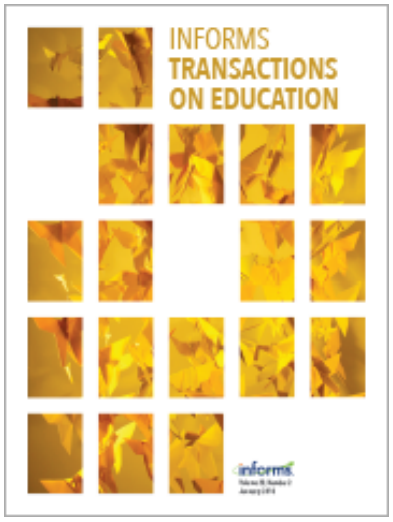


INFORMS Transactions on Education

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

Modeling First: Rethinking Undergraduate Operations Management with AI

Andrew Daw, Vishal Gupta, Paat Rusmevichientong

To cite this article:
Andrew Daw, Vishal Gupta, Paat Rusmevichientong (2026) Modeling First: Rethinking Undergraduate Operations Management with AI. INFORMS Transactions on Education

Published online in Articles in Advance 11 May 2026

. https://doi.org/10.1287/ited.2025.0168

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

INFORMS TRANSACTIONS ON EDUCATION
Articles in Advance, pp. 1–21
ISSN 1532-0545 (online)

Modeling First: Rethinking Undergraduate Operations Management with AI

Andrew Daw,[a,*] **Vishal Gupta,**[a] **Paat Rusmevichientong**[a]

[a] Department of Data Sciences and Operations, University of Southern California Marshall School of Business, Los Angeles, California 90007
*Corresponding author
Contact: andrew.daw@usc.edu, https://orcid.org/0000-0003-1758-0200 (AD); guptavis@usc.edu (VG); rusmevic@marshall.usc.edu (PR)

Received: August 21, 2025
Revised: January 19, 2026; March 16, 2026
Accepted: March 27, 2026
Published Online in Articles in Advance: May 11, 2026

https://doi.org/10.1287/ited.2025.0168

Abstract. As part of an institutional initiative to differentiate core undergraduate business classes, we have recently launched a new operations management (OM) course focused on modeling. Whereas the Traditional OM (TOM) course reviews classical models in inventory, queueing, and supply chains, the new Operations Modeling with AI (OMAI) course centers on the *practice* of mathematical modeling, particularly optimization and simulation. A distinctive element of OMAI is the integration of generative AI–based pair-programming assistants through what we call *Incremental Prompting*, which prioritizes learning essential modeling skills—abstracting real situations and expressing them with mathematical precision—rather than programming syntax. We evaluated the course through an end-of-term self-assessment survey across both offerings ($n = 250$; TOM = 156, OMAI = 94). Results indicate that OMAI students reported greater comfort and confidence in understanding, valuing, and applying model-based decision making. Interestingly, these findings become more pronounced after controlling for prior coding experience, suggesting that OMAI benefits both technically experienced and less experienced learners. In survey responses, students credited the pair-programming framework with their increased comfort and understanding of modeling. Taken together, our findings demonstrate how generative AI can be leveraged in OM and OR education to reduce technical barriers and sharpen focus on core modeling skills.

Funding: A. Daw was supported by the National Science Foundation Division of Civil, Mechanical, and Manufacturing Innovation [Grant CMMI-2441387].

Keywords: undergraduate business education • Incremental Prompting • AI pair-programming • optimization and simulation modeling

1. Introduction

Traditionally, the approach to teaching operations research (OR) in engineering and mathematics has focused on the combination of models and methods (e.g., Jensen and Bard 2002). While methods are certainly important and intellectually engaging, it seems likely that between the two, models are the gateway to learning the discipline. Both a skill set and a mindset, modeling teaches students how to reason quantitatively about complex real-world problems in order to solve them. As the "simple models for complex realities" mantra suggests (e.g., Cai et al. 2019), this perspective provides a valuable toolkit across contexts.

By comparison, in operations management (OM) courses in business schools, the typical focus is neither on models nor on methods. Instead, curricula stress common trade-offs and takeaways, largely by presenting the *results* of classical OR/OM models—the economic order quantity formula, a G/G/c waiting time approximation, etc. Though this may be an effective strategy for distilling managerial insights for future business leaders, it does not teach the core skill of modeling because students are given the model "upfront." Moreover, because these populations often exhibit highly varied mathematical preparation (and interest), such courses rarely delve deeply into the underlying OR methods. Thus, traditional OM courses ultimately leave students with a poor grasp of what OR and OM practitioners actually do. Moreover, the prevailing course's perspective jumps straight to the long run and overlooks the critical initial phases. Particularly in the context of undergraduate business education, students do not graduate and simply immediately assume senior leadership roles. On the contrary, in an increasingly data-driven world, more and more entry-level roles have become increasingly tangential, if not outright reliant, on quantitatively reasoned decision making.

1.1. Organization and Contributions of the Paper

As part of a larger institutional initiative to differentiate undergraduate business courses, we introduced a version of our core OM course *specifically* focused on OR modeling as the primary learning objective. In this paper, we describe this new approach to teaching undergraduate OM, the initial success of the course since its launch in Fall 2023, and the ways generative AI has made this possible. We view the relevant takeaways of this paper and this new course for OR/OM educators as belonging to three main streams:

1. **An AI-assisted approach to center undergraduate OM education on modeling:** We claim that AI presents an opportunity for a new way for people to learn OR, specifically by teaching *modeling*. By comparison with the focus on managerial insights in the preexisting Traditional Operations Management (TOM) course, our new class on Operations Modeling with AI (OMAI) introduces modeling (particularly for ill-structured problems) through optimization and simulation as the core learning objectives. Because business education essentially requires practical applicability yet also inherently serves a student population with a wide variety of technical backgrounds, it has not previously been realistic to teach undergraduate business students to formulate and implement models at a real-world scale. Thanks to the nascent leaps of generative AI, we can now leverage AI assistants to close the coding background gap, allowing us to focus on teaching modeling in undergraduate OM.

2. **Incremental Prompting — Leveraging generative AI to reduce implementation burden:** Beyond the specific context of OMAI, we articulate and study a particular way of integrating generative AI into modeling-focused courses. Our approach, which we call "Incremental Prompting," uses AI-assisted pair-programming to support the implementation of models through a sequence of small, precise prompts that describe individual modeling components, rather than relying on one-shot prompts that generate full formulations or solutions. The core philosophy is to use AI to *lower technical barriers while sharpening students' engagement with modeling as a cognitive skill*—reducing implementation burden while keeping students cognitively responsible for modeling decisions themselves. Although developed in the context of undergraduate OM education, we believe this Incremental Prompting philosophy may apply more broadly to settings where the goal is to teach modeling and structured reasoning, rather than to automate problem solving.

3. **Preliminary evidence that this new approach to undergraduate OM works:** The results of the new course so far, measured both in terms of student achievement on learning assessments in OMAI and through a self-assessment survey on conceptual confidence among both TOM and OMAI students, show that students are indeed learning about OR. Notably, we find that OMAI students report greater confidence in their ability to model and a better understanding of the value of model-based decision making—even when controlling for prior technical background. Furthermore, when specifically asked about the value of the assistance, OMAI students overwhelmingly report that AI makes it easier to implement their models, derive decisions from them, and generally tackle complex business problems with confidence through model-based decision making.

We organize the remainder of the paper as follows. First, to set the context for our course and its learning objectives, we close this section by specifying what we mean by "modeling" in Section 1.2. Then, in Section 2, we survey related literature, including on AI in OR/OM problems and AI in education. Then, in Section 3, we introduce Incremental Prompting and describe how we use AI in our new OMAI course. Next, in Section 4, we provide a broader context on OMAI beyond its AI usage, including contrast to the preexisting TOM. In Section 5, we deepen the TOM-OMAI contrast both by comparison of learning assessments and through a survey study administered to both groups of students. Finally, we conclude in Section 6.

1.2. What Do We Mean by Modeling?

The term *modeling* is used differently across disciplines and educational contexts. Consequently, it is important to be explicit about what *we* mean by modeling and about the pedagogical goals that guided the design of OMAI. In our view, modeling is not a single skill but a process that involves several distinct stages, each of which presents different learning challenges and pedagogical opportunities.

We organize this process into three components: *abstraction*, *mathematical encoding of modeling choices*, and *implementation*. Table 1 summarizes these three dimensions of modeling difficulty and highlights how they also define a natural learning progression and how AI pair-programming in OMAI supports implementation at all learning levels.

1.2.1. Abstraction. Abstraction concerns deciding *what should be modeled* in the first place. Some modeling problems are *well structured*: the objective and constraints are largely specified by the problem statement, and the primary task is to express them correctly. A canonical example is dividing a class of students into study groups with requirements such as a maximum group size or at least one student with a particular attribute per group. The mathematical structure closely mirrors the natural language description, and, given the description, there are only a few (equivalent) models one might reasonably form.

Table 1. Three Dimensions of Modeling as a Learning Progression

Dimension	Novice skill	Mature skill
Abstraction	Well-structured modeling problems (objectives and constraints specified)	Ill-structured modeling problems (judgment and abstraction required; multiple viable models)
Mathematical encoding	Literal encodings (direct mapping to individual constraints)	Structural encodings (semantic blocks, auxiliary variables, interacting constraints)
Implementation	Manual implementation dominates attention. Focused on small-scale problems to limit debugging and avoid distracting from broader modeling goals.	Real-world applications leveraging control flow ("if" statements, "for" loops, etc.) to iteratively build large-scale models from realistic data.
How AI pair-programming supports Implementation	Removes the need to learn syntax and accelerates early model development.	Handles iteration, control flow, and data wrangling, allowing students to focus on modeling choices rather than programming mechanics.

Other problems are *ill structured*: the modeler must determine what objectives, constraints, and trade-offs are appropriate. In these settings, multiple, qualitatively different models could all be reasonable. For example, consider designing a high-quality running playlist for Spotify (part of the course final project in spring 2025). Modeling involves deciding what qualities matter (e.g., tempo, variation, progression, artist diversity, or thematic coherence) and how those qualities should be reflected in a model. The difficulty is not coding or constraint syntax or even mathematical formalism, but deciding what should even be modeled. Developing this kind of judgment—and recognizing that modeling choices are neither unique nor automatic—is a central pedagogical goal of OMAI.

1.2.2. Mathematical Encoding of Modeling Choices.

Once high-level modeling decisions have been made, they must be translated into a precise mathematical formulation. We refer to this step as the *mathematical encoding of modeling choices*. Some modeling statements admit *literal encodings*, where a single sentence maps directly to a single constraint (or small set of constraints) with a clear semantic meaning (for example, "Each study group should have at most four people").

Other modeling statements require *structural encodings*. Here, a modeling idea is realized through auxiliary variables and a collection of interacting constraints, where individual constraints may have only an abstract interpretation in isolation. Common examples include piecewise-linear cost curves (e.g., volume discounts), fixed-charge behavior, flow-conservation structures in networks, or demand-satisfaction requirements. Mature modelers naturally reason about these formulations in terms of *semantic blocks*—coherent groups of variables and constraints that jointly implement a modeling concept. Another key learning objective of OMAI is to help students recognize, construct, and reason about these semantic blocks, rather than viewing models as flat lists of constraints.

1.2.3. Implementation.

Finally, implementation refers to translating a mathematical formulation into executable code, managing indexing, debugging, and interfacing with solvers or simulation tools. This step is pedagogically important: implementing models allows students to see concrete outputs and check whether solutions "make sense." It is also *fun*. In our experience, listening to a Spotify playlist generated by a model and debating whether it really is "good" as a group can light up a room.

However, for novice modelers, implementation can easily dominate attention and obscure higher-level modeling goals. In OMAI, generative AI is used primarily at this stage, helping students implement models so that they can focus their cognitive effort on abstraction and mathematical encoding.

In summary, our primary learning objectives deliberately target ill-structured problems and structural encodings while using generative AI to lower technical barriers and support implementation. These goals directly inform the Incremental Prompting strategy and the overall classroom flow, which we describe in subsequent sections.

2. Related Literature

2.1. Generative AI in Higher Education

The use of AI, generally, and generative AI, more specifically, in education is an active subject of ongoing research with studies across educational levels investigating different aspects of education, from different stakeholder viewpoints, and with respect to different AI tools. An exhaustive summary is impossible. We focus our literature review primarily on the use of generative AI tools in higher education.

Yan et al. (2024) survey different use cases of LLMs in education, focusing on practical obstacles to adoption and ethical issues around transparency, privacy, equality, and beneficence. They identified 53 use cases under nine main categories: profiling/labeling, detection, grading, teaching support, prediction, knowledge representation, feedback, content generation,

and recommendation. Our work—leveraging GitHub Copilot for code generation—falls under "content generation." Similarly, Belkina et al. (2025, p. 2) survey case studies, discussing the challenges of "ensuring that educational technologies are used effectively, ethically, and inclusively." Bien et al. (2024) is another notable survey centering on use cases specifically in the business (both undergraduate and graduate) curriculum.

