_for_{r}")

## Minimum number of students per room constraints
for r in rooms:
    model.addConstr(sum(x[i] for i in assignment_decision_variables if i[1] == r) >=
    minimum_number_of_students_in_room[r], f"min_num_students_constraint_for_{r}")

## Same gender per room constraints
for r in rooms:
    model.addConstr(sum(student_gender[i[0]] * x[i] for i in
    assignment_decision_variables if i[1] == r) - y[r] * sum(x[i] for i in
    assignment_decision_variables if i[1] == r) == 0, f"single_gender_constraint_for_{r
    }")

```

Question: What if we require that students with ID numbers 115 and 155 are in the same
     room?
Answer Code:
```
```

````
```python
s1 = get_ind_by_id(115)
s2 = get_ind_by_id(155)

for r in rooms:
    model.addConstr(x[s1, r] - x[s2, r] == 0)

"""

#%%

doc_str = """
----------
def get_ind_by_id(sid):
    return next((key for key, val in student_id.items() if val == sid), None)
"""
#%%

agent = OptiGuideAgent(name="OptiGuide Room Assignment",
                   source_code=code,
                    debug_times=1,
                   example_qa=example_qa,
                   doc_str = doc_str,
                llm_config={
        "request_timeout": 600,
        "seed": 42,
        "config_list": config_list,
    })

user = UserProxyAgent("user", max_consecutive_auto_reply=0,
                          human_input_mode="NEVER", code_execution_config=False)

#%%
user.initiate_chat(agent, message="What if a new room with a capacity 4 is added?")

#%%
user.initiate_chat(agent, message="What if we remove room 7?")

#%%
user.initiate_chat(agent, message="Room 1 can only accomodate 1 student. Adjust the
    capacity of room 1 to 1 and set the minimum number of students it can take to 0")

```
````

```
#%%
user.initiate_chat(agent, message="Students with ID numbers 78 and 41 are siblings and
    want to stay in the same room. Make the necessary adjustments to the room
    assignment plan")


#%%
user.initiate_chat(agent, message="What if we remove student with ID 100?")

#%%
user.initiate_chat(agent, message="What if we add a 12-year-old female student with
    Student ID 2? She is affiliated with Church 13 and expresses interest in
    participating in the seminar titled 'Where did the Bible come from?'")


#%%
user.initiate_chat(agent, message= "What if we are no longer interested in maximizing
    similarity and would like to minimize similarity within rooms?")

#%% rerun
user.initiate_chat(agent, message = "What if we would like to increase the weight on
    church similarity to 10000?")

#%%
user.initiate_chat(agent, message = "What if we want to minimize church similarity?")