Perhaps closer to our focus is the growing literature on whether generative AI tools improve student learning in higher education. Dong et al. (2025) provide a recent survey. They note that past research has a varied focus, some authors studying the effects of a particular tool on student achievement (as measured by a standardized assessment) and others on educational learning experiences. Moreover, this research is inconclusive: in some contexts, AI tools improve student achievement, whereas in others, they have little or even a negative effect. These past works span educational contexts from nurse training (Chang et al. 2022) to massive open online courses (MOOCs) for elementary school students (Bachiri et al. 2023). Dong et al. (2025) consequently perform a meta-analysis of past work (spanning generative AI and other AI tools across multiple education levels) and find overall that the introduction of AI tools improves student achievement. Zheng et al. (2023) and Wang and Zhao (2024) also perform similar meta-analyses (again across all AI tools and educational contexts), finding positive effects of introducing AI tools on student achievement, although Zheng et al. (2023) surprisingly find smaller effects on self-perception of learning and experience. We stress, however, that there is far from a consensus on the benefits of generative AI tools. Bastani et al. (2025) conduct a large RCT with high school math students and argue that LLM tutors can hurt long-term learning when students can no longer access the tutor.

Our study differs from these previous works in an important respect. In most studies, AI tools are used to help assist in achieving the primary learning objective. For example, in Bastani et al. (2025), tutors aim to teach students mathematics, and later, students are assessed via an exam on mathematics. By contrast, in our work, pair-programming is used to help students translate code comments into executable code, but our primary learning objective is not teaching students to code. Rather, it is teaching students how to model, and the relevant modeling steps are still performed by hand and assessed with "paper and pencil." (See Section 4 for further course details.) The use of pair-programming tools such as Github Copilot is to develop code that augments understanding of the model and connects material to real-world applications. Thus, whereas one might plausibly argue that Copilot hinders students' learning of Python, it is less clear how it affects their learning of modeling skills.

In this respect, our use of pair-programming more closely resembles the use of software tools throughout STEM education. Recently, generative AI tools have been used in introductory statistics and data science courses (Bien and Mukherjee 2025, Bray 2025). Within OR, there is a long history of using spreadsheets (i.e., Microsoft Excel) for teaching both simulation (Evans 2000, Eckstein and Riedmueller 2002, Hill 2002) and optimization (Pachamanova 2006, Martin 2010, Mason 2012). Web-based tools for simulation (Dobson and Shumsky 2006, Snider and Balakrishnan 2013) and interactive games (Chen and Samroengraja 2000, Griffin 2007, Pasin and Giroux 2011) are also common, particularly in supply chains. For optimization problems, specifically, authors have also advocated for various mathematical modeling languages (Fourer et al. 1990, Lofberg 2004, Grant et al. 2006, Mason 2013, Dunning et al. 2015, Dunning et al. 2017, Bynum et al. 2020, Chen and Xiong 2023). The plethora of such software tools and best practice articles for them is strong evidence that there is wide agreement across the field that these types of tools support and promote student learning.

We see our use of GitHub Copilot to translate code comments to executable Python code as philosophically similar to these existing use cases, but we also note a few advantages. Relative to Excel, coding in Python allows one to more easily represent large and complex optimization problems and simulation contexts. For example, when using Excel's Solver to model a linear optimization problem, one is essentially forced to set up individual linear constraints in matrix form. It is difficult, if not impossible, to represent the iteration concepts so often encountered in real-world constraints: "for every arc in the network, inflow equals outflow" or "every job should be assigned to one worker." Often, this makes it difficult for students to appreciate this level of abstraction when thinking about families of "related" constraints in large-scale applications. By contrast, these iteration concepts are naturally described with for loops and iterable collections (e.g., Python dictionaries). And whereas one could, in principle, model them using a proprietary language such as AMPL or JMP, doing so requires students to learn the programming syntax of that language while they are simultaneously struggling to learn underlying optimization concepts. By leveraging generative AI's ability to parse natural language text, students can focus on concepts and avoid syntax entirely. Finally, once students understand iteration over constraints and variables, they can easily model large, real-world-scale optimization problems that can motivate and inspire them. Similar arguments can be made with respect to creating discrete event simulations. Indeed, writing such simulations using Crystal Ball or @Risk can be notoriously finicky, requiring a

great deal of "spreadsheet planning" to lay out data and computations that obscures the fundamental concepts in simulation, whereas working directly in a programming language such as Python requires a detailed discussion of syntax. In these various respects, we see our use of GitHub Copilot with Incremental Prompting as a potential improvement on existing software tools supporting OR/OM education.

2.2. Using Generative AI to Formulate Mathematical Optimization Problems

We contrast our work with recent attempts to use generative AI to translate natural language descriptions of optimization problems into executable code or mathematical formulations. This task was the subject of the recent NL4Opt competition (Ramamonjison et al. 2022). Various approaches have emerged for the problem. AhmadiTeshnizi et al. (2025) develop a system that takes in the natural language description of an optimization problem and attempts to formulate it, code it for a solver, call the solver, and debug its own errors. Xiao et al. (2024) propose an alternate "chain of experts" strategy that links together different LLMs with prescribed subtasks to achieve a similar goal. Other authors have looked at more specialized tasks. For example, Lawless et al. (2024) use natural language descriptions of constraints to formulate and solve a constraint programming problem for meeting scheduling. Importantly, all of these approaches are *not* aimed at students. They are meant to fully replace the need for an expert modeler and are perhaps best suited to application contexts where established models exist but are hard to tailor to an organization's specific needs.

Moreover, many of the benchmarks in this literature naturally focus on *well-structured modeling problems* with relatively canonical formulations. This focus is largely pragmatic: to evaluate large-scale benchmarks, it must be possible to programmatically verify whether an AI-generated formulation is correct or equivalent to a reference model. Ill-structured problems, which admit many qualitatively different but reasonable formulations, are difficult to include in such evaluations.

By contrast, we are interested in teaching modeling (especially for ill-structured problems) as a skill and hence use a less "black-box" approach in our classroom. In the context of optimization, this involves (i) developing a paper-and-pencil model formulation (as in traditional optimization courses) and then interactively (ii) writing very explicit code comments to GitHub Copilot to describe variables and constraints while (iii) reviewing the few lines of code generated by Copilot as it "translates" each comment. Steps (ii) and (iii) are performed incrementally, one comment at a time, so that students can see the model organically come together, in line with the Incremental Prompting framework described in Section 3.

Overall, students must still reason carefully about the problem context to identify variables and constraints themselves, engaging in modeling in much the same way as a mature operations researcher. We view this engagement as critical for learning fundamental modeling skills and therefore design our assessments around it.

2.3. Student Achievement vs. Student Self-Assessment

We conclude by noting that, unlike some previous work that measures student achievement via standardized testing, our primary survey endpoint is a self-assessment of learned modeling skills. Indeed, given the practical constraints of our teaching environment—two tracked core classes with overlapping but distinct course topics and a need for consistency across sections and semesters—administering an appropriate standardized test across both groups and incentivizing them to perform well was logistically impossible. (See Section 4 for more details on our learning context.) Admittedly, our survey results might have little information about actual cognitive learning, as Falchikov and Boud (1989) have shown at best a weak correlation between self-assessment and actual skills, with a wide variability across settings and contexts. Worse, when a correlation does exist, the seminal work from Kruger and Dunning (1999) shows that less-skilled people tend to systematically overestimate their abilities, whereas those more skilled tend to underestimate. Finally, there is a known strong confounding between instructional methods and the relationship between self-assessment and ability. For example, Deslauriers et al. (2019) show that under active learning, students tend to learn more but perceive that they have learned less because of the additional cognitive effort required.

All that said, we still believe that our survey endpoint is useful to educators. Sitzmann et al. (2010) show that self-assessment is a good predictor of self-motivation and satisfaction, whereas Duckworth and Yeager (2015) show that as a measurement, it is sensitive to experiential aspects of learning (like belonging and confidence) in a way that traditional standardized testing is not. Multon et al. (1991) and Chemers et al. (2001) show a link between self-efficacy—a student's belief in their own ability to successfully perform a task—and academic persistence and academic performance. In these respects, while fully recognizing a rigorous assessment of differences in student achievement is still needed, we do believe our survey results are informative for educators experimenting with this style of course in their classrooms.

3. Incremental Prompting, Not One-Shot Prompting

We first describe a core component of the design of OMAI: Incremental Prompting. Incremental Prompting encapsulates both the philosophy and mechanics of how we use generative AI assistance in class. Unlike *one-shot prompting*—where a student provides a complete problem description to a chatbot and requests a full formulation or implementation in a single query—Incremental Prompting breaks implementation into a sequence of smaller precise prompts that correspond to individual modeling components. In practice, one-shot prompting often performs passably on well-structured problems with canonical formulations, but it is poorly aligned with educational goals around modeling. In particular, for ill-structured problems—where modeling judgment and structural encodings matter most—one-shot prompting can avoid or even replace the act of modeling itself, rather than support it. By contrast, the highly detailed, fine-grained prompts involved in Incremental Prompting reinforce our pedagogical objectives. Viewed through the modeling taxonomy introduced in Section 1.2, Incremental Prompting is deliberately designed to support the *implementation* stage of modeling while leaving abstraction and the mathematical encoding of modeling choices squarely in the hands of students.

Let us make these ideas more concrete via an OMAI-based example. This particular example is intentionally well structured, with a largely canonical formulation, making it useful for illustrating the mechanics of Incremental Prompting without introducing additional abstraction complexity. In our first week of linear optimization, we develop an advertisement allocation problem following Chickering and Heckerman (2003). After formulating the problem on the board, we want to implement the model and obtain its optimal solution. With Incremental Prompting, students translate the model to code piece by piece—defining variables, objectives, and constraints in sequence, with each modeling component expressed as its own prompt.

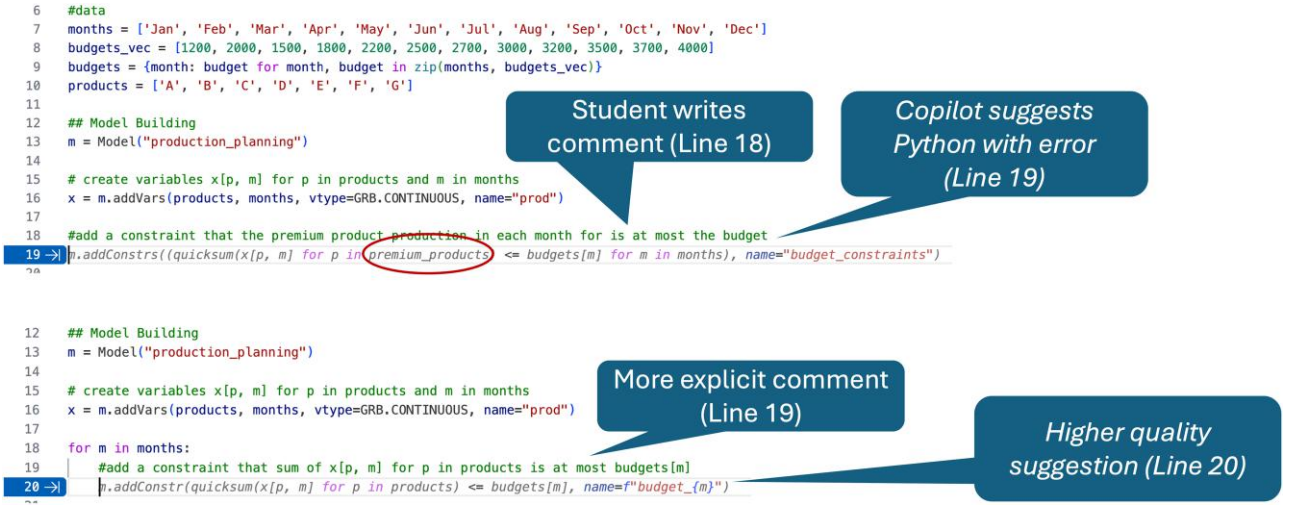
As part of Incremental Prompting, we heavily leverage an AI-based pair-programming assistant within an integrated development environment (IDE), specifically the GitHub Copilot extension in Visual Studio Code (both of which are available free to students). Like ChatGPT and other well-known chatbots, pair-programming assistants are also based on large language models, but the way users interact with them is quite different. Pair programmers essentially provide AI-supercharged autocompletes within the IDE: users write plain-language comments within the code, and then the AI assistant suggests at most a handful of lines of actual code that effectuate the comment. Returning to the ad allocation example, if we were using pair-programming to implement a constraint that the total impressions for ads A through E allocated to the "News" page of the site do not exceed the 2,000 forecasted views via the Gurobipy package for Python, we might write a comment such as "`# add a constraint that the sum of AN, BN, CN, DN, and EN is less than or equal to 2000, and label the constraint "News",`" to which the pair-programming assistant may then provide a line of code such as "`m.addConstr(AN + BN + CN + DN + EN <= 2000, "News")`" in response. To fully implement the linear program (LP) via Incremental Prompting, an OMAI student would write a similar comment for each component of the model. Notice that to do so, a student does *not* need to know the syntax or other particulars of the coding language: they simply need to precisely describe the model.

This advertisement example showcases both the advantages and disadvantages of Incremental Prompting. First, highly specific prompts for short code fragments reduce opportunities for error in AI-generated output, and their brevity makes mistakes easier to detect. Both features promote accuracy in the AI-generated code, an especially important benefit for students who are new to modeling. As demonstrated in the example shown in Figure 1, Incremental Prompting can still produce erroneous code, such as when the pair programmer hallucinates a variable name that does not exist within the local model file (but may exist in a different open file or code base). However, the AI's mistake is actually the OMAI instructor's teachable moment: to reduce the propensity of such errors, we teach students to write their comments in mathematically precise, low-entropy language. In doing so, their code is not only more likely to be accurate, but we also keep their eyes on the real learning objective, which is mathematical modeling.

Similarly, and *far* more importantly, Incremental Prompting requires students to think through the primitive "elements" of a model. These elements might be a set of variables representing production, a family of constraints capturing flow conservation or, in the case of the ad example, the number of ads to allocate to different sections of the website. They do not necessarily correspond to a single line of code but, instead, to a distinct semantic idea in the model, often implemented through a collection of variables and constraints that together form a structural encoding. Experienced modelers often reason about large models in these kinds of "semantic chunks." By encouraging students to prompt piece by piece, we are implicitly teaching them to decompose and conceptualize models in this way. We see this as a key approach for teaching structural encodings (cf. Section 1.2).

A drawback of Incremental prompting is that it is deliberately slow. It requires students to take on the almost tedious task of carefully describing each piece

Figure 1. The Incremental Prompting Workflow



Notes. Students write short, explicit comments describing self-contained modeling steps. They proceed *incrementally* to build up an executable model. In the first example shown in (a), a very generic prompt leads to a hallucination, referencing an undefined variable, *premium_products*. In the second example shown in (b), a more explicit, mathematical form of the prompt elicits a higher-quality suggestion.

of their model to the AI assistant. Particularly for expert modelers who are not learning while implementing, there are much faster ways to use AI to translate a model to code (e.g., one-shot prompting plus a post–"clean-up" pass). On the other hand, for students in an introductory course such as OMAI, we believe this trade-off of speed for understanding is not only well worth it, but it is also with good historical precedent.

Indeed, it is useful to draw an analogy to an earlier technological shift in education: the introduction of calculators in mathematics classrooms. Although initially controversial, educators ultimately recognized that calculators didn't make math obsolete. They shifted what it meant to be mathematically fluent—away from computation toward conceptual understanding.

More specifically, consider the familiar pencil-and-paper algorithm for multiplying two numbers: lining the numbers up, multiplying digit by digit, "carrying the one," and drawing a neat row of partial products. This procedure, often called the "place-value" algorithm for multiplication, emerged in 15th-century Europe among merchants and clerks who needed reliable and replicable arithmetic to facilitate commerce. It was later codified in Robert Recorde's *The Grounde of Artes* (the first arithmetic textbook written in English (Recorde 1543)), and by the 18th and 19th centuries, it had become embedded in the English education system (Easton 1967). It was prized for its uniformity, replicability, and clerical precision, rather than for any advancement of fundamental mathematical understanding (Howson 1982).

In modern times, with calculators widely available, efficiency and replicability are no longer the key bottlenecks in arithmetic. The difficulty for students isn't in multiplying two large numbers; it's deciding if multiplying those numbers is the right approach to solving a problem. The National Council of Teachers of Mathematics (1989) framed this shift well: we should view computation as a means, not an end, and technology should be used to extend understanding rather than replace it. In this spirit, the common core (see National Governors Association Center for Best Practices and Council of Chief State School Officers 2010) no longer introduces multiplication through the place value algorithm. Students instead learn partial-products and area-model methods that expose the inner logic of multiplication: the distributive property, decomposition into tens and ones, etc. These newer methods are slower and sometimes more error prone, but they prioritize conceptual transparency over procedural speed, laying the groundwork for algebraic manipulation and order-of-magnitude reasoning later in the curriculum.

We see a similar inflection point for modeling with AI-based assistance. Just as the calculator shifted focus away from computation toward mathematical understanding, AI-based pair-programming can shift attention away from coding syntax and boilerplate toward modeling structure, assumptions, and interpretation. However, realizing this pedagogical shift depends on *how* the tool is used. In particular, Incremental Prompting encourages students to build models in small, inspectable steps, making their modeling choices explicit and reviewable as the formulation emerges. This workflow naturally cultivates metacognitive habits that are specific to model building: students must articulate the intent of each modeling component, verify that the

generated code matches that intent, and continually check intermediate outputs for reasonableness and consistency with the underlying modeling assumptions. In other words, students do not merely obtain working models; they learn to *think about* models as they construct them. And, like the partial-products approach to multiplication, although the approach is arguably slower and possibly more error prone than a one-shot prompting approach, we argue the pedagogical benefits merit the trade-off.

Finally, this historical analogy also shapes some of our classroom choices. With the introduction of calculators, assessments were often split into two sections: with a calculator and without. The consensus with the educational literature was that each type of assessment measured something different (Wolfe 2010). Inspired by this distinction, we also adopt a two-tiered assessment approach in our course, particularly for exams, where an in-class "modeling" portion is completed entirely with pencil and paper without AI, and an at-home "solving" portion allows AI usage. (See Section 4 for more details on assessments in OMAI.)

In summary, Incremental Prompting is a pedagogical tool for teaching modeling, rather than an approach aimed at automatically generating formulations to replace modelers. The role of AI in this introductory course is to support implementation and is hence orthogonal to our primary learning objectives. The AI is never used to develop a model itself.

4. Curricular Context and Overview of OM Courses at Our Institution

In this section, we provide an overview of undergraduate operations management education at our institution and context for the two core classes: TOM and OMAI. We provide this context to clarify some of the institutional constraints on our survey design and learning objectives.

At our institution, there is simply one "business administration" major for all undergraduates at the business school, which is actually the largest single major at the university. A small fraction of undergraduate business students are enrolled in one of a handful of double/interdisciplinary majors, including "business and cinematic arts," "business and accounting," and (most relevant for our paper) "computer science and business administration," but the majority of students are simply business majors.

TOM is a long-standing core course required of all undergraduate business majors, typically taken in the sophomore or junior year. It is considered one of the more quantitatively demanding courses in the curriculum. OMAI, first launched in Fall 2023, is the focus of this paper. Both courses fulfill the same core requirement and share the same prerequisites, and students are free to choose between them. As previously noted, the two courses differ significantly in content, pedagogical orientation, and assessment design. Informally, TOM aims to educate future managers about the processes and trade-offs commonly encountered in operations, whereas OMAI emphasizes modeling and model-based decision making.

4.1. The Traditional Operations Management Course

TOM is likely a familiar course to anyone who has taught OM in undergraduate business education. As a core course, it aims to provide future managers with a foundational understanding of the processes and trade-offs that typify operational contexts. Its primary objective is not to develop students as modelers or technical specialists but, rather, to equip them with an operational mindset: to understand how decisions about things such as capacity, inventory, quality, and pricing affect organizational performance, broadly speaking. Designed for undergraduate business majors, TOM emphasizes breadth over depth, with a focus on conveying managerial insights.

At this introductory, overview level, the OM canon is well represented in TOM. The subjects covered include process analysis (e.g., bottlenecks and throughput), waiting and service management (e.g., Little's law, psychology of wait, and an average wait approximation formula), optimization (e.g., small LPs and integer programs (IPs), Excel Solver, sensitivity analysis through Solver, and decision trees), revenue management (e.g., linear demand models, price differentiation), and inventory and supply chain (e.g., EOQ, Newsvendor, reorder point policies). Additional topics such as forecasting may be included by instructor interest, as can interactive game sessions, such as the beer game (Chen and Samroengraja 2000).

Like most core courses at our institution, TOM is delivered in person twice a week using a primarily lecture-based format. Though the inclusion of any resources is up to instructor preference, the class typically draws on classic OM textbooks (e.g., Cachon and Terwiesch 2008). Learning assessment in TOM is primarily exam based, with quizzes and tests being primarily quantitative in nature, as based on application of the formulas introduced in the various topics covered in class. Although questions may introduce slight twists or modified scenarios, students are not asked to model a problem from scratch. As aligned with the managerial focus of the course, many examples and assessments are case based.

4.2. Our New Modeling-Focused Alternative: Operations Modeling with AI

Given the sheer size and variety of interests within the population of undergraduate business students at

our institution, there has been a recognition over the last few years that our core courses could benefit from product differentiation. OMAI was developed to offer an alternative path to operations education for students interested in getting their hands dirty with modeling, data, and analytical tools. Whereas TOM emphasizes operational reasoning, OMAI focuses on the technical formulation and computational solving of quantitative decision problems. However, by comparison with OR courses as found in engineering programs, OMAI does not include algorithmic specifics, and it certainly does not present proofs. Its central aim is to build students' capabilities as modelers and quantitative reasoners—individuals who can abstract real-world systems into formal mathematical representations, apply algorithmic tools to obtain solutions, and, critically, translate those solutions back to decisions in practice. At its core, OMAI is a class about mathematical modeling.

4.2.1. Student Population. The course remains targeted at business majors and requires no prerequisites beyond those for TOM, but it presumes an interest in—or willingness to engage with—technical material. Students must choose to take either TOM or OMAI in order to satisfy the requirements of the degree. Though OMAI has grown in each of the four semesters in which it has existed so far, TOM is indisputably the larger course. In the most recent semester, spring 2025, there were three sections of OMAI and nine of TOM. OMAI sections are also smaller by design, capped at approximately 40 students each, by comparison with TOM, which has 70 seats per section.

4.2.2. OMAI Course Topics. OMAI covers two common types of OM models: optimization and simulation. A brief opening unit on process analysis introduces essential trade-offs in resource allocation and throughput, but the bulk of the semester is split between the two modeling approaches. The optimization half of the semester covers LPs and IPs (specifically, mixed-integer linear programs) with a range of complexity. Topics covered include tightness of constraints and sensitivity analysis, variable indexing and constraint enumeration for large-scale problems, logical constraints in binary optimization models, and linearization of problems through the addition of new decision variables or constraints. By comparison with the Excel-based optimization models covered in TOM, a central theme in OMAI is the translation of operational situations into a general mathematical form, which often involves construction of scalable models that are both not well handled by the rigidity of a spreadsheet and not likely to satisfy the limitations on problem size as set by the default Solver limitations.

Then, in the second half of the semester, OMAI focuses on Monte Carlo simulation. With the course being in its relative infancy, the simulation modeling material has varied across semesters, but each iteration has begun with a review of probability distributions as models of randomness, an introduction of decision trees for an initial manner of making decisions under uncertainty, and a primer on the essentials of discrete event simulation. Then, in the advanced topics that follow, queueing models have been covered every semester, whereas Markov chains, stochastic optimization, and dynamic programming have each been among the rotation in special topics of instructor interest.

Importantly, OMAI is deliberately designed to span a learning progression across the modeling dimensions introduced in Section 1.2. Students are expected to work with both *well-structured* problems, where objectives and constraints are largely specified and most modelers would converge on similar formulations, and *ill-structured* problems, where core modeling decisions are open-ended and multiple qualitatively different models may all be reasonable. For example, students formulate and solve an optimization model for scheduling office hours during finals week—an exercise with a largely canonical structure—but they also design and implement models to personalize a Spotify playlist based on user listening history. In the latter case, modeling requires judgment: what makes one playlist better than another, how user data should be incorporated, and even whether a modeling feature belongs in the objective or as a constraint are all design choices rather than matters of correctness. Developing the ability to reason through this subjectivity, and to recognize that modeling is not unique or automatic is a central learning objective of OMAI.

4.2.3. OMAI Daily Class Flow. Like TOM, OMAI also meets twice per week, and class sessions are a mix of lecture content and interactive problem-solving sessions. Unlike TOM, each class session in OMAI involves what we refer to as a "three screens" presentation of the day's material: one screen for the problem prompt, one "screen" (or whiteboard) for the model, and one screen for the code. Each problem-solving session begins with a problem prompt, and then we develop the model together both through collective discussion as a class and distributed breakout discussions or individual deliberations. Once we have developed and discussed the model, we will then work on implementing the model together, consistently referring back to the model on the board as we translate each piece of the formulation to code via Incremental Prompting. Students are also directed to keep the same threefold activity throughout the session: a screen (or browser tab) for the

prompt, paper for the model, and a screen (or Jupyter notebook) for the code.