#%%
user.initiate_chat(agent, message = "Older students (i.e students with ages 15 and
    above) can serve as mentors. Reassign students to rooms so that there is atleast
    one mentor in each room")
```

Appendix B. Email Survey

Survey: Exploring Skills in Coding and GenAI Usage in Class

For students who took 94.867 Decision Analytics for Business and Policy at CMU in Fall 2023, this 1-minute survey aims to gather information about your coding skills and how you perceived the abilities of Generative AI (GenAI) tools, like GPT, during the time you took the course in **Fall 2023**. Please answer all questions based on your skills and experiences **during that time period**. Your responses will help shape the future of education at CMU.

There are no right or wrong answers. As a thank you, you will be entered into a random drawing for a $25 Amazon gift card upon completion.

Sign in to Google to save your progress. Learn more

* Indicates required question

How would you rate **your** skill on coding in Python? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate **your** skill on writing mathematical formulations (e.g., optimization models, equations)? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate **your** skill on debugging Python code? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate **your** skill on learning a new Python library **without** the help of GenAI tools (e.g., GPT)? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate **GenAI's** skill on coding in Python? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate **GenAI's** skill on writing mathematical formulations? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate **GenAI's** skill on debugging Python code? (Fall 2023) *

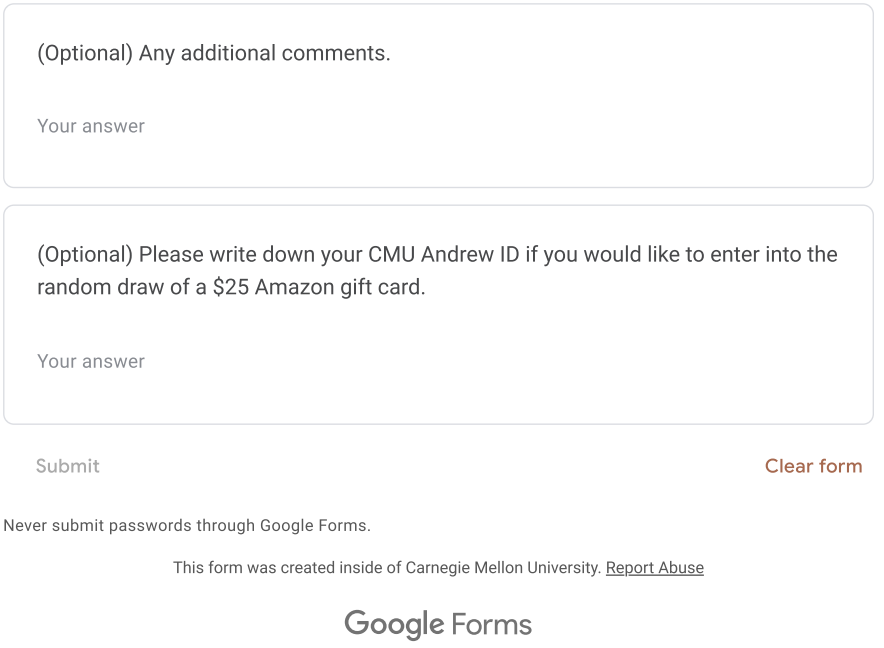
	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

How would you rate your skill on learning a new Python library **with the help of GenAI** tools (e.g., GPT)? (Fall 2023) *

	1	2	3	4	5	
Very Poor	○	○	○	○	○	Expert

For quality control, please select “4” for this question.

1	2	3	4	5
○	○	○	○	○


(Optional) Any additional comments.

Your answer

(Optional) Please write down your CMU Andrew ID if you would like to enter into the random draw of a $25 Amazon gift card.

Your answer

Submit Clear form

Never submit passwords through Google Forms.

This form was created inside of Carnegie Mellon University. Report Abuse

Google Forms

References

Bernabei M, Colabianchi S, Falegnami A, Costantino F (2023) Students' use of large language models in engineering education: A case study on technology acceptance, perceptions, efficacy, and detection chances. *Computers Ed.: Artificial Intelligence* 5:100172–102666.

Gurobi (2024) Supply network design – Gurobi optimization. Accessed May 6, 2024, https://www.gurobi.com/jupyter_models/supply-network-design/.

Ibrahim H, Liu F, Asim R, Battu B, Benabderrahmane S, Alhafni B, Adnan W, et al. (2023) Perception, performance, and detectability of conversational artificial intelligence across 32 university courses. *Sci. Rep.* 13(1):12187.

Li B, Mellou K, Zhang B, Pathuri J, Menache I (2023) Large language models for supply chain optimization. Preprint, submitted July 8, https://arxiv.org/abs/2307.03875.

Svanberg M, Li W, Fleming M, Goehring B, Thompson N (2024) Beyond AI exposure: Which tasks are cost-effective to automate with computer vision? Preprint, submitted February 8, https://doi.org/10.2139/ssrn.4700751.

Terwiesch C (2023) Would ChatGPT3 get a Wharton MBA? A prediction based on its performance in the operations management course. White paper, Mack Institute for Innovation Management at the Wharton School, University of Pennsylvania, Philadelphia, 45.