4.2.4. Software Tools Used in OMAI. OMAI does not rely on a single textbook or proprietary software platform. Instead, the course is organized around a carefully chosen suite of freely available tools for optimization, simulation, and AI-assisted programming. Two principles guided these choices. First, all core tools had to be available to students at no cost, ensuring equitable access. Second, we sought tools that students could plausibly take with them into summer internships or early-career roles, facilitating transfer of course skills beyond the classroom.

To date, we have used Python as the sole programming language for both optimization and simulation. Students work primarily in Jupyter notebooks (`.ipynb` files) within the freely available Visual Studio Code (VS Code) environment. For students without a prior Python installation, we direct them to the Anaconda distribution while encouraging development within VS Code to enable use of AI pair-programming tools. For AI-assisted implementation, we use GitHub Copilot via GitHub's education program, which provides free access for students and instructors. Copilot integrates directly into VS Code and supports the Incremental Prompting workflow described in Section 3. For optimization, we rely on the academic licenses provided by Gurobi through `gurobipy`, whereas simulation components primarily use standard scientific Python libraries such as NumPy and SciPy.

At the time of course design (Fall 2023), this combination represented the best freely available ecosystem that jointly supported mathematical modeling, optimization, and AI-based pair-programming. That said, helping students install this software stack at the beginning of the course was a nontrivial hurdle, particularly for students accustomed to cloud-based tools. Early-semester setup is one of the most operationally challenging aspects of the course. We hope that as the software landscape continues to evolve, additional free or low-friction alternatives will emerge, and we plan to revisit these choices in future iterations of the course.

Finally, it is worth emphasizing that this software stack does not require students to enter the course with prior coding experience. OMAI assumes no background in Python or programming more generally. By pairing Incremental Prompting with industry-standard tools, the course allows students to focus on modeling concepts while gradually building practical fluency with tools they are likely to encounter again in professional settings.

4.2.5. Learning Assessments. Both the model-first philosophy and the incorporation of pair-programming are reflected in the assessments and exercises in OMAI. Because practical implementation is a core value for a business education context, students are given several project-based assignments throughout the semester in which they must work through the full cycle of model formulation (in many cases, based on real-world data), implementation to code, and interpretation of the solution. On the assignments, students will be assessed on each of these fronts, and they have all resources available to them. On the exams (typically one for optimization and one for simulation), modeling and solving are decoupled and assessed separately through two different exam segments. The first segment asks students to develop a model for a specific scenario in an in-class, closed-notes, pencil-and-paper modeling assessment. Then, a second at-home segment asks students to implement and solve a different specific model and then obtain insights from it. All resources, including pair-programming assistants, are available to students on the solving segment.

Like we discussed for the calculator analogy in Section 3, we feel that there are different skills involved in modeling (without AI help) and solving (with AI help), and thus, we feel it is important to assess these skills separately. Moreover, the in-class modeling segment of the exams is essential both as a measurement and as an incentive. We find it is easier to help students adhere to the Incremental Prompting approach if they know that their modeling skills will eventually be assessed without AI, and this goal creates a positive reinforcement effect, given that we have also argued that Incremental Prompting is better for their learning as new modelers.

5. Evidence of Success: Learning and Student Self-Assessments

In the four semesters that we have taught OMAI at our institution, we have observed anecdotally through learning evaluations and student interactions that the course has had impact: students have told us that the class has influenced their career paths and expanded their academic interests, and the growing enrollment suggests that the course has a positive reputation among students. We are encouraged by its success so far. (See Section 6 for our ideas on how we might improve the course in future semesters.) Nevertheless, this feedback is informal and not measured relative to its predecessor and parallel option, TOM. Moreover, because students self-select into OMAI versus TOM under institutional constraints (cf. Section 4), the two populations are unlikely to be comparable at baseline along all dimensions of preparation or interest.

On the one hand, student performance on exams, homework, and other assessments in OMAI provides meaningful internal evidence that students are able to

engage with technically demanding, large-scale, and open-ended modeling tasks. Course assessments routinely require students to work with problems that exceed the scale and scope of standard undergraduate OM coursework, ranging from extended textbook-style models to messy, open-ended problems involving real data. That students are able to formulate, implement, and reason about such models—often well beyond the computational limits of tools such as Excel's Solver—suggests substantial progress relative to their starting point at the beginning of the semester.

On the other hand, this evidence should not be interpreted as a direct causal comparison with TOM, both because the two courses differ in content and learning objectives and because student self-selection creates unavoidable baseline differences between the populations. The two courses do not share learning assessments, and designing a common assessment instrument is complicated by differences in course scope, sequencing, and learning objectives. In particular, it would not be informative to test students on topics or modeling approaches that were not part of their respective curricula.

Hence, to measure the pedagogical efficacy and to understand how generative AI impacts that efficacy, we administered a survey to students in both TOM and OMAI at the end of the spring 2025 semester. This study was reviewed by the University of Southern California Institutional Review Board and determined to be exempt from further review under Section 46.104(d) (1) (Study ID UP-25-00209). The survey was entirely anonymous in both response and record of completion, students' participation was entirely voluntary, and to prevent any potential conflict of interest among active instructors in that semester, only the first author was involved in constructing the survey, collecting the responses, and analyzing the data. Participants were recruited both in person during class time and via email to the course rosters. A total of 250 respondents participated, with 156 from TOM and 94 from OMAI.

The survey contained four main sections of student self-assessment questions: three background questions on the students' experience prior to taking either TOM or OMAI, six questions on students' comfort and confidence for general modeling content in the overlap of the two courses, five questions on the effects of pair-programming asked only to OMAI students, and two questions on students' comfort and confidence in the use of modeling concepts in a particular example. Because optimization constitutes most of the overlap between TOM and OMAI, the specific example is an optimization problem. All of the survey questions were on a Likert scale: on each question, respondents were presented with a statement and asked to select the option among "strongly disagree," "somewhat disagree," "neither agree nor disagree," "somewhat agree," and "strongly agree" that best described their agreement with the statement.

In the background section, respondents were asked to self-assess their experience with coding prior to taking TOM or OMAI, their prior experience with Python specifically, and their prior experience with pair-programming AI assistants such as GitHub Copilot. Then, the general course content section asked respondents to self-assess their comfort implementing LPs and IPs, their understanding of how LPs and IPs are used in decision making, their comfort in using LPs and IPs at work, their comfort incorporating uncertainty in business analysis and decision making, their understanding of how mathematical models can be valuable for decision making, and their confidence in developing a new model when faced with a new problem. In the pair-programming section of questions that was shown only to OMAI participants, respondents were asked to self-assess whether pair-programming assistants made it easier to implement and solve LPs and IPs, whether pair-programming assistants made it easier to incorporate uncertainty into decision making, whether pair-programming helped them understand how they can use modeling to drive decision making, whether pair-programming assistants expanded their comfort zone in implementing mathematical models, and whether pair-programming increased their confidence in solving new and complex business problems. Finally, the last section of the survey presented a specific problem scenario and asked respondents (from both courses) to self-assess their comfort in formulating an optimization model for that problem and their comfort in implementing and solving a model for that problem. The full statements of all 16 questions on the survey are available in Appendix A, and Table B.1 in Appendix B.1 contains the full distribution of response selections for each question in each of the two course populations.

Given that the survey focuses on modeling and emphasizes LPs, IPs, and Monte Carlo simulation, it should not be surprising for OMAI students to express higher comfort and confidence with this technical material. Indeed, this is what the survey results immediately reveal, and the contrast between TOM and OMAI can be striking. For instance, in the question on whether respondents report that they understand how LPs and IPs are used to make decisions and solve real-world problems (row Q5 of Table B.1), 99% of OMAI respondents indicated agreement, with the majority showing strong agreement, and none indicated disagreement: 1 out of 94 respondents chose "neither agree nor disagree" (1.1%), 36 chose "somewhat agree" (38.3%), and 57 chose "strongly agree" (60.6%). By comparison, the TOM responses were dense across the spectrum of options: 28 out of 156 respondents

chose "strongly disagree" (18.0%), 18 chose "somewhat disagree" (11.5%), 23 chose "neither agree nor disagree" (14.7%), 56 chose "agree" (35.9%), and 31 chose "strongly agree" (19.9%). The stark differences in response distributions from each course population are also apparent in the specific scenario questions. For instance, when asked about their comfort implementing and solving a given optimization model for the example problem (row Q16 of Table B.1), TOM responses contained 7 "strongly disagree" (4.5%), 22 "somewhat disagree" (14.1%), 39 "neither agree nor disagree" (25.0%), 66 "agree" (42.3%), and 22 "strongly agree" (14.1%), whereas OMAI responses contained no "strongly disagree" (0%), 4 "somewhat disagree" (4.3%), 7 "neither agree nor disagree" (7.4%), 44 "somewhat agree" (46.8%), and 39 "strongly agree" (41.5%) selections.

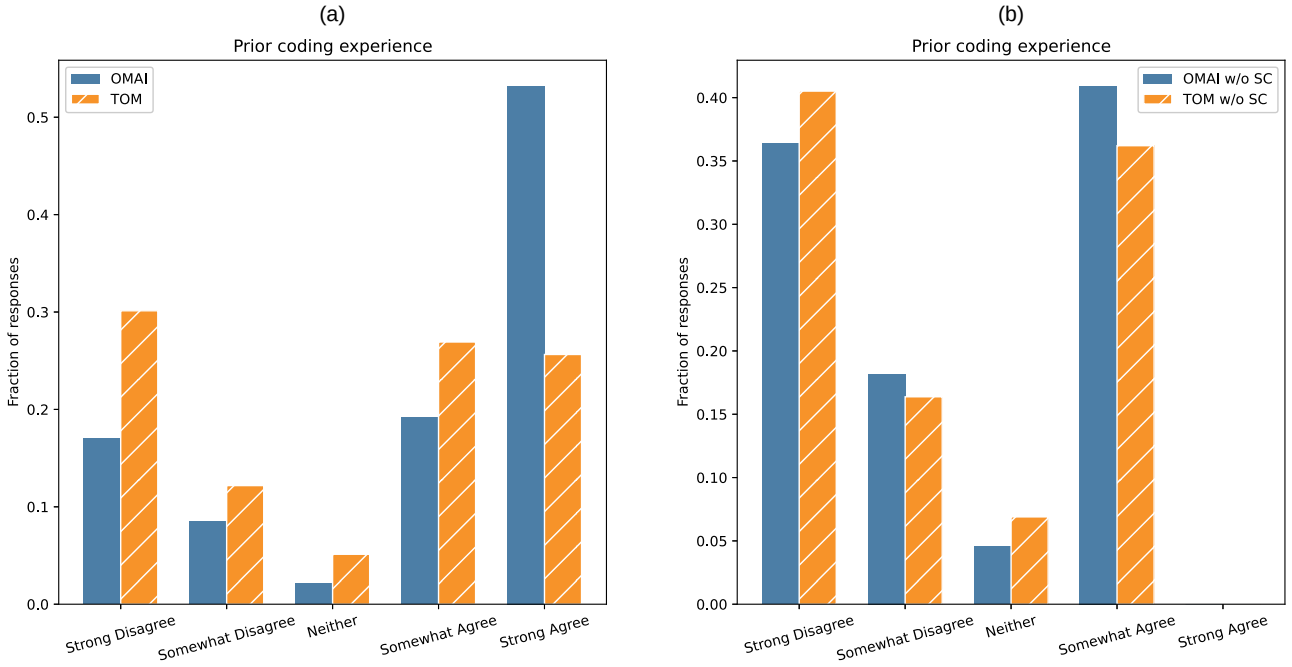
Though there is the potential for Dunning–Kruger effects in self-reporting that could blur the distinction between the two groups, we believe these immediate results are not surprising for two main reasons. First, OMAI is essentially entirely focused on these topics, and thus, students have had ample practice and experience to build the comfort and confidence that these questions ask about—that is essentially the goal of the course and, thus, what we hoped to see in the survey results. Second, as we have mentioned in the contrast of TOM and OMAI in Section 4, there is an inherent self-selection effect to this pair of courses: students with stronger prior technical background may be more inclined to enroll in the more technical course. By comparison with the first reason, the prior experience of OMAI students is not what we hope to be driving the successful outcomes of the course. With that in mind, let us take a closer look at the responses to the background questions in each course.

5.1. Stratifying Student Responses by Prior Coding Experience

Because students self-select into OMAI versus TOM under institutional constraints (cf. Section 4), the two groups are unlikely to be comparable at baseline across all dimensions of preparation and interest. In particular, prior coding experience is plausibly correlated with both course choice and comfort with modeling tools. In what follows, we therefore stratify survey responses by students' self-reported prior coding experience. This stratification does not support causal claims but serves as a coarse way to form more comparable subpopulations when interpreting descriptive survey differences.

Recall the survey had three background questions: prior coding experience, prior Python experience, and prior use of pair-programming AI assistants. Among these, prior coding experience plays a central role in our comparison, as it is both unevenly distributed across courses and closely tied to students' facility with modeling software. In Figure 2(a), we plot histograms for the distribution of responses to the prior coding question in each of the courses (which is also available in row Q1 of Table B.1 in Appendix B).

Figure 2. Prior Coding Experience Overall and Among Nonstrong Coders



Notes. (a) Histogram of respondents' reported confidence with coding prior to taking the course. (b) Histogram of respondents' reported confidence with coding prior to taking the course, conditioned on not reporting strong confidence.

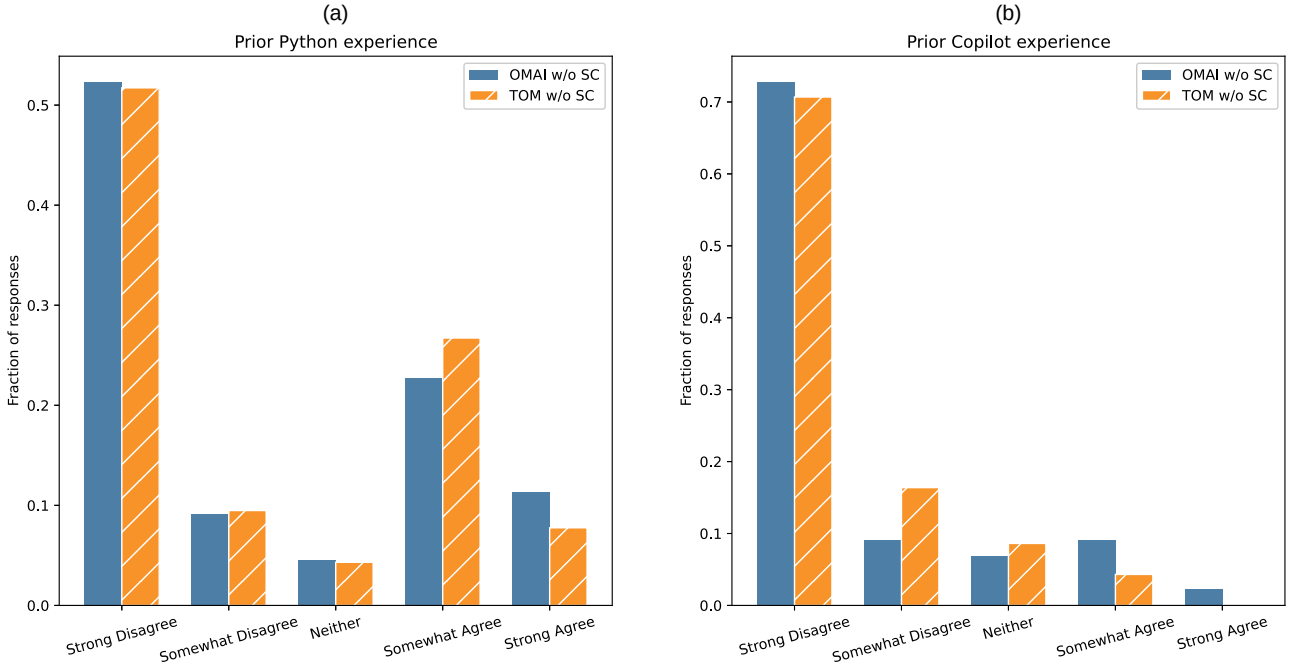
Notably, the OMAI population has a majority of respondents indicating that they had strong prior coding experience (53.2%). By comparison, barely a quarter of TOM respondents reported strong prior coding experience (25.6%), and a plurality of TOM respondents selected "strongly disagree" for the question (30.1%).

However, in Figure 2(a), the relative proportions of the bars for "strongly disagree," "somewhat disagree," "neither," and "somewhat agree" are not so dissimilar across the TOM and OMAI distributions. This observation motivates the stratified view shown in Figure 2(b). This plot shows the histograms of the survey responses when removing those that indicated strong prior coding experience, meaning the conditional distributions given that the respondent's self-reported prior coding experience was less than strong. When conditioning on not reporting strong prior coding experience, a likeness of subpopulations emerges across the two courses. In fact, the four conditional fractions of responses (from "strongly disagree" to "somewhat agree") have an absolute difference of less than 0.05 in each selection ([0.364, 0.182, 0.045, 0.409] for OMAI and [0.405, 0.164, 0.069, 0.362] for TOM).

Interestingly, this split on strong coders reveals that the course subpopulations also look more alike in the other background questions. For instance, as shown in Figure 2(b), the conditional distributions (from "strongly disagree" to "strongly agree") of prior Python experience for nonstrong coders are [0.523, 0.091, 0.045, 0.227, 0.114] for OMAI and [0.517, 0.095, 0.043, 0.267, 0.078] for TOM, and among these two groups of nonstrong coders, a very similar majority of respondents selected "strongly disagree" for prior pair-programming use (72.7% for OMAI and 70.7% for TOM). Then, similarities also appear among the strong-coder subpopulations, as shown in Figure 3. For instance, the prior Python distributions once again look quite similar, with [0.040, 0.020, 0.020, 0.140, 0.780] for OMAI and [0.077, 0.051, 0.000, 0.128, 0.744] for TOM, though, interestingly, a higher fraction of TOM strong coders indicate prior use of pair-programming assistants, with [0.380, 0.200, 0.020, 0.160, 0.240] for OMAI and [0.325, 0.100, 0.050, 0.125, 0.400] for TOM. Taken together, these patterns suggest that stratifying by prior coding experience yields student subgroups that are more comparable along other background dimensions, providing a clearer lens through which to interpret subsequent survey responses.

Of course, there are likely other important but unmeasured differences between the student populations in TOM and OMAI. Both classes have a reputation among students as being hardcore courses, but OMAI is both newer and positioned as faster moving and rigorous, and these self-selection effects are not measured in our data. Nevertheless, for the information available to us, controlling for prior coding background experience closely aligns the subpopulations

Figure 3. Prior Experience with Python and Copilot Among Nonstrong Coders



Notes. (a) Histogram of respondents' reported confidence with coding specifically in Python prior to taking the course, conditioned on not reporting strong confidence for coding in general. (b) Histogram of respondents' reported confidence with GitHub Copilot prior to taking the course, conditioned on not reporting strong confidence for coding in general.

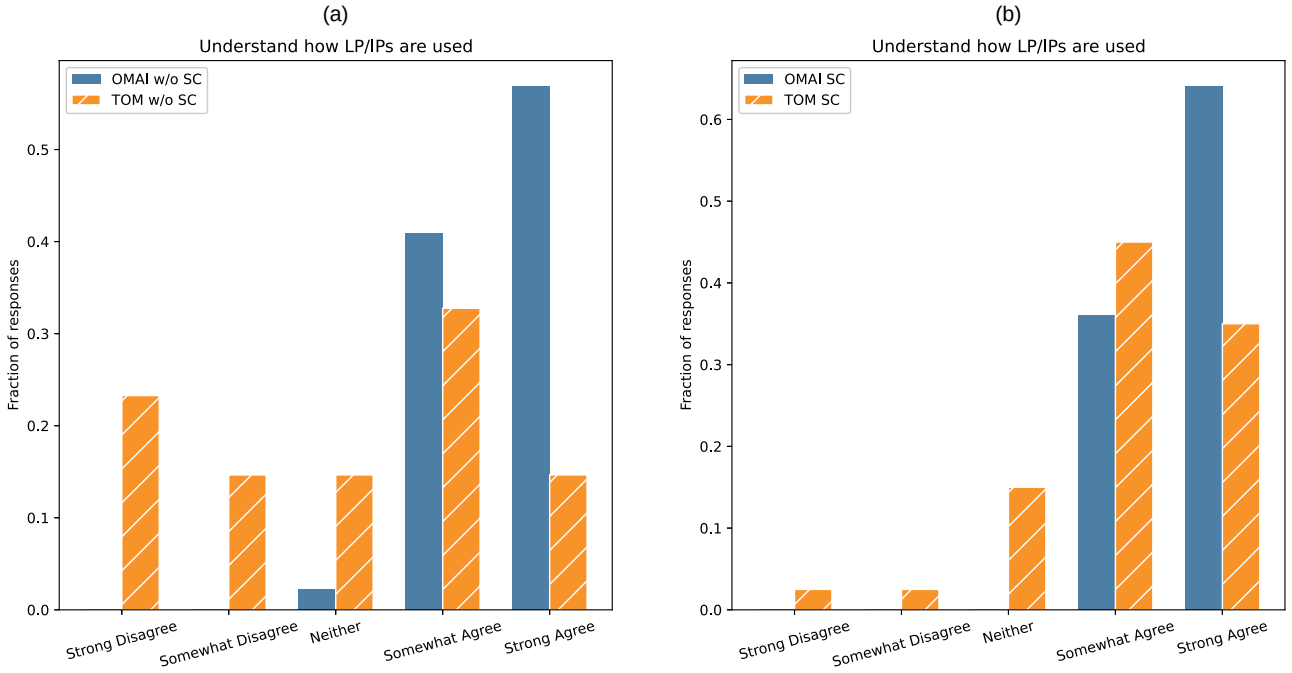
across the two courses in an unexpected way, giving us the best head-to-head comparison we can make between the courses.

Following the recognition of these similar subpopulations of differing sizes within the OMAI and TOM groups, let us review the distributions of responses to the background questions conditioned on either being from strong coders or nonstrong coders (i.e., "somewhat agree" or lower). The full conditional distributions for all questions can be found in Tables B.2 and B.3 in Appendix B for without strong coders and for exclusively strong coders, respectively. As these tables show, both the more and less technical subpopulations appear to be better served in the new class, with OMAI respondents reporting higher comfort and confidence across the board. Let us highlight these findings in a few example questions.

First, let us return to the question regarding the use of LPs and IPs for decision making, which we summarized for the full population level at the start of this section (row Q5 in Tables B.2 and B.3 in Appendix B. We now plot the distribution of responses in each subpopulation across each course in Figure 4, with nonstrong coders in Figure 4(a) and strong coders in Figure 4(b). Among nonstrong coders in TOM, the responses are fairly uniform across the options ([0.233, 0.147, 0.147, 0.328, 0.147]), whereas among nonstrong coders in OMAI, a majority of respondents report strong confidence in understanding how LPs and IPs are used to solve real-world problems, with no respondents disagreeing with the statement ([0.000, 0.000, 0.023, 0.409, 0.568]). For the strong coders, a majority of respondents in both courses report agreement, but in TOM, the mode of selections is "somewhat agree," with 20% of respondents not selecting either agree option. By comparison, all strong coders in OMAI agreed with the LP and IP usage statement, and 64% did so strongly.

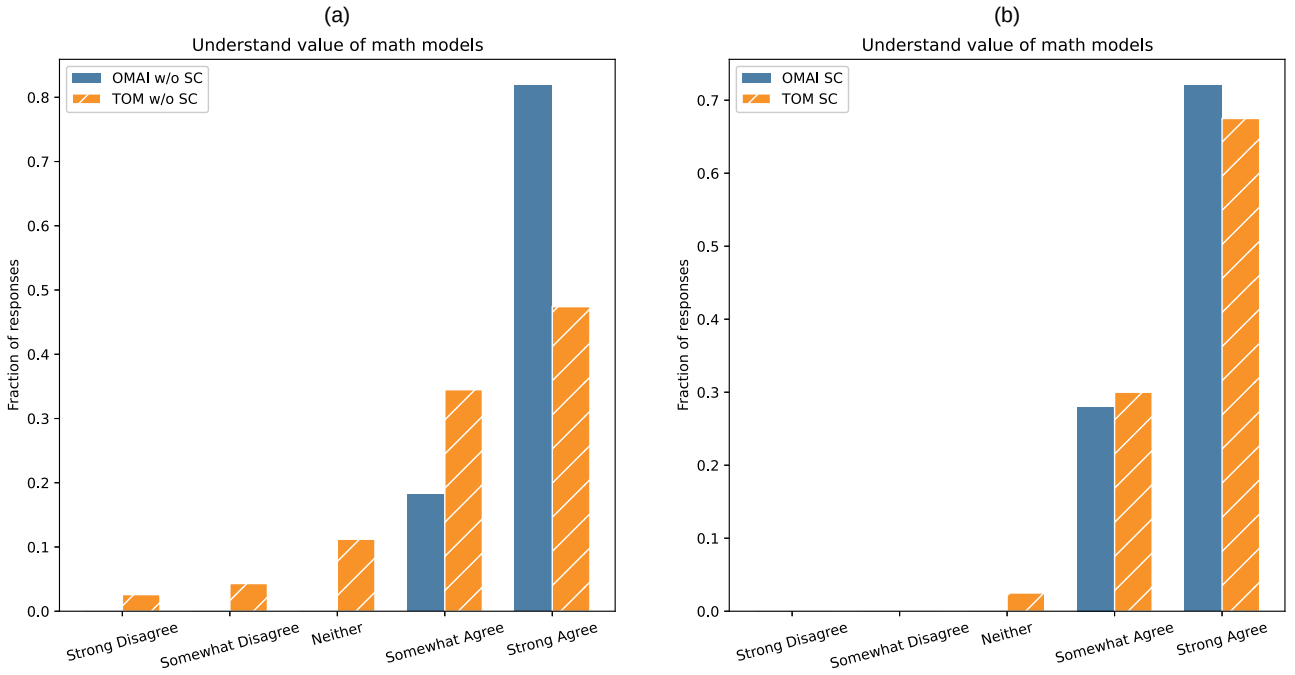
These conditional distributions can also reveal dimensions on which the less technically experienced students particularly benefit. For example, consider the question on general appreciation of modeling as a decision-making framework (row Q8 in Tables B.2 and B.3 in Appendix B), which we plot for each subpopulation in Figure 5. As one may notice in Figure 5(b), the distribution of responses for strong coders is very similar across the two courses, suggesting that perhaps students with high levels of prior coding experience already appreciated the value of modeling before taking either course. However, in Figure 5(a), we see that 82% of nonstrong coders in OMAI reported strong agreement with the statement that they "understand how mathematical models can be valuable for decision making in real-world problems." Moreover, all OMAI respondents reported at least some form of agreement. By comparison, strong agreement has a plurality but not quite a majority among nonstrong coders in TOM (47.4%), with 18% of respondents not selecting either agreement option. Thus, these survey results suggest that the new course may be more effective at conveying the value of mathematical

Figure 4. Mathematical Programming Understanding Stratified by Coding Experience



Notes. (a) Histogram of reported understanding use of LPs and IPs among nonstrong coders. (b) Histogram of reported understanding use of LPs and IPs among strong coders.

Figure 5. Valuation of Mathematical Modeling Stratified by Prior Coding Experience



Notes. (a) Histogram of reported understanding of the value of mathematical models among nonstrong coders. (b) Histogram of reported understanding of the value of mathematical models among strong coders.

modeling to students who had not been particularly familiar with models before the class.

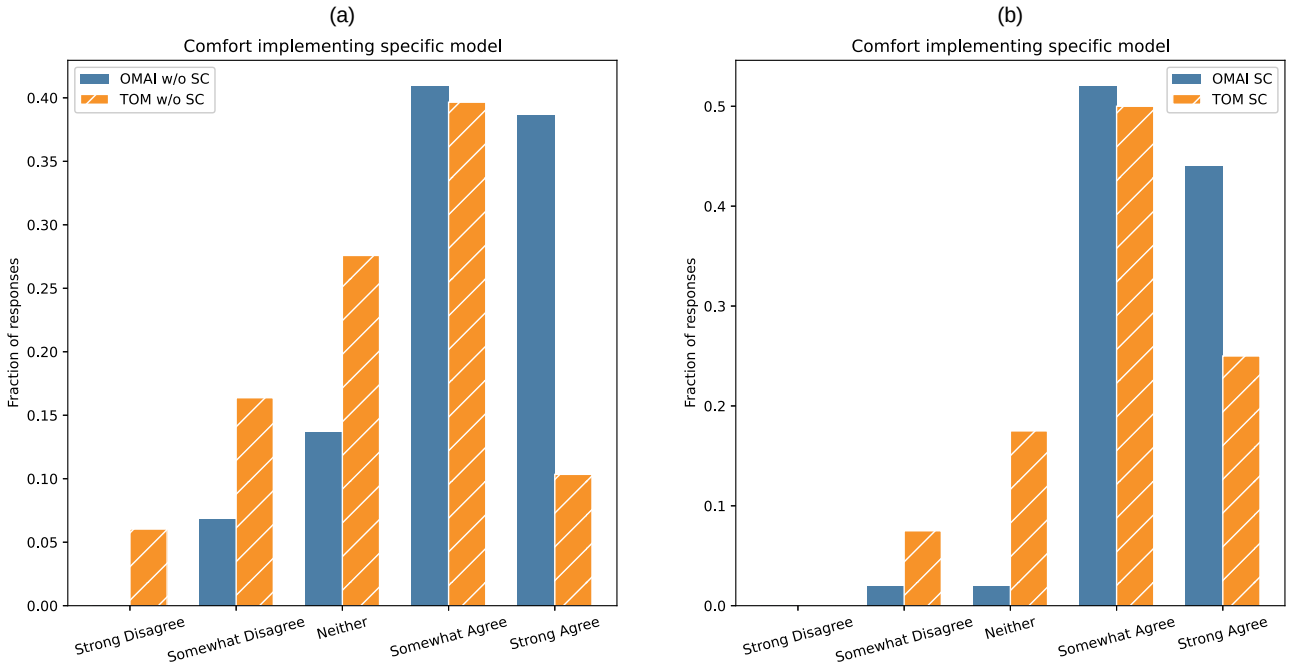
As a final example of the differences in conditional response distributions, let us return to the question of student's comfort implementing and solving a given optimization model for a specific example problem (row Q16 in Tables B.2 and B.3). These distributions are shown now in Figure 6. In Figure 6(a), we see that 80% of nonstrong coders in OMAI indicate some level of agreement that they could implement and solve a given model, with 39% indicating strong confidence in their ability to implement and solve. Though a similar fraction of TOM nonstrong coders somewhat agree with the statement, only 10% feel strongly that they could, and 41% did not select either agreement option. Among strong coders, there is a similar pattern of nearly equivalent amounts of respondents somewhat agreeing across the two courses (52% for OMAI and 50% for TOM), with more OMAI strong coders strongly agreeing (44% versus 25%) and far fewer not selecting an agreement option at all (4% versus 25%). Hence, these survey results suggest that the new course is more effective in the delivery of OM modeling skills for students at both the less or the more technically experienced levels.

5.2. OMAI Students Report Increased Confidence from AI Pair-Programming

As a final set of takeaways from these survey results, let us now consider the distributions of selections on the questions about the use of the pair-programming assistant GitHub Copilot in class, which were only asked to OMAI students (rows Q10 through Q14 in Tables B.1–B.3). Whether viewed at the full population or subpopulation levels, the majority of responses are positive on each of the five questions: students overall feel that pair-programming makes it easier to implement and solve LPs and IPs and easier to build simulations, pair-programming has helped them understand how to apply modeling to drive decision making and translate models into code, and pair-programming has increased their confidence in solving new problems. We highlight three of these questions in Figure 7. In general, we interpret these survey results to mean that pair-programming is a beneficial use of AI for students learning modeling in undergraduate business education.

However, these responses also suggest interesting subtrends for the student subpopulations. In particular, agreement with the statements is not universal, and as can be seen through either Figure 7 or in Tables B.1–B.3, the strong coders consistently outnumber the nonstrong coders in disagreement to these questions. Though the sample size of this survey is too small to draw any strong conclusions away from this, these observations may at least suggest some hypotheses. For instance, it is well-known that generative AI is prone to errors and hallucinations, and so, it is not a guarantee that the code produced by a

Figure 6. Comfort Implementing a Model Stratified by Prior Coding Experience



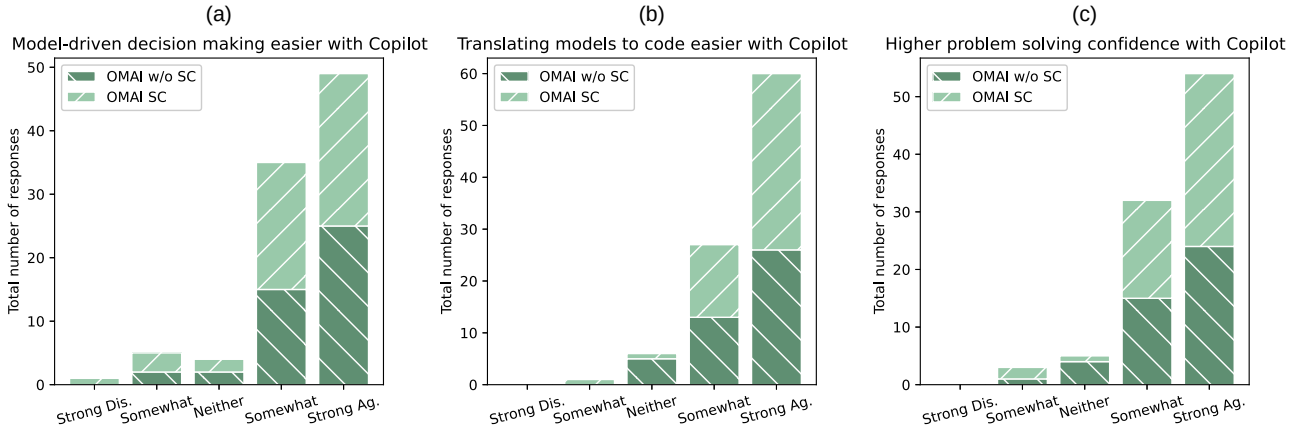
Notes. (a) Histogram of reported comfort implementing and solving a given model for a specific example scenario among nonstrong coders. (b) Histogram of reported comfort implementing and solving a given model for a specific example scenario among strong coders.

pair-programming assistant will be correct. In general, we feel that Incremental Prompting may be better equipped to manage this risk of error by comparison with wholesale code produced through one-shot chatbot prompting, but some risk exists regardless. Hence, some strong coders may be recognizing the potential flaws of overreliance on generative AI here, and some may believe they are better off without it. Furthermore, as aligned with the findings in Bastani et al. (2025), students who have strong prior technical experience may also be recognizing that they could become overly dependent on the assistance generative AI and not fully learn how to implement and solve models on their own. That hypothesis is of particular concern to us, and the balance of class time, exercises, and assessments spent with and without AI is a key focus of our reflections from teaching this course so far.

6. Conclusion: Best Practices, Future Directions, and Broader Implications

We conclude by summarizing a set of best practices that emerged from our experience designing and teaching OMAI, followed by directions for future course

Figure 7. Student Reflections on AI Pair-Programming in OMAI



Notes. (a) Histogram of reported higher ease of model-driven decision making with Copilot. (b) Histogram of reported higher ease in translating models to code with Copilot. (c) Histogram of reported higher problem-solving confidence with Copilot.

development. Whereas our discussion is grounded in undergraduate operations management education, many of these lessons extend more broadly to teaching modeling and analytical reasoning with generative AI support.

6.1. Best Practices

- *Clearly communicate learning objectives and the role of AI.* A key lesson from OMAI is the importance of making the course philosophy explicit to students. We are deliberate in explaining that modeling is an intellectual activity that remains their responsibility, whereas AI is used narrowly to reduce implementation friction. This distinction helps students understand that success in the course depends on reasoning about objectives, constraints, and assumptions before involving AI. This clarity is particularly important given emerging evidence of skill regression when automation displaces core cognitive tasks, documented in settings ranging from mathematics education (Bastani et al. 2025) to expert medical practice (Budzyń et al. 2025). By explicitly framing AI as a support tool rather than a substitute for modeling judgment, we have found that students engage more thoughtfully and align their behavior with course learning objectives.
- *Prioritize modeling before any use of AI.* A central design principle in OMAI is that modeling precedes implementation. In each class session, we devote substantial time to discussing the underlying business context and developing the optimization or simulation model on the whiteboard or on paper, without using any AI tools. Only after students demonstrate a clear understanding of objectives, decision variables, and constraints do we transition to implementation. This sequencing reinforces the idea that AI supports model execution, not model conception.
- *Tailor in-class implementation time to student backgrounds.* Student cohorts vary widely in prior coding experience, and the appropriate balance between modeling and implementation time should reflect this heterogeneity. In cohorts with strong coding backgrounds, instructors may emphasize in-class modeling and defer implementation to homework or recap sessions. Conversely, when many students lack coding experience, dedicating class time to Incremental Prompting and AI-assisted implementation can be valuable. Hybrid approaches, such as optional recitations or asynchronous demonstrations, may be most effective in mixed-background cohorts.

6.2. Future Directions

One area where we see substantial room for growth is in the explicit teaching of model validation. Beyond verifying syntax or feasibility, we want students to develop the ability to assess whether a model meaningfully captures reality and whether its outputs would be appropriate for practice. Whereas OMAI currently addresses validation through activities such as scrutinizing AI-generated code, sensitivity analysis, robustness checks, and edge-case reasoning, we view this as a critical area for further pedagogical development.

More broadly, we see opportunities to experiment with alternative software ecosystems, delivery formats, and assessment structures as AI-enabled tools continue to evolve. We welcome feedback and collaboration from the broader community as we refine these aspects of the course.

6.3. Broader Implications

Beyond the specifics of OMAI, our experience teaching modeling to a business audience also speaks to broader questions about where modeling education may have the greatest long-run impact.

Operations research has long been seen to face a "PR problem." This concern has appeared in reflections on the field's future and direction (Kan 1989, Corbett and Van Wassenhove 1993), as well as in organized efforts to communicate its value more broadly, including initiatives such as the "Science of Better" campaign (Sodhi and Tang 2008) and the growing emphasis on analytics (Liberatore and Luo 2010, Mortenson et al. 2015, Gorman 2021). Rarely do these concerns stem from shortcomings in the underlying methods themselves. Rather, they reflect the simple fact that many potential users and decision makers have limited exposure to modeling and to what OR practitioners actually do.

In this context, undergraduate business education represents a particularly important—and historically underutilized—setting for expanding modeling literacy. Business majors constitute the largest single category of bachelor's degrees awarded in the United States, with 375,418 degrees conferred annually (18.6% of all bachelor's degrees), compared with 123,017 degrees (6.1%) across all engineering disciplines (Irwin et al. 2023). Moreover, business students broadly reflect the demographic composition of the undergraduate population. Like we said in Section 4, business is actually the largest single major at our university. Whereas students in business programs may, on average, arrive with less technical preparation than their engineering counterparts, advances in AI-assisted implementation substantially reduce the technical barriers that have traditionally limited exposure to modeling in these settings.

From this perspective, efforts to teach modeling and quantitative reasoning within undergraduate business curricula may have a disproportionate downstream impact. By reaching students who will go on to managerial and leadership roles across a wide range of industries, such courses can help broaden understanding, adoption, and support for modeling-based

decision making—an outcome that aligns closely with long-standing aspirations of the OR community.

Acknowledgments

We are grateful to colleagues who have informed both our teaching practice and our assessment of it, including Professor Sam Gutekunst, who suggested we conduct this survey, and we are likewise grateful to colleagues who have shared materials with us from which we have based the new course, including Professor Amy Ward, who shared simulation materials, and Professor Huseyin Topaloglu, who shared optimization materials. Following that spirit, we would be very happy to share any amount of materials upon request from any prospective future instructors.[1]

Appendix A. Survey Questions

A.1. Background and Prior Technical Experience

All students were asked to rate their agreement with the following statements regarding their background prior to taking the OM course:

Q1: "I had prior experience with coding before taking BUAD 311 or BUAD 313."

Q2: "I had prior experience with coding specifically in Python before taking BUAD 311 or BUAD 313."

Q3: "I had prior experience with using pair-programming generative AI assistants (i.e., GitHub Copilot) for coding before taking BUAD 311 or BUAD 313."

A.2. Self-Assessment of General Course Content

Additionally, all students were asked to rate their agreement with the following statements regarding their general comfort with the mathematical modeling concepts common in both courses:

Q4: "I feel comfortable implementing optimization models such as linear programs (LPs) and integer programs (IPs)."

Q5: "I understand how optimization models such as linear programs (LPs) and integer programs (IPs) are used to make decisions and solve real-world problems."

Q6: "I would feel comfortable using optimization models such as linear programs (LPs) and integer programs (IPs) in projects during an internship or full-time employment, either individually or as part of a team."

Q7: "I feel comfortable incorporating uncertainty in my analysis of business problems and evaluating the impact of the uncertainty on my decisions."

Q8: "I understand how mathematical models can be valuable for decision making in real-world problems."

Q9: "When faced with a business problem that does not fit with a standard formula, I feel confident that I can think through the problem, develop a mathematical model, and derive an effective decision for the underlying problem."

A.3. Self-Assessment of Pair-Programming Usage

Students in the advanced course (BUAD 313) were asked to rate their agreement with the following statements regarding their experience with pair-programming generative AI assistants (i.e., GitHub Copilot) in the context of the course:

Q10: "Pair-programming generative AI assistants (i.e., GitHub Copilot) make it easier for me to implement and solve optimization models such as linear programs (LPs) and integer programs (IPs)."

Q11: "Pair-programming generative AI assistants (i.e., GitHub Copilot) make it easier for me to incorporate uncertainty into decision making through simulation."

Q12: "Pair-programming generative AI assistants (i.e., GitHub Copilot) have helped me understand how I can apply mathematical modeling to drive decision-making for real-world problems."

Q13: "Pair-programming generative AI assistants (i.e., GitHub Copilot) have expanded my comfort zone for translating mathematical models into scientific computing languages and libraries."

Q14: "Pair-programming generative AI assistants (i.e., GitHub Copilot) have increased my confidence in solving complex business problems that I have never seen before."

A.4. Self-Assessment of Specific Course Content and Usage

Finally, all students were asked to rate their agreement with two statements regarding their comfort with developing a model and implementing it for the following specific scenario regarding school transportation planning:

> "The Santa Barbara Unified School District (SBUSD) manages 4 elementary schools collectively serving 12 distinct neighborhoods. Each elementary school offers 6 grade levels, from kindergarten to fifth grade.
> In anticipation of the next academic year, SBUSD has compiled the following information:
> • The number of students each elementary school can support at each grade level,
> • The actual number of students at each grade level from each neighborhood,
> • The distance from each neighborhood to each school."

The survey then presented the following two questions about this scenario:

Q15: "Given exact numbers for the information described above and ample time to think, I feel comfortable formulating an optimization model (i.e., identifying decision variables, objective function, and constraints) to assign elementary-age students in Santa Barbara to schools in order to minimize the total distance traveled by students to school each day."

Q16: "Given an optimization problem formulation that is based on the information described above, I feel comfortable implementing the model and solving for the school assignment which minimizes SBUSD student travel."

Appendix B. Tables of Survey Results

B.1. Full Course Populations

Table B.1 provides the distributions of response selections from all respondents within both course populations across all 16 questions. For each question that was asked to all respondents (Q1 through Q9, Q15, and Q16), the statistic and p-value for a chi-squared test across the two course distributions are reported. Please note that Q2 and Q15 each received 155 responses from TOM, whereas all other questions were answered by all 156 respondents from

Table B.1. Summary Statistics of Selected Survey Responses for the Two Course Populations

	OMAI					TOM						
	S.D.	D.	N.	A.	S.A.	S.D.	D.	N.	A.	S.A.	χ^2	p-value
Q1	0.170	0.085	0.021	0.191	0.532	0.301	0.122	0.051	0.269	0.256	19.894	0.001
Q2	0.266	0.053	0.032	0.181	0.468	0.406	0.084	0.032	0.232	0.245	13.587	0.009
Q3	0.543	0.149	0.043	0.128	0.138	0.609	0.147	0.077	0.064	0.103	4.865	0.301
Q4	0.011	0.064	0.117	0.585	0.223	0.244	0.212	0.199	0.263	0.083	55.266	0.000
Q5	0.000	0.000	0.011	0.383	0.606	0.179	0.115	0.147	0.359	0.199	66.937	0.000
Q6	0.000	0.117	0.170	0.489	0.223	0.218	0.205	0.192	0.276	0.109	35.869	0.000
Q7	0.011	0.128	0.191	0.468	0.202	0.083	0.173	0.256	0.359	0.128	11.177	0.025
Q8	0.000	0.000	0.000	0.234	0.766	0.019	0.032	0.090	0.333	0.526	20.709	0.000
Q9	0.011	0.096	0.181	0.436	0.277	0.071	0.269	0.167	0.346	0.147	19.347	0.001
Q10	0.000	0.000	0.032	0.202	0.766	—	—	—	—	—	—	—
Q11	0.000	0.021	0.053	0.362	0.564	—	—	—	—	—	—	—
Q12	0.011	0.053	0.043	0.372	0.521	—	—	—	—	—	—	—
Q13	0.000	0.011	0.064	0.287	0.638	—	—	—	—	—	—	—
Q14	0.000	0.032	0.053	0.340	0.574	—	—	—	—	—	—	—
Q15	0.000	0.043	0.032	0.521	0.404	0.045	0.097	0.200	0.516	0.142	35.319	0.000
Q16	0.000	0.043	0.074	0.468	0.415	0.045	0.141	0.250	0.423	0.141	37.810	0.000

Note. S.D., strongly disagree; D, somewhat disagree; N, neither agree nor disagree; A, Somewhat agree; S.A., strongly agree.

TOM. All 94 respondents from OMAI answered all 16 questions.

B.2. Stratified by Prior Coding Experience

Then, Table B.2 provides the distributions of response selections from the nonstrong coder respondents within both course populations across all 16 questions. That is, the distributions shown here are calculated by removing responses that selected "strong agree" for Q1. For each question that was asked to all respondents (Q1 through Q9, Q15, and Q16), the statistic and p-value for a chi-squared test across the two course distributions are reported. All questions received 44 responses from the OMAI respondents without strong coders and 116 responses from the TOM respondents without strong coders, except for Q2, which received 115 responses from TOM without strong coders.

Finally, Table B.3 provides the distributions of response selections from the exclusively strong coder respondents within both course populations across all 16 questions. That is, the distributions shown here are calculated by including only the responses that selected "strongly agree" for Q1. For each question that was asked to all respondents (Q1 through Q9, Q15, and Q16), the statistic and p-value for a chi-squared test across the two course distributions are reported. All questions received 50 responses from OMAI

Table B.2. Summary Statistics of Selected Survey Responses for the Two Course Populations Without Respondents with a Strong Prior Coding Background

	OMAI without strong coders					TOM without strong coders						
	S.D.	D.	N.	A.	S.A.	S.D.	D.	N.	A.	S.A.	χ^2	p-value
Q1	0.364	0.182	0.045	0.409	—	0.405	0.164	0.069	0.362	—	0.671	0.880
Q2	0.523	0.091	0.045	0.227	0.114	0.517	0.095	0.043	0.267	0.078	0.684	0.953
Q3	0.727	0.091	0.068	0.091	0.023	0.707	0.164	0.086	0.043	0.000	5.257	0.262
Q4	0.023	0.045	0.227	0.545	0.159	0.302	0.224	0.198	0.233	0.043	32.493	0.000
Q5	0.000	0.000	0.023	0.409	0.568	0.233	0.147	0.147	0.328	0.147	43.246	0.000
Q6	0.000	0.045	0.295	0.545	0.114	0.284	0.224	0.181	0.241	0.069	30.161	0.000
Q7	0.023	0.114	0.227	0.477	0.159	0.112	0.190	0.267	0.362	0.069	8.040	0.090
Q8	0.000	0.000	0.000	0.182	0.818	0.026	0.043	0.112	0.345	0.474	17.430	0.002
Q9	0.000	0.136	0.182	0.432	0.250	0.086	0.293	0.198	0.328	0.095	13.532	0.009
Q10	0.000	0.000	0.045	0.250	0.705	—	—	—	—	—	—	—
Q11	0.000	0.023	0.068	0.341	0.568	—	—	—	—	—	—	—
Q12	0.000	0.045	0.045	0.341	0.568	—	—	—	—	—	—	—
Q13	0.000	0.000	0.114	0.295	0.591	—	—	—	—	—	—	—
Q14	0.000	0.023	0.091	0.341	0.545	—	—	—	—	—	—	—
Q15	0.000	0.068	0.068	0.477	0.386	0.052	0.104	0.243	0.522	0.078	26.350	0.000
Q16	0.000	0.068	0.136	0.409	0.386	0.060	0.164	0.276	0.397	0.103	21.490	0.000

Note. S.D., strongly disagree; D, somewhat disagree; N, neither agree nor disagree; A, Somewhat agree; S.A., strongly agree.

Table B.3. Summary Statistics of Selected Survey Responses for the Two Course Populations for Only Respondents with a Strong Prior Coding Background

	OMAI strong coders					TOM strong coders						
	S.D.	D.	N.	A.	S.A.	S.D.	D.	N.	A.	S.A.	χ^2	p-value
Q1	—	—	—	—	1.000	—	—	—	—	1.000	0.000	1.000
Q2	0.040	0.020	0.020	0.140	0.780	0.077	0.051	0.000	0.128	0.744	2.008	0.734
Q3	0.380	0.200	0.020	0.160	0.240	0.325	0.100	0.050	0.125	0.400	4.235	0.375
Q4	0.000	0.080	0.020	0.620	0.280	0.075	0.175	0.200	0.350	0.200	16.413	0.003
Q5	0.000	0.000	0.000	0.360	0.640	0.025	0.025	0.150	0.450	0.350	14.107	0.007
Q6	0.000	0.180	0.060	0.440	0.320	0.025	0.150	0.225	0.375	0.225	6.858	0.144
Q7	0.000	0.140	0.160	0.460	0.240	0.000	0.125	0.225	0.350	0.300	1.489	0.685
Q8	0.000	0.000	0.000	0.280	0.720	0.000	0.000	0.025	0.300	0.675	1.345	0.510
Q9	0.020	0.060	0.180	0.440	0.300	0.025	0.200	0.075	0.400	0.300	5.510	0.239
Q10	0.000	0.000	0.020	0.160	0.820	—	—	—	—	—	—	—
Q11	0.000	0.020	0.040	0.380	0.560	—	—	—	—	—	—	—
Q12	0.020	0.060	0.040	0.400	0.480	—	—	—	—	—	—	—
Q13	0.000	0.020	0.020	0.280	0.680	—	—	—	—	—	—	—
Q14	0.000	0.040	0.020	0.340	0.600	—	—	—	—	—	—	—
Q15	0.000	0.020	0.000	0.560	0.420	0.025	0.075	0.075	0.500	0.325	7.193	0.126
Q16	0.000	0.020	0.020	0.520	0.440	0.000	0.075	0.175	0.500	0.250	9.792	0.020

Note. S.D., strongly disagree; D, somewhat disagree; N, neither agree nor disagree; A, Somewhat agree; S.A., strongly agree.

strong coders and 40 responses from TOM strong coders, except for Q2, which received 39 responses from TOM strong coders.

Endnote

[1] A sample syllabus can be found at https://faculty.marshall.usc.edu/Andrew-Daw/BUAD313_Syllabus-fall23.pdf.

References

AhmadiTeshnizi A, Gao W, Brunborg H, Talaei S, Lawless C, Udell M (2025) Optimus-0.3: Using large language models to model and solve optimization problems at scale. Preprint, submitted July 29, 2024, https://arxiv.org/abs/2407.19633.

Bachiri YA, Mouncif H, Bouikhalene B (2023) Artificial intelligence empowers gamification: Optimizing student engagement and learning outcomes in e-learning and MOOCs. *Internat. J. Engrg. Pedagogy* 13(8):4–19.

Bastani H, Bastani O, Sungu A, Ge H, Kabakcı Ö, Mariman R (2025) Generative AI without guardrails can harm learning: Evidence from high school mathematics. *Proc. Natl. Acad. Sci. USA* 122(26):e2422633122.

Belkina M, Daniel S, Nikolic S, Haque R, Lyden S, Neal P, Grundy S, Hassan GM (2025) Implementing generative AI (GenAI) in higher education: A systematic review of case studies. *Comput. Ed.: Artificial Intelligence* 8:100407.

Bien J, Mukherjee G (2025) Generative AI for data science 101: Coding without learning to code. *J. Statist. Data Sci. Ed.* 33(2):129–142.

Bien J, Burgos M, Cardon P, Carnevale P, Chen F, Guo Y, Gupta V, et al. (2024) Embracing AI in business education: A case study of USC Marshall School of Business. Preprint, submitted October 16, https://doi.org/10.2139/ssrn.4975641.

Bray RL (2025) A tutorial on teaching data analytics with generative AI. *INFORMS J. Appl. Anal.* 55(4):319–343.

Budzyń K, Romańczyk M, Kitala D, Kołodziej P, Bugajski M, Adami HO, Blom J, et al. (2025) Endoscopist deskilling risk after exposure to artificial intelligence in colonoscopy: A multicentre, observational study. *Lancet Gastroenterology Hepatology* 10(10):P896–P903.

Bynum ML, Hackebeil G, Hart WE, Laird CD, Nicholson BL, Siirola JD, Watson JP, Woodruff DL (2020) *Pyomo—Optimization Modeling in Python*, Springer Optimization and Its Applications, 3rd ed., vol. 67 (Springer, Cham, Switzerland).

Cachon G, Terwiesch C (2008) *Matching Supply with Demand*, vol. 20012 (McGraw-Hill Publishing, New York).

Cai J, Mandelbaum A, Nagaraja CH, Shen H, Zhao L (2019) Statistical theory powering data science. *Statist. Sci.* 34(4):669–691.

Chang CY, Kuo SY, Hwang GH (2022) Chatbot-facilitated nursing education. *Ed. Tech. Soc.* 25(1):15–27.

Chemers MM, Tze Hu L, Garcia BF (2001) Academic self-efficacy and first year college student performance and adjustment. *J. Ed. Psych.* 93(1):55–64.

Chen F, Samroengraja R (2000) The stationary beer game. *Production Oper. Management* 9(1):19–30.

Chen Z, Xiong P (2023) RSOME in Python: An open-source package for robust stochastic optimization made easy. *INFORMS J. Comput.* 35(4):717–724.

Chickering DM, Heckerman D (2003) Targeted advertising on the web with inventory management. *Interfaces* 33(5):71–77.

Corbett CJ, Van Wassenhove LN (1993) The natural drift: What happened to operations research? *Oper. Res.* 41(4):625–640.

Deslauriers L, McCarty LS, Miller K, Callaghan K, Kestin G (2019) Measuring actual learning versus feeling of learning in response to being actively engaged in the classroom. *Proc. Natl. Acad. Sci. USA* 116(39):19251–19257.

Dobson G, Shumsky R (2006) Web-based simulations for teaching queueing, Little's law, and inventory management. *INFORMS Trans. Ed.* 7(1):106–124.

Dong L, Tang X, Wang X (2025) Examining the effect of artificial intelligence in relation to students' academic achievement in classroom: A meta-analysis. *Comput. Ed.: Artificial Intelligence* 8:100400.

Duckworth AL, Yeager DS (2015) Measurement matters: Assessing personal qualities other than cognitive ability for educational purposes. *Ed. Res.* 44(4):237–251.

Dunning I, Huchette J, Lubin M (2017) Jump: A modeling language for mathematical optimization. *SIAM Rev.* 59(2):295–320.

Dunning I, Gupta V, King A, Kung J, Lubin M, Silberholz J (2015) A course on advanced software tools for operations research and analytics. *INFORMS Trans. Ed.* 15(2):169–179.

Easton JB (1967) The early editions of Robert Recorde's ground of Artes. *Isis* 58(4):515–532.

Eckstein J, Riedmueller ST (2002) YASAI: Yet another add-in for teaching elementary Monte Carlo simulation in Excel. *INFORMS Trans. Ed.* 2(2):12–26.

Evans JR (2000) Spreadsheets as a tool for teaching simulation. *INFORMS Trans. Ed.* 1(1):27–37.

Falchikov N, Boud D (1989) Student self-assessment in higher education: A meta-analysis. *Rev. Ed. Res.* 59(4):395–430.

Fourer R, Gay DM, Kernighan BW (1990) AMPL: A mathematical programming language. *Management Sci.* 36(5):519–554.

Gorman MF (2021) INFORMS Journal on Applied Analytics editor's statement: The critical role of applied research in analytics. *INFORMS J. Appl. Anal.* 51(1):1–5.

Grant M, Boyd S, Ye Y (2006) Disciplined convex programming. Liberti L, Maculan N, eds. *Global Optimization*, Nonconvex Optimization and Its Applications, vol. 84 (Springer, Boston), 155–210.

Griffin P (2007) The use of classroom games in management science and operations research. *INFORMS Trans. Ed.* 8(1):1–2.

Hill RR (2002) Process simulation in Excel for a quantitative management course. *INFORMS Trans. Ed.* 2(3):75–84.

Howson AG (1982) *A History of Mathematics Education in England* (Cambridge University Press, Cambridge, UK).

Irwin V, Wang K, Tezil T, Zhang J, Filbey A, Jung J, Mann FB, Dilig R, Parker S (2023) Report on the condition of education 2023. Report No. NCES 2023-144, National Center for Education Statistics, U.S. Department of Education, Washington, DC.

Jensen PA, Bard JF (2002) *Operations Research: Models and Methods* (John Wiley & Sons, Hoboken, NJ).

Kan AHR (1989) The future of operations research is bright. *Eur. J. Oper. Res.* 38(3):282–285.

Kruger J, Dunning D (1999) Unskilled and unaware of it: How difficulties in recognizing one's own incompetence lead to inflated self-assessments. *J. Personality Soc. Psych.* 77(6):1121–1134.

Lawless C, Schoeffer J, Le L, Rowan K, Sen S, St Hill C, Suh J, Sarrafzadeh B (2024) "I want it that way"': Enabling interactive decision support using large language models and constraint programming. *ACM Trans. Interactive Intelligent Systems* 14(3):1–33.

Liberatore MJ, Luo W (2010) The analytics movement: Implications for operations research. *Interfaces* 40(4):313–324.

Lofberg J (2004) YALMIP: A toolbox for modeling and optimization in MATLAB. *2004 IEEE Internat. Conf. Robotics Automation* (IEEE, Piscataway, NJ), 284–289.

Martin K (2010) Tutorial: COIN-OR: Software for the OR community. *Interfaces* 40(6):465–476.

Mason AJ (2012) OpenSolver - An open source add-in to solve linear and integer progammes in Excel. Klatte D, Lüthi HJ, Schmedders K, eds. *Oper. Res. Proc.* 2011 (Springer, Berlin, Heidelberg), 401–406.

Mason AJ (2013) SolverStudio: A new tool for better optimisation and simulation modelling in Excel. *INFORMS Trans. Ed.* 14(1): 45–52.

Mortenson MJ, Doherty NF, Robinson S (2015) Operational research from Taylorism to terabytes: A research agenda for the analytics age. *Eur. J. Oper. Res.* 241(3):583–595.

Multon KD, Brown SD, Lent RW (1991) Relation of self-efficacy beliefs to academic outcomes: A meta-analytic investigation. *J. Counseling Psych.* 38(1):30–38.

National Council of Teachers of Mathematics (1989) Curriculum and evaluation standards for school mathematics: Report of the National Council of Teachers of Mathematics' Commission on Standards. Report, National Council of Teachers of Mathematics, Reston, VA.

National Governors Association Center for Best Practices, Council of Chief State School Officers (2010) *Common Core State Standards for Mathematics* (National Governors Association Center for Best Practices, Council of Chief State School Officers, Washington, DC).

Pachamanova D (2006) Introducing integer modeling with Excel solver. *INFORMS Trans. Ed.* 7(1):88–98.

Pasin F, Giroux H (2011) The impact of a simulation game on operations management education. *Comput. Ed.* 57(1):1240–1254.

Ramamonjison R, Yu TT, Li R, Li H, Carenini G, Ghaddar B, He S, et al. (2022) NL4Opt competition: Formulating optimization problems based on their natural language descriptions. Ciccone M, Stolovitzky G, Albrecht J, eds. *Proc. NeurIPS 2022 Competitions Track*, Proceedings of Machine Learning Research, vol. 220 (PMLR, New York), 189–203.

Recorde R (1543) *The Grounde of Artes: Teaching the Perfect Worke and Practise of Arithmeticke* (Early English Books Online, London).

Sitzmann T, Ely K, Brown KG, Bauer KN (2010) Self-assessment of knowledge: A cognitive learning or affective measure? *Acad. Management Learn. Ed.* 9(2):169–191.

Snider B, Balakrishnan J (2013) Lessons learned from implementing web-based simulations to teach operations management concepts. *INFORMS Trans. Ed.* 13(3):152–161.

Sodhi MS, Tang CS (2008) The OR/MS ecosystem: Strengths, weaknesses, opportunities, and threats. *Oper. Res.* 56(2):267–277.

Wang L, Zhao M (2024) Can artificial intelligence technology promote the improvement of student learning outcomes?—Meta analysis based on 50 experimental and quasi experimental studies. *Proc. 3rd Internat. Conf. Ed. Innovation Multimedia Tech., EIMT 2024* (EAI, Ghent, Belgium).

Wolfe EW (2010) What impact does calculator use have on test results? Technical report, Pearson Education, London.

Xiao Z, Zhang D, Wu Y, Xu L, Wang YJ, Han X, Fu X, et al. (2024) Chain-of-experts: When LLMs meet complex operations research problems. Kim B, Yue Y, Chaudhuri S, Fragkiadaki K, Khan M, Sun Y, eds. *Twelfth Internat. Conf. Learn. Representations* (OpenReview.net), 48519–48537.

Yan L, Sha L, Zhao L, Li Y, Martinez-Maldonado R, Chen G, Li X, Jin Y, Gašević D (2024) Practical and ethical challenges of large language models in education: A systematic scoping review. *British J. Ed. Tech.* 55(1):90–112.

Zheng L, Niu J, Zhong L, Gyasi JF (2023) The effectiveness of artificial intelligence on learning achievement and learning perception: A meta-analysis. *Interactive Learn. Environments* 31(9): 5650–5664